AN IMPLEMENTATION OF NATURAL LANGUAGE USER INTERFACE IN SMART HOME SYSTEM

NGOI SE KENG

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SOFTWARE ENGINEERING

FACULTY OF COMPUTER SCIENCE AND
INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

NOVEMBER 2010

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

[redacted]

Name of Candidate: NGOI SE KENG

Registration/Matric No: WGC 030049

Name of Degree: MSE

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

AN IMPLEMENTATION OF NATURAL LANGUAGE USER INTERFACE IN SMART HOME SYSTEM

Field of Study:

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

[redacted]

11/11/2010
Date

Subscribed and solemnly declared before,

11/11/2010

Witness's Signature

Date

[redacted]

Name:
Designation: DR. ZARINAH MOHD KASIRUN
Senior Lecturer
Dept. of Software Engineering
FCSIT, Universiti of Malaya

Abstract

Smart Home system has facilitated the control of home appliances through computer system. However, the physical interactions between the user and existing graphical user interface such as pointing and clicking button could be a challenge to elderly, whom are generally having vision and motor impairment. This project specifically focuses on the development of a user interface that does not require physical interaction. In this regard, a prototype smart home system namely Natural Language Instruction Driven System (NLIDS) that employed natural language (speech) based user interface has been developed. The prototype system consists of three modules, which are Input & Response Module, Language Processing Module, and Visualization Module. Generally, Input & Response Module is responsible for receive speech instruction as well as provide response to user; Language Processing Module applies business logic to the received instruction; Visualization Module simulates the result of an instruction in computer animation. This project had successfully integrated several state-of-the art technologies and tools such as Java Speech Application Programming Interface, LinkGrammar, Java3D, WordNet in the development of prototype system. The evaluation carried out at the end of the development shows that the participants in the evaluation are believe that natural language (speech) interface is easier to use, easier to remember, and easier to learn than the graphical user interface.

Acknowledgement

Firstly, I wish to thank my first supervisor, Pn. Norisma Idris for being so patient, supportive and helpful during the development of this dissertation. She was always there despite her packed schedule, to provide guidance and always leading me back to the right path.

Besides, I am heartily thankful to my second supervisor, Dr. Zarinah Kasirun, whose encouragement, guidance and support enabled me to revise and amend this dissertation. Despite the tight schedule and short notification, her willingness to discuss and review this dissertation contributed tremendously to the completion of rewriting this dissertation

My sincere gratitude to my family members, especially to my wife for keeping me going through difficult times. Their encouragement throughout the whole time enabled me to strive further.

Last but not least, my friends, who were always there when I needed them. I never could have gotten through without their help.

Table of Contents

List of Figures

List of Tables

Chapter 1.0 Introduction

"Language is one of the fundamental aspects of human behavior and is crucial component of our live." (Allen, 1995)

1.1 Introduction to Smart Home

Smart Home is a technology that will improve lifestyle, promoting home comfort, convenience, security and entertainment (Altrich, 2003). It is a natural extension of current electronic, information and communication technologies (Chan et al., 2008). According to Begg (2006), Smart Home has been the topic of interest for researchers since the 1940s. The major aim has been to automate control of the appliances and systems within the home environment that can work independently.

Advances in technology have facilitated the usages of an increased number of automated devices in our home environment to make the daily tasks easier. However, as more and more devices are used in the home, the problem of controlling and managing these devices increases exponentially. Unlike humans, devices do not have the inherent understanding of the environmental condition and also do not know what actions are to be selected under which situations (Wu, 2001).

In Malaysia, house design, internal space and layout must be planned properly to enable for creation of the smart home and interactive functions at home for business, private and government related transactions, education and recreation. The role of the automation system as the house's brain links every electronic function in the housing estate. The gadgets in smart homes also link to security, communication, lighting and entertainment. Automation would enable the owner to control the use of the equipment at home, before reaching home or from a distant place (Ministry of Housing and Local Government Malaysia, 1999).

There are a number of housing development projects featured with certain aspects of Smart Home functionality have been developed nowadays. For instance, Bayswater Resort Condominium Penang developed by IJM Properties Sdn Bhd. In the project, every unit in the condominium is equipped with a central control panel which acts as touch screen interface to control electronic devices such as light and fan. In addition, there is remote control ability to the smart home system via internet (IJM, 2008).

Another example is Vista Millennium Apartment in Puchong. The apartment comes with a smart home management system that allowed home owner to control some of the electronic appliance remotely. In addition, the lighting system can be configured to set exquisite tone to suit mood such as mood for reading, mood for listening, and mood for romance (iProperty, 2008).

Obviously, advances in Smart Home technology have facilitated the controlling of an increased number of electronic devices in our home environment to make the daily tasks easier.

1.2 Background of Problem

According to "The Population and Housing Census 2000" (Department of Statistic Malaysia, 2001), the proportion of population of Malaysia below 15 years of age in Census 2000 was 33.3% compared to 36.7% in 1991. Conversely, the proportion of population 65 years and above for Malaysia in Census 2000 was recorded at 3.9% compared to 3.7% in 1991. Consequently, the median age for Malaysia as a whole increased from 21.9 years in 1991 to 23.6 years in 2000. All these different age parameters point clearly towards a continuation of the trend of population ageing in Malaysia.

Smart Home technology offers the prospect of significant improvements in the quality of life and levels of independence for everyone especially elderly (Orpwood et al., 2005). However, the existing user interface such as touch screen, icon based menu, physical remote control, and Dual Tone Multiple Frequency of telephone require sort of physical interaction when accessing such interfaces. For instance, clicking icon on a graphical interface requires hand movement as well as vision focusing. This could be a challenge to elderly due to their vision and mobility impairment.

In addition, user of the mentioned user interfaces need to be trained prior to use the interfaces. According to Park (2003), adults as they get older experiences a wide range of age related impairments including loss of vision, hearing, memory and mobility, the combined effects of which contribute to loss of confidence and difficulties in orientation and absorption of new information.

1.3 Research Problem

As pointed out by Ghorbel et al. (2004), vision and mobility impairment is the limitation of elderly people in accessing Smart home. Besides that, according to Mynatt (2000), user interface that is more simplified in the interaction with users shall be implemented in Smart Home system to ease the user especially elderly people in accessing the system.

In addition, the training for the system dependent instruction is also a challenge to elderly for using smart home system. The limitations of the existing user interfaces of Smart Home systems which are concern in our research are as below:

1. Physical interaction when accessing the existing systems has restricted elderly especially those are mobility impaired from using the systems for improving their quality of life.
2. The vision of elderly is generally poorer. Therefore, it is a challenge for them to interact with the existing Smart Home system which its interface is implemented in window, icon, mouse and pointer (WIMP).
3. The instructions of the existing Smart Home systems are system dependent. Every system has their sets of predefined instruction. Hence, user is required to be trained and retrained prior to use the system.
4. The interaction between systems and users (humans) do not reflect the interaction between human beings. Human using language and gesture to communicate with each other rather than through WIMP. This has confined the acceptance of the system especially by elderly and those who are not computer literacy.

1.4 Objectives of Study

Generally, the main objective of the study is to develop a speech based user interface for Smart Home system to overcome the limitation mentioned in the earlier section. According to Allen (1995), natural language or speech is believed as one of the efficient mediums for the interaction between computer and users since it is one of the channels that human kind used to communicate with each others. Thus, the research aims to design and develop a speech user interface that process speech based instruction for Smart Home system.

Besides that, this research is also aimed to serve as a reference for research that interested to integrate speech user interface in their domain systems.

The objectives of this research are as follows:

1. To develop a speech based user interface on Smart Home system that will overcome the limitations mentioned in the previous section.
2. To demonstrate the possibility of using speech as an input in Smart Home system by developing a prototype that process speech based instruction.
3. To integrate the state-of-art speech processing technology into the user interface of Smart Home systems to enhance the user's experience in accessing such system.

1.5 Scope of Project

As discussed, the research is not intended to replace the existing interface of Smart Home system, but it is to add-on a speech processing interface to the system conversely. As a result, a prototype system that is able to process speech instruction is expected to be developed at the end of this research.

Generally, the proposed system is designed to instruct a virtual smart home system which consists of virtual electronic devices in a virtual environment such as light, fan, television, and radio. The user of the system will be able to instruct the system to perform specific tasks such as changing television channels, changing the speed of the fan and turning off lights by using speech instruction. Since our scope is excluded from designing real electronic circuits, therefore, the respective response will be presented to user in animation form.

As an overview, the system is named as Speech for Smart Home (SSH). SSH consists of front-end and back-end portions. The front-end is the user interface that is equipped with an input dialog and graphical view area. Meanwhile, the back-end or also known as the "processor" of the system consists of a parser, an object of animation generation and a set of virtual world operations database. The overview or conceptual diagram of the system is shown in Figure 1.1 below:

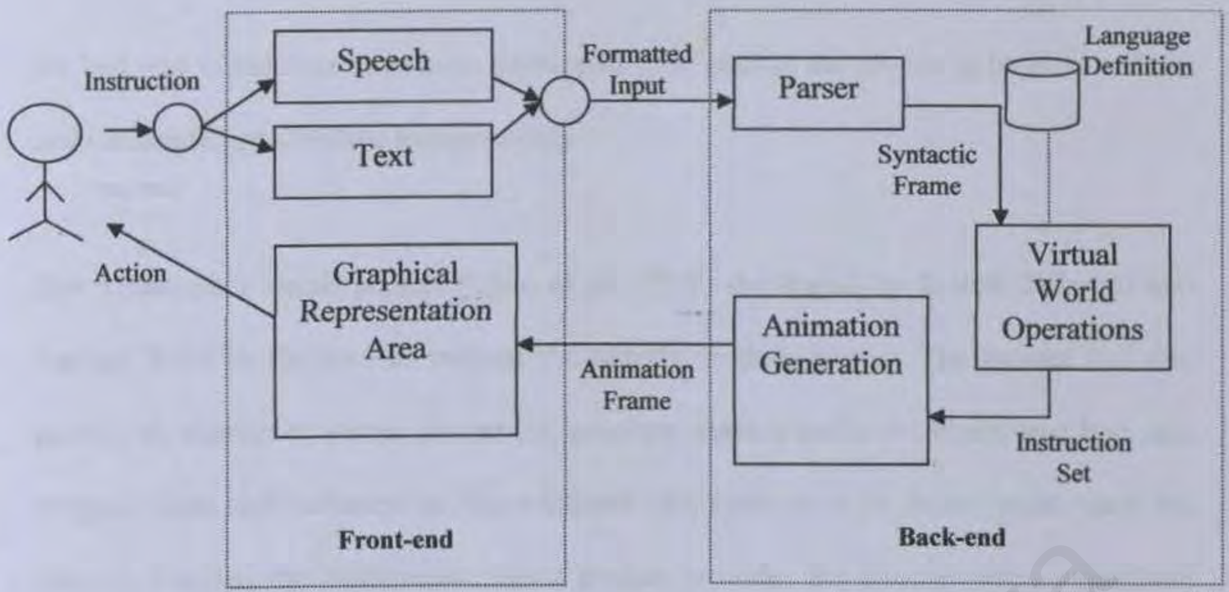


Figure 1.1: Conceptual Diagram of the Proposed System

1.6 Significance of Project

Several similar researches for support elderly living autonomously have been carried out, for instance, AmbienNet (Abascal et al., 2008) which is an ongoing project that developed in the Universities of the Basque Country to achieve the mobility freedom of elderly. In the project, a localization system that monitor network of sensors and intelligent devices such as wheelchairs has been developed. The system acts as an assisted navigation system that guiding elderly to detect obstacle and distinguishing them from the floor.

Besides that, Intelligent Sweet Home (Park et al., 2007) that includes a smart "robotic" bed fitted with a "smart hand", an intelligent wheelchair to transfer the elderly between the bed and wheelchair have been developed. The goal of the project is to assist elderly and the handicapped to live independently.

The Millennium Home project (Chan et al., 2008) developed by British Telecom and Anchor Trust in England to support the elderly in their homes. The project includes passive IR sensors to detect movement, pressure sensors under the chairs and bed, and burglar alarm-style sensors on the windows and front door to detect when they are opened. Besides, the Millennium Home project provides the elderly with a quick and easy mechanism to cancel false alarms and more rapid response to a real emergency.

Smart Home systems improved living conditions for most of the users. Unfortunately, the levels of independence for elderly who require support with both physical and cognitive functions are not well taken due to the vision and motor required interface. Speech interface is a comfort interface as it needs no special devices to handle such as a mouse, wands and data gloves and has high accessibility for diverse users including the elderly and the disabled (Kim, 2008).

According to Roy et al. (2000), elderly people often have difficulties interacting through unfamiliar means, such as keyboards and computer screens. It is therefore of great importance that a system communicates in ways familiar to elderly people. To that end, spoken interaction with the system is absolutely essential. In addition, speech is one of the channels that used by human to communicate with each others. Hence, the training for using speech instruction is relatively low compare to using predefined system instruction.

Lastly, the project is response to the appeal of the effort of Malaysia government to tackle the challenge and problem the aging population. Several researches on the Smart Home implementation have been carried out under Malaysia Research & Education Network, Ministry of Energy, Communication & Multimedia. For instance, the research titled "Smart Homes: Applications, control, devices, security" (MYREN, 2004).

1.7 Limitation on the Research

Information processing from speech is a complicated subject and advances are required in many subfields of natural language processing (Allen, 1995). Therefore, the input to the proposed system must be domain specific, which within the smart home domain. In other words, the system will not process any instruction that irrelevant or does not have knowledge with it.

In addition, although the system is designed to be as robust as possible, it might still fail to process ill formed input, such as ellipsis. For instance, a scenario whereby a user's command: "turn the radio", without the adjective on or off. In this case, the command will be ignored and rejected.

Other than that, as stated in the earlier section, this research is to demonstrate the possibility of using speech in smart home systems rather than enhancing existing speech processing techniques. Therefore, the limitations of the existing techniques such as difficulty in word boundary detection, word sense disambiguation, syntactic ambiguity, and ellipsis will be expected by the proposed system.

In general, the limitations of the proposed system are suggested as follows:

1. Virtual domain presentation - The output of the proposed system is represented in computer graphical form. In other words, the proposed system does not provide any real interactions with physical devices.
2. Language limitation - English is used and referred to as "speech" by the system due to the constrained of English parser used in the research. In other words, the "speech" to the system is limited to English only.
3. Platform dependent - The system is developed within Microsoft Windows environment and it does not run on other types of platform such Solaris or Macintosh.
4. Problems in existing parser - Since the research is not focusing on improving the existing parser, all problems in the existing parser such as difficulty in word boundary detection, and word sense disambiguation might contribute some limitations to the system.
5. Speech recognition problem - The system employs an existing speech recognition engine that required to be trained prior to use the system.

1.8 Outline of Research Report

This research report is organized into the following chapters:

Chapter One: Introduction

This chapter provides the introduction and requirement specification related to the research. It is organized into several sections as project background, problem description, objective study, scope of the project, significance of the project, limitation of the project, outline of research report, and the summary of the chapter.

Chapter Two: Literature Review

This chapter reviews on related systems as well as the techniques employed by the existing systems. In addition, we would provide the introduction to the tools which are used to develop the proposed system.

Chapter Three: Methodology

This chapter introduces and outlines the methodology use by the NLIDS. It will be discussed in two phases: the problem definition phase, and the development of the NLIDS phase.

Chapter Four: System Analysis

This chapter examines and analyzes the functional requirements and functionality for NLIDS.

Chapter Five: System Design

This chapter provides the design of the proposed system from three layers which are Input & Response Module, Language Processing Module, and Visualization Module. Besides that, the functionality for each module will be captured using UML diagrams.

Chapter Six: System Implementation

This chapter discusses the processes involved in the realization of system design. On top of that, this chapter also analyzes the configuration and installation of required software.

Chapter Seven: System Testing and Evaluation

This chapter documents the testing that carried out against the proposed system. Besides that, the evaluation to the system is discusses as well.

Chapter Eight: Conclusion and Future Works

This chapter discusses the system constraints and bottlenecks, further work that may be carried out in the future, and conclusion of the NLIDS project.

1.9 Conclusion

The interaction between system and user is believed will be simplified by using speech based instruction. In addition, the training of using speech based instruction system is relatively less than training of using computer dependent instruction since speech is the natural way of human being communicating with each other.

Chapter 2.0 Literature Review

2.1 Introduction

In this chapter, review on existing researches in an attempt to address the related issues pertaining to the subject of this research has been carried out. In addition, several existing speech user interface based are also been reviewed to identify the appropriate implementation techniques that could be employed in this research.

The chapter is organized into two parts. In the first part, we reviewed existing smart home systems and related speech user interface based system as our empirical studies. While in the second part, we provided a short overview of tools and environment that would be used to develop the proposed system.

2.2 Review of Smart Home Systems

Several Smart Home Systems such as Smart Home Management Systems (Lee et. al., 2007), Smart Home Microcontroller: Telephone Interfacing (Leong et. al., 2006), and Smart Home Control via PDA (Ringbauer, 2005) have been chosen for review. The main purpose of reviewing the mentioned systems is because all the mentioned systems implementing popular type of user interface which are instant messaging interface (web based), telephony interface (signal based), and PDA interface (touch screech based). In addition, the chosen systems provide remote functionality to operate electronic appliances in a smart home without any physical interaction with the appliances which is a requirement of our proposed system. Based on the research interest, only the user interface related components of the selected systems will be focused and reviewed.

2.2.1 Smart Home Management System

Smart Home Management system (SHMS) is a home automation system developed by Department of Computer Engineering, Kyungpook National University. Several technologies including Jini which is an open architecture to create network-centric services (Sun Microsystems, 2008) and instant messaging technology are used in the system to provide a uniform graphical user interface. Hence, user of the system is provided the functionality for monitoring and controlling their home appliances via instant messenger while the output from the system will be sent to user via push technology (Lee et. al., 2007). The overview of the SHMS is shown in Figure 2.1.

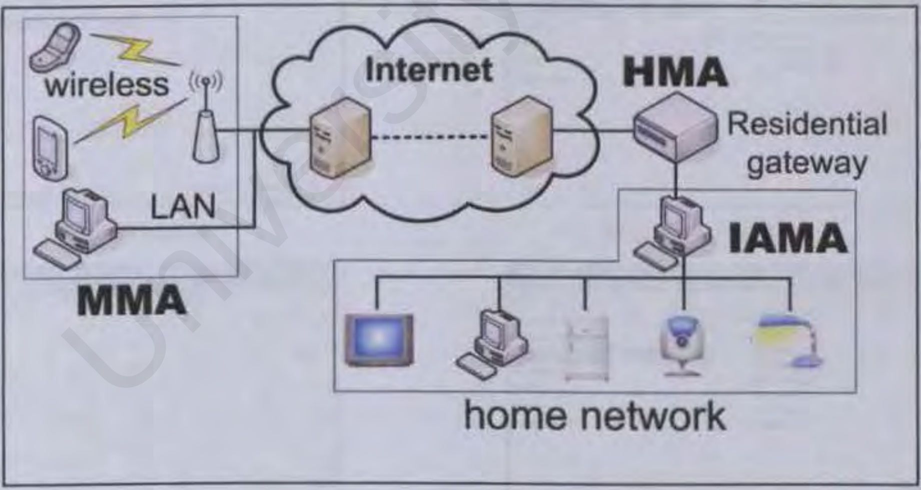
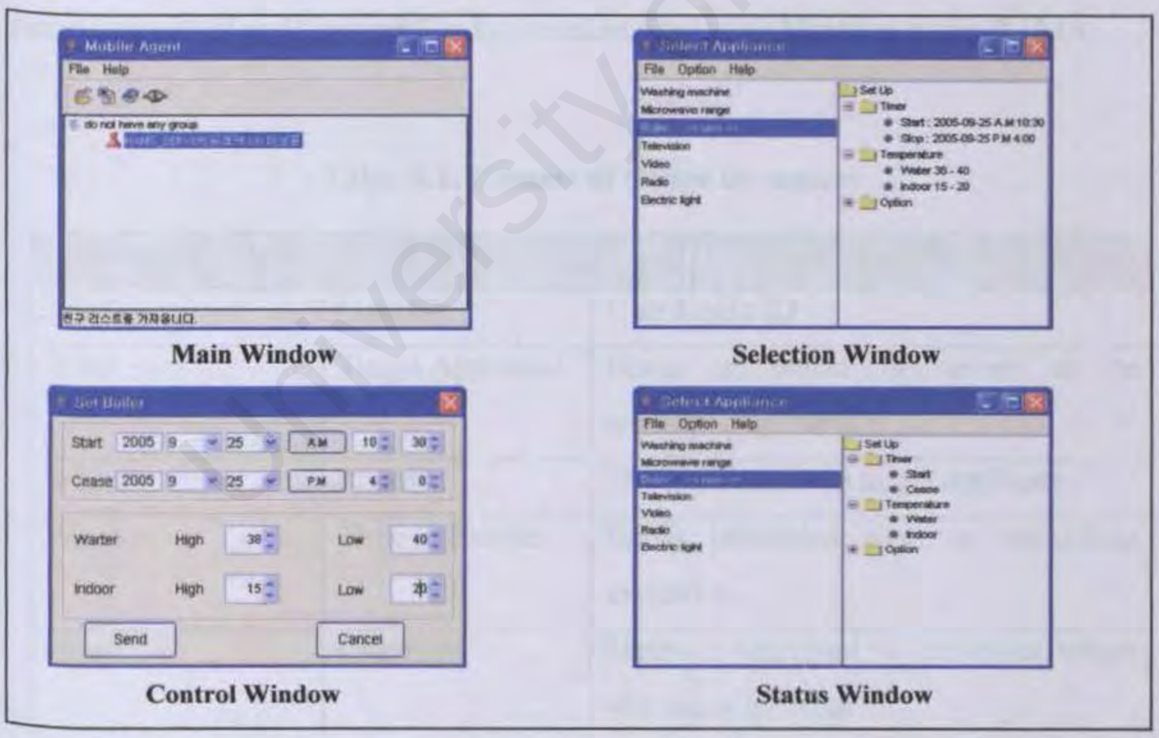


Figure 2.1: Architecture of SHMS (Lee et. al., 2007)

Generally, the system consists of 3 main parts that are related to user interface processing. They are:

i. ***Mobile Messenger Agent (MMA)***

The MMA is served as the graphical user interface of SHMS. The agent is designed to be executed on personal computer or portable devices that working under Microsoft Window platform. There are several windows built for the MMA such as main windows that responsible for logon verification, selection window that used for appliance selection, status window that displays status of the managed appliances and the command windows that received input from user via mouse click as displayed in Figure 2.2 below. When a command is received at the MMA, it will be forwarded to Home Messenger Agent for further processing.



Start : 2005-09-25 A.M 10:30
Stop : 2005-09-25 P.M 4:00
Water 35 - 40
Indoor 15 - 20

Start 2005 9 25 AM 10 30
Cease 2005 9 25 PM 4 0
Warter High 38 Low 40
Indoor High 15 Low 20

Start
Cease
Water
Indoor

Figure 2.2: Windows of Mobile Messenger Agent (Lee et. al., 2007)

ii. Home Messenger Agent (HMA)

The HMA is the agent that acts as the interface between the MMA and the IAMA. Generally, the main responsible of HMA is to translate the user command received from MMA to an internal format, namely Order Document XML before forwarded to IAMA for further processing. The Order document is the context which consists of 4 Ws (who, what, where, when, and why) and 1H (how) that means certain behavior information of user as shown in Table 2.1. In addition, the HMA is also used to manage user preference such as adding nick name and changing contact information.

Whenever an appliance related command is received from MMA, the HMA shall determine the target appliance requested as well as the action need to be performed on to the appliance. Once the target and action has been determined, the Order Document will then be generated and forwarded to Information Appliance Manager Agent (IAMA).

Table 2.1: Context of Order Document

Behavior Context	Concept	Information Factor
Who	User ID	User Login ID
What	Target Appliance	Name of object or device to be controlled by 'who'
Where	Room	The location of the target appliance
When	Time to Execute	Either scheduled time or immediate execution
How	Operation	Methods described in the event which will occur in 'what'

2.2.2 Smart Home Microcontroller: Telephone Interfacing

The project is to enable the user to control electronic home appliances remotely using any touch-tone phone. The basic idea of this project is to take advantage of the vast network of telephone lines and the proliferation of cell phones to extend human's reach and possibilities (Leong et. al., 2006).

Generally, dual tone multi frequency (DTMF) is used as the command in controlling electronic appliances via telephone's network in the project. The typical scenario for the control work flow is as follow: The user dials the home telephone number like an ordinary telephone call and if nobody picks the call up to 3 rings, then the system picks up the call and offered a voice menu. The user will then is required to choose an item from the menu by pressing a button on the phone keypad. By pressing a button on the phone, a DTMF signal generated which, through the telephone network, will reach the system at home. The system will recognize the received signal and based on that will switch on/off the chosen appliance either through a physical wiring connection or infrared connection.

Once the requested command has been performed, the system will transmit the requested command serially to the PC for logging purpose. In addition, the PC triggers the GSM mobile phone to send a SMS to the pre-defined telephone number in order to notify him/her the action taken on the system.

The designed system has several components such as telephone interfacing circuit, Two-Wire to Four-Wire Converter, microcontroller, DTMF decoder, voice chip, and Infrared Transmitter and Receiver as shown in Figure 2.4:

i. Telephone Interfacing Circuit

The functions of telephone interfacing circuit are to detect incoming ringing signals, take the line and hang up. It is consists of ring detection circuit and on/off-hook circuit. It is also equipped with protection circuit to protect the system from high voltage transient that might occur on.

ii. Two-Wire to Four-Wire Converter

Since telephone line is full duplex medium, therefore, a two-wire to four-wire converter which converts the single pair of wire into separate transmit and receive audio paths for the transmission of two channels of information in opposite directions is required.

iii. Microcontroller

Microcontroller is responsible to control the functions of the other components in the system. It controls the system by performing the following functions at specified time: Detecting ring detection circuit output, switching on/off the On/Off-hook circuit, playing/disabling the voice chip, interpreting the output of DTMF decoder, switching the controlled appliance On/Off, and communicating serially with PC.

iv. DTMF Decoder

The DTMF decoder decodes DTMF signals and represents them in a sequence of binary numbers. The binary number will then be used to determine the target appliance and the requested action.

v. Voice Chip

The voice menu offered to the user is recorded in the voice chip. In the system, voice chip plays the recorded message to the caller for different scenarios.

vi. Infrared Transmitter and Receiver

Infrared (IR) is used to remotely control the home appliances in the system.

In general, the reviewed Smart Home system with telephone interface is the least flexible system in the enhancement of supporting new controlling device since the enhancement involves both hardware and software changes. In addition, pressing telephone's button is a challenge to visually and mobility impaired elderly.

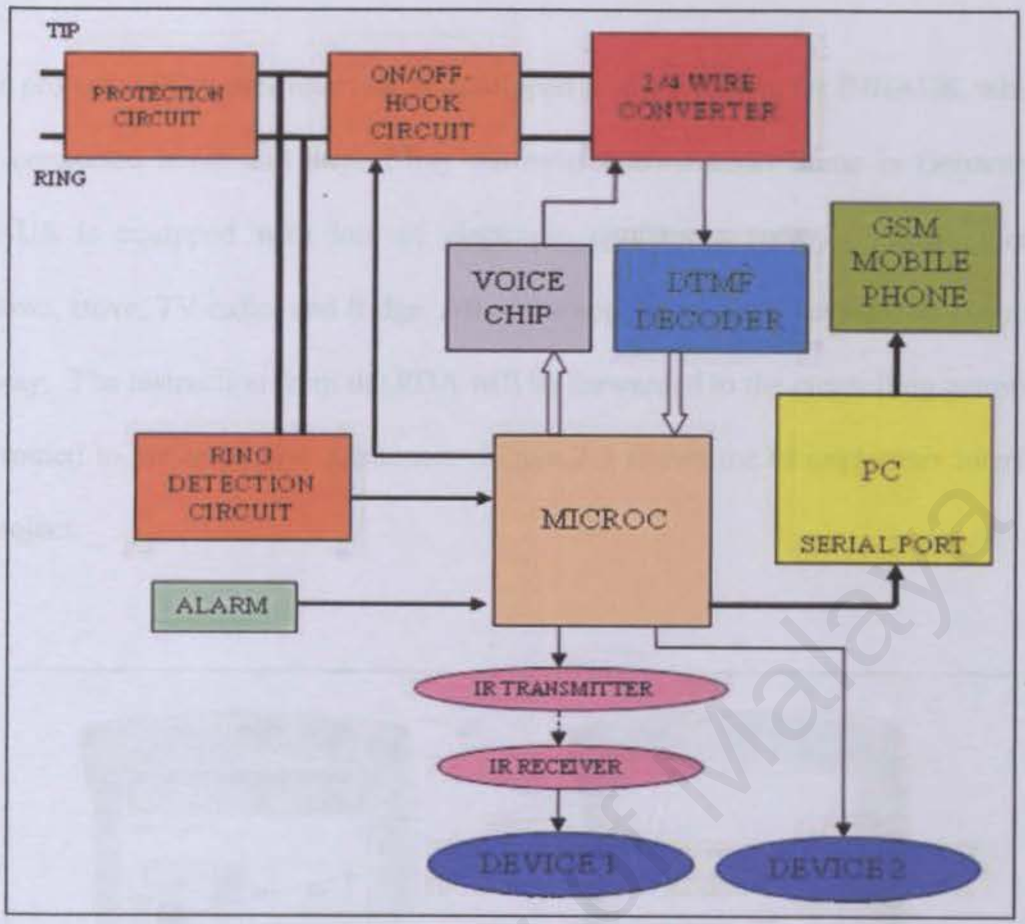
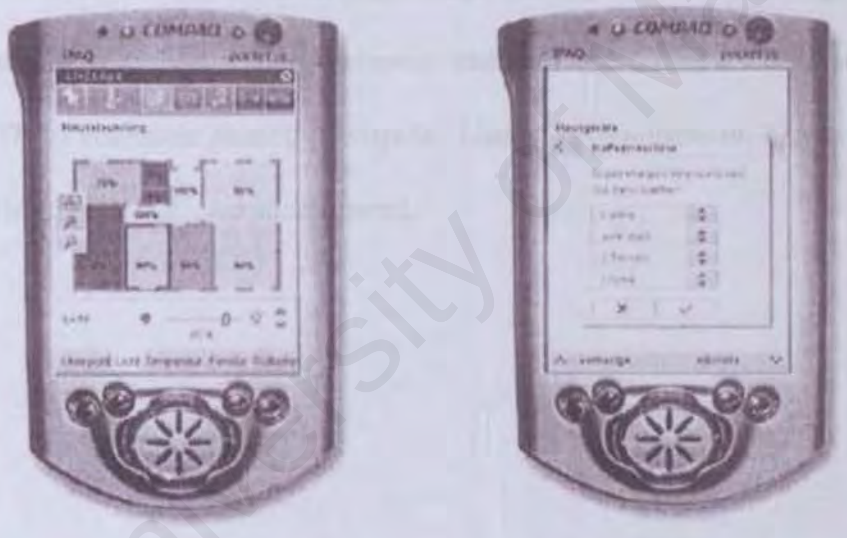


Figure 2.4: Block Diagram of Smart Home Microcontroller (Leong et al., 2006)

2.2.3 Smart Home Control via PDA

Smart Home Control via PDA is a multi-device user interface design with focus on usability issues developed in the research project LIVEEfutura funded by the German Ministry of Education and Research (Seffah et al., 2004). Basically, LIVEfutura is a project dealing with the specification and prototypical implementation of an integrated overall home control concept covering all relevant application areas: home automation, audio video control, home appliances, home computer network, home telecommunication network, and even the private car (Ringbauer, 2006).

In the project, a PDA user interface is developed as the interface for INHAUS, which is a fully connected home and is probably the most known smart home in Germany. The INHAUS is equipped with lots of electronic appliances such as Lights, electronic windows, stove, TV radio, and fridge. All of the appliances are connected to a controlling gateway. The instruction from the PDA will be forwarded to the controlling gateway and then routed to the respective appliance. Figure 2.5 shows the example user interface of the project.



Main Screen **Coffee Machine Screen**

Figure 2.5: Example Screen of Smart Home Control via PDA

2.2.4 Comparison of the Smart Home Systems

Table 2.2 summarizes the outcome of the comparison of the reviewed systems. Generally, the comparison is carried out in three categories, namely Design category, Enhancement category, and User Behavior category.

The Design category is to compare the implementation techniques of the reviewed systems. Several attributes such as Platform, Interface, Architecture, Notification, Voice Indication, and Graphical representation are included in this category. Secondly, the Enhancement category is to compare the scalability of the reviewed system. The attributes involved are Hardware Design Requirement, Flexibility in support additional appliance and action; Lastly, User Behavior category is to compare the readiness of user interface. Three attributes namely Computer Literacy Requirement, Physical Interaction, and Disability Support are been compared.

Table 2.2: Comparison of Technology Used in User Interface

Attribute		Smart Home Management System	Smart Home Microcontroller: Telephone Interfacing	Smart Home Control via PDA
Design	Platform	Instant Messaging	Telephone	PDA
	Interface	Web-Based	Dialing	Touch Screen
	Architecture	Jini	Dual Tone Multiple Frequency	CORBA Server-Client
	Notification	Status Window	Short Messaging Service	Status window
	Voice Indication	No	Yes	No
	Graphical Representation	Yes	No	Yes
Enhancement	Hardware Design Requirement	No	Yes. Electronic circuit design required.	No
	Flexibility in support additional appliance	Easy	Difficult, since hardware changes is required.	Easy
	Flexibility in support additional action	Easy	Difficult, since hardware changes is required.	Easy
User Behavior	Computer Literacy Requirement	Yes	No	Yes, but minimum only.
	Physical Interaction	Yes	Yes	Yes
	Disability Support	Not for vision impaired, handicapped	Not for vision impaired, handicapped	Not for vision impaired, handicapped

Generally, SHMS and Smart Home Control via PDA provide graphical user interface and notification to user either via computer systems or PDA. However, since the later is using touch screen technology for accessing user interface, it provides better look and feel to users (Koskela, 2004). In addition, both of the systems are easier to be enhanced to support additional command and appliance compared to the Telephony Interfacing system as there is no hardware design is required.

As in Table 2.2, the interaction with the user interface of all the reviewed systems required vision focusing and certain degree of movement. For example, a user needs to use mouse to point and click at the icon in web based interface of Smart Home Management system. As discussed, the focusing and clicking on an icon could be a challenge to elderly whom are generally having vision and mobility impairment It is arguable that we could size up the icon in graphical user interface to overcome the vision limitation of elderly. However, the bigger an icon is, the bigger of screen is required or the lesser of number of icon could be viewed in a single screen. The bigger the screen is, the lesser of the portability of a device. By the way, the lesser of number of icon could be viewed in a single screen means that the elderly needs extra effort to navigate between different screens.

Besides that, the table 2.2 also revealed that a user of reviewed systems particularly Smart Home Management system and Smart Home Control via PDA should posses certain degree of computer knowledge. Normaliza et al. (2008) pointed out that elderly in Malaysia felt insecure and decreased in level of confidence towards information and communication technology (ICT), which include computer usage. Additionally, training on the user interface could be another challenge due to the elderly often has difficulties interacting through unfamiliar means, such as keyboards and computer screens (Roy et al., 2000).

Therefore, our proposed system shall provide speech user interface in addition to the existing graphical user interface to provide Smart Home facilities to this minority group. Besides that, notification through voice or speech which is an important element of Smart Home for the minority group (Mouller, 2005) will be implemented in the proposed system

2.3 Application of Speech User Interface System

Speech is a natural form of communication that is pervasive, efficient, and can be used at a distance (Grasso, 1998). A speech user interface or also known as spoken dialogue interface is the script to a conversation between an automated system and a user. This script contains all the utterances that the automated system will speak to the user and the logic to decide which utterances to speak in response to user input (Hura, 2008). Underlying the speech user interface is speech recognition that has the ability to capture and decode the user's spoken input. Aside to the to the speech recognition technology, speech synthesis is important as the output technology in the speech user interface.

Generally, the input and output of the speech recognition and speech synthesis respectively are in natural language form and the language plays an important part in the communication between human and the speech user interface (Fraser, 1991). To facilitate research, linguists have defined at least seven levels of language analysis which spoken natural language and text can be extracted. These levels are (Allen, 1995):

- phonetic level – study of the speech sounds.
- morphological level – study of the structure and form of words from basic meaning units called morpheme.
- syntactic level – study of the rules whereby words or other elements of sentence structure are combined to form grammatical sentence.
- semantic level – study of the dictionary meaning of words, phases, and sentences, and also for the meaning they derive from the context of the sentences.
- pragmatic or practical level – study of the ways that the setting of the sentence in a discourse is used to determine its correct interpretation and its effects on the listener.
- discourse level – concern the effects of preceding sentences to the following sentence.
- world level – concerns knowledge of the physical world, the world of human social interaction, and the role of goals and intentions in communication.

With the recent progress in Speech Recognition, a number of speech based techniques have been explored and being implemented in the computer systems (Gilbert, 2003). For examples, NOVeLLA (Hodas, 2001), an electronic book (e-book) reader which delivering a powerful speech based user interface; Voice Access Booking System (Zajicek, 2004), which enables clients of Age Concern Oxfordshire to organize their IT taster session by booking, canceling or rearranging them using a speech dialogue via telephone; CHIL projects (Yang, 1999), which employs speech technology to automatically take minutes of the meeting, detect meeting progress and roles of the meeting participants.

2.3.1 Reviewing of Speech Based user Interface Systems

This section reviews three existing systems, namely, Spoken Dialogue Infrastructure for Location based Services, RNIB's Online Shop (IMAGINE), and Natural Language Interface to an Information Visualization Environment.

The systems are being selected as review systems since the speech user interface of the systems are capable to process English language input, which is the language requirement for our proposed system. Besides that, the systems employ the state-of-the art speech related software which could be employed in our proposed systems.

2.3.1.1 Spoken Dialogue Infrastructure for Location Based Services (SPIN)

SPIN (Nepper, 2007) is designed to integrate spoken dialogue interface to location based services. The system allows the location based services' user to use her voice to select a corresponding service for execution. As shown in the Figure 2.6, the SPIN consists of three layers, namely, the dialogue layer, the portal layer, and the service layer.

i. *The Dialogue Layer*

This layer is acts as the user interface of the system. It includes Automatic Speech Recognizer (ASR) which used to receive input from the user, Text-to-Spech (TTS) which is for synthesis output, a multi-modal browser for Javascript implementation, and SPIN specific internal modules such as DiagloueHandler, AsyncRequest, SyncRequest that are related to generate synchronous or asynchronous requests from the ASR results. These SPIN specific internal modules are made available to other location based services system via the SPIN API.

ii. *The Portal Layer*

As shown in the Figure 2.6, this layer consists of SPIN internal modules, namely MultiModalManager, DialogueAgregator, GrammarAggregator, ProavtibeEvent, and User. Generally, all these internal value are responsible for integrating graphical user interface and spoken dialogue interface for providing multiple services to the user simultaneously. For instances, the layer is used for collecting information about services that are available to the user, registering the user with all these LBSs, merging the speech dialogue and GUI interface into a single multimodal interface.

iii. *The Service Layer*

Generally, the service layer is used by Location Based Services provider to implement an adapter that connects an existing system to SPIN. In order to provide such functionality, this layer introduced the modality related functionalities to Location Based Services' system. For examples, proactive speech dialogue initiation, grammar handling, dialogue handling and forwarding a speech/GUI request to the respective Location Based Services' system.

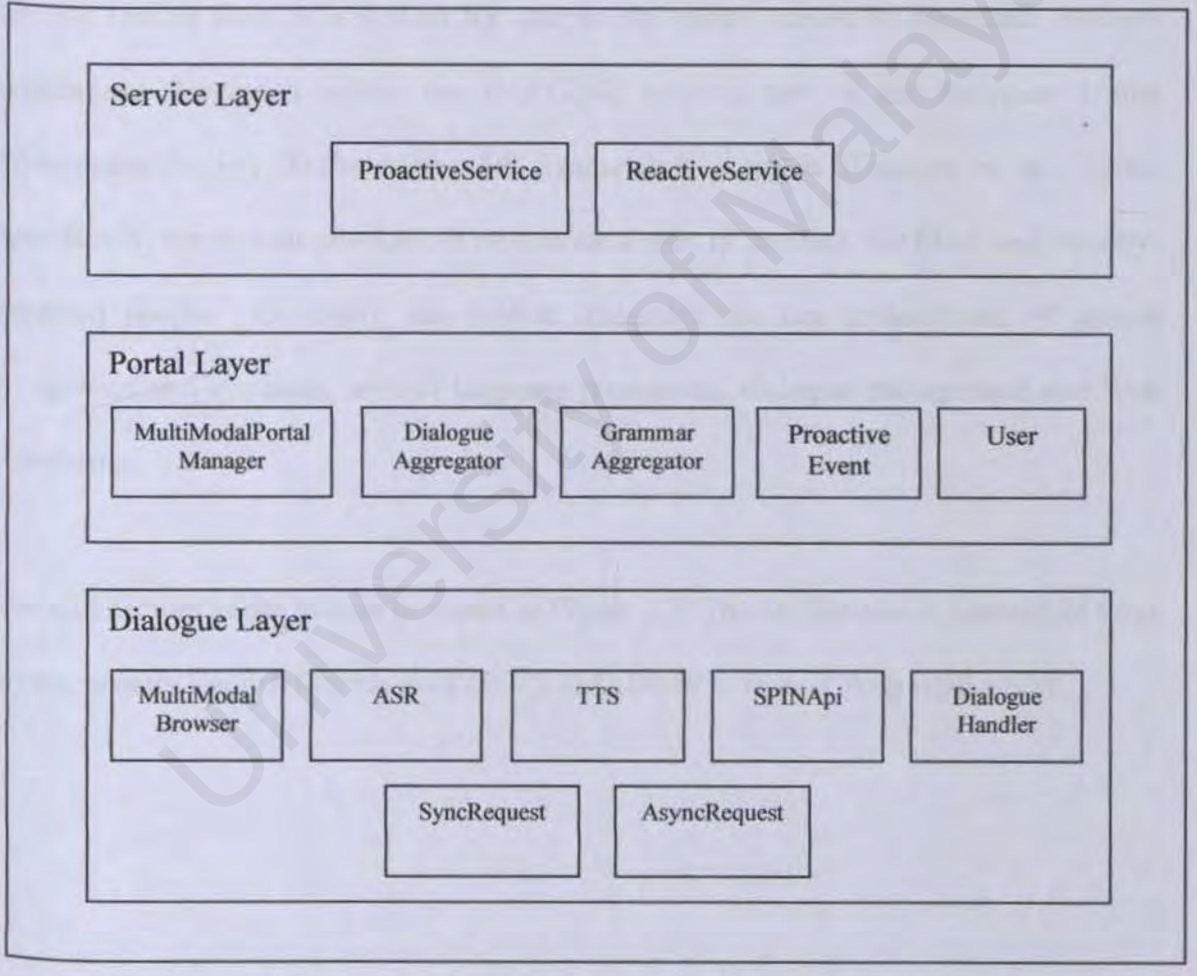


Figure 2.6: The three layers of the SPIN architecture (Nepper, 2007)

The three layers architecture of SPIN as shown in the Figure 2.6 has provided the flexibility of independent development of Location Based Services System and its multimodal interface and portal. In other words, the Location Based Services provider can develop a spoken dialogue interface for existing Location Based Services and deploy it without using a multi-modal portal.

2.3.1.2 RNIB's Online Shop (IMAGINE)

RNIB's Online Shop is a system for interactive speech access to electronic business applications developed within the IMAGINE project, part of the European Union Information Society Technologies 5th Framework program (Damper et al., 2006). Specifically, the system provides an on-line catalogue of products for blind and visually-impaired people. Generally, the system integrates the key technologies of speech recognition and synthesis, natural language processing, dialogue management and Web interfacing.

The architecture of the system is shown in Figure 2.7. The architecture is consists of three layers, namely English Speech Tool (EST), IMAGINE core, and Web application.

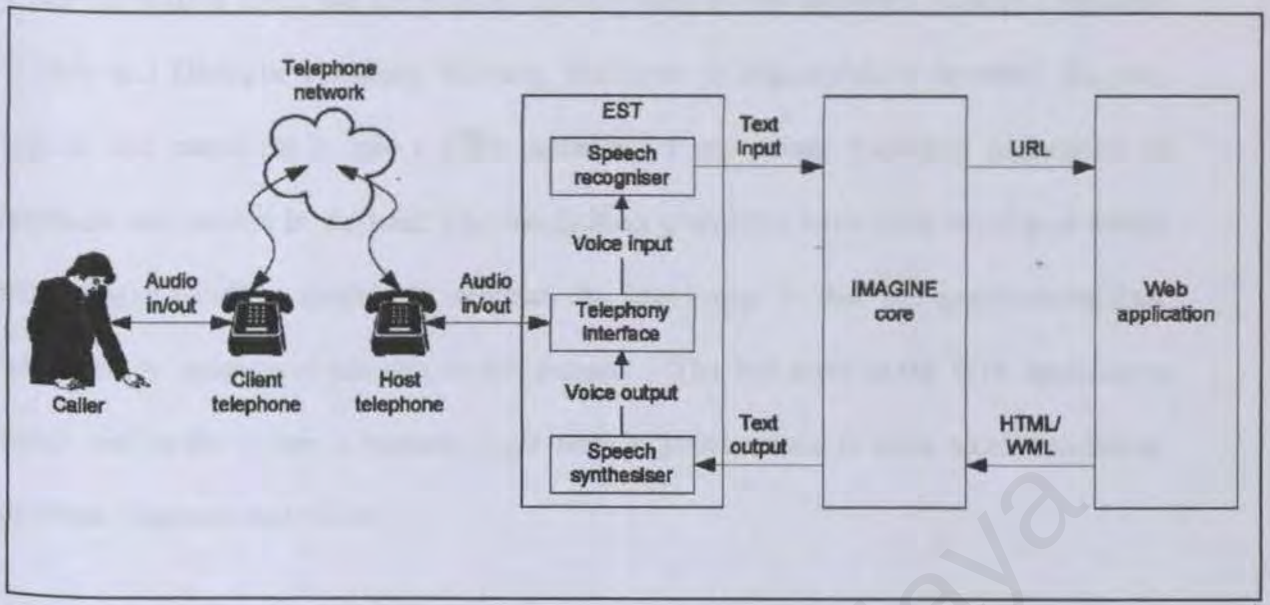


Figure 2.7: The architecture of RNIB's Online Shop (Damper et al., 2006)

In the EST layer, the telephony interface is responsible for connecting IMAGINE to a telephone line and the audio input is then forwarded the speech recognizer. On the other hand, the text output from the other two layers will be synthesis to the user via speech synthesizer. Nuance, the off-the-shelf commercial product, is employ as the engine for the recognizer and synthesizer. Generally, the EST will analyze the transcription provided by the recognizer and examines the confidence value associated with it. If it is below a threshold, the user is prompted to repeat or rephrase their last utterance. In addition, the EST also determines if prompts or other speech outputs are appropriate to the user input and, if so, the necessary actions are carried out internally. A purpose-written prompt definition language allows randomly-selected, alternative prompts to add variety to the interaction.

While the second layer, the IMAGINE Core consists of two modules, namely Linguistic Module and Dialogue Handling Module. The layer is responsible to interpret the user request and transform it into a form suitable for execution, including generation of feedback and prompt to the user. The recognition grammars have been developed within the dialogue handling module to structure the interaction so that the speech recognizer 'understands' utterances relevant to the domain. The last layer is the Web application which resides the system's business logic such as procurement to item, stock validation, payment clearance and others.

2.3.1.3 Natural Language Interface to an Information Visualization Environment System

The system is equipped with a speech based user interface to an information visualization (infoVis) system (Cox et al., 2000). The implementation of new speech based user interface enabled the user accesses infoVis via natural language and thus, the burden of translating intuitively questions into complex queries and view selections required by the existing infoVis system will be removed. In addition, the author of the system also suggested that user experience to the system will be improve since the system is fully multi-modal, which allowing users to move among different input modes according to their preference. For instance, the input mode as in a traditional point-and-click or as in natural language is allowed. Figure 2.8 shows the architecture of the system.

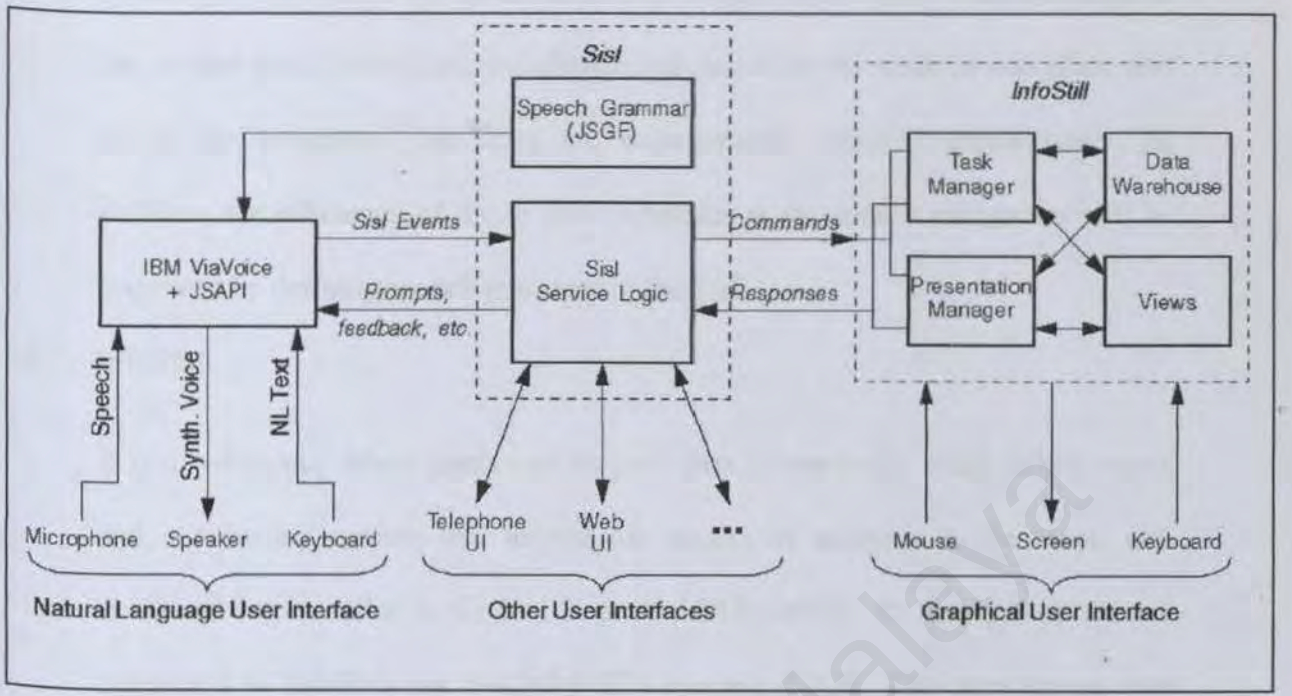


Figure 2.8: The Architecture of Natural Language Interface to an Information Visualization Environment System (Cox et al., 2000)

Based on the Figure 2.8, the system is built up with three main components, namely, IBM ViaVoice, Sisl, and infoStill.

i. IBM ViaVoice

IBM ViaVoice is employed as the speech recognizer and synthesizer engine of the system. User input will be processed and transformed to Sisl event for further processing. On the other hand, the feedback message will be synthesis to user.

ii. Sisl

Sisl, is an architecture and domain-specific language for designing and implementing interactive services with multiple user interfaces (Ball et al., 2000). By using Sisl, the multiple interfaces, e.g speech and GUI that employed in the system are provided to the same centralized service logic. Thus, when a change in the service logic is required, the change only needs to be made in one place and all of the associated interfaces are automatically updated appropriately. In addition, the efficiency of the system particular at the speech recognizer will be improved by defining speech grammar in the Sisl.

iii. InfoStill

It is a workspace where users can analyze data interactively using linked views and a reporting system that allows the results of analyses to be saved and examined later similar to Crystal Reports (SAP, 2008). At the system, Sisl is connected to InfoStill via the InfoStill's command API. This also means that when InfoStill is run with Sisl, users can interact with InfoStill through both the GUI included with InfoStill and the alternative interaction modes provided by Sisl.

Table 2.3 shows the comparison of characteristics of reviewed speech based user interface systems. Generally, the comparison is carried out at seven characteristics, namely Domain, Architecture, Application Model, Speech Recognition, Speech Synthesis, Language Processing Engine, and User Interface.

Table 2.3: Comparison of Characteristics of Reviewed Speech Based System

Characteristics	SPIN	IMAGINE	Natural Language to infoVis
Domain	Location Based Service	E-commerce / Online Shopping	Data Mining
Architecture	Layered	Web based	Layered
Application Model	Standalone	Server Client	Standalone
Speech Recognition	Yes	Yes	Yes
Speech Synthesis	Yes	Yes	Yes
Language Processing Engine / Parser	Automatic Speech Recognizer	Nuance	ViaVoice
User Interface	Graphical User Interface	Telephony	Graphical User Interface

2.4 Review of Implementation Tool

In this section, the review of required software tools for the implementation of the proposed system is carried out. The review begins with LinkGrammar, which is a natural language parser, followed by WordNet, the lexical reference or so-called 'dictionary' to the system (Miller et al., 1995). Java3D, a Java library for graphic or animation generation is the third item to be reviewed. And lastly, the review on Java Speech API 1.0 is carried out.

2.4.1 LinkGrammar

LinkGrammar is used in the back end as the language parser of the proposed system. It is a parser used to analyze English sentences which is based on dependency theories and was developed at the Carnegie Mellon University (Sleator and Temperley, 1991). LinkGrammar is written in C and it is available for both UNIX and Windows platforms. LinkGrammar comes with an API, which gives access to vital functions of LinkGrammar. The API grants access to the dictionary as well as the link structure.

The LinkGrammar has option to alter the way it processes a text. It contains over 60,000 words and it parses one sentence at a time. LinkGrammar can handle typographical errors. The result from a parsed sentence is the dependency structure: links between words, which reflects the grammatical relations between them. The output from LinkGrammar can be shown in two ways. The first way shows the dependency links that connect two words.

There are five different kinds of files used to parse sentences: dictionary files, post-processing files, constituent-knowledge files, affix files and word files. The dictionary files contain rules about links. The post-processing file is described above. The constituent file contains the rules to create a constituent tree. The affix file contains rules to handle punctuation and special characters. The word files contain categories of words.

LinkGrammar comes with standard files but they can be edited or replaced (Akeberg et al., 2003). LinkGrammar can also set a verbosity level to determine the amount of output.

Parsing of a sentence in LinkGrammar consists of several passes. First, it tries to find a linkage without null links. If it does not succeed, the parser will try to find a complete linkage with one null link and so on. When everything else has failed, LinkGrammar enters panic mode (that can be switched off) and tries again with different rules. It will only consider links of a certain length and will allow islands of disconnected words. In this mode, parsing will always be completed in a reasonable time. Notice that LinkGrammar is not one hundred percent reliable. It sometimes fails to produce a valid result.

To get a valid linkage in LinkGrammar, all words must be connected either directly or indirectly to each other and the links cannot cross. The words' part-of-speech is indicated by suffixes such as .n designating a noun, .v a verb, .a an adjective, and .e an adverb. Depending on the type of grammatical relation between the words, different labels annotate the link: subject, object, and others. The word that the S-link connects to the left is a subject (or the headword of the subject phrase). The word to the right is the finite verb. The part-participle link enables the location of the subject and the agent in a passive sentence.

2.4.2 WordNet

WordNet is a lexical reference that was developed at the Cognitive Science Laboratory at Princeton University under the direction of Professor Miller (Miller, 1995). Initially, it is designed for English but later the project extends to support other languages as well. WordNet separates words into nouns, verbs and adjectives. These are organized into synonym sets (synsets). Words in each part of a speech class have specific functions. These functions include synonyms, and hyponyms for the noun class.

The proposed system used the WordNet to find the synonym of the particular word. For instance the words "turn". In our domain, the sentence "turn on the light" is equal to "switch on the light".

2.4.3 Java3D

Java 3D is used to build the graphical representation of the proposed system. User instructions will be processed by back end processes such as the LinkGrammar module and the outcome is displayed as an animation form that is generated using Java3D. The API is a hierarchy of Java classes which serve as the interface to a sophisticated three dimensional graphics rendering and sound rendering system (Bouvier, 2000). It works with high-level constructs for creating and manipulating 3D geometric objects. These geometric objects reside in a virtual universe, which is then rendered. The API is designed with the flexibility to create precise virtual universes of a wide variety of sizes, from astronomical to subatomic (Bouvier, 2000).

Despite all this functionality, the API is still straightforward to use. The details of rendering are handled automatically. By taking advantage of Java threads, the Java 3D rendering is capable of rendering in parallel. Moreover, it can also automatically optimize for improved rendering performance.

A Java 3D program creates instances of Java 3D objects and places them into a scene graph data structure. The scene graph is an arrangement of 3D objects in a tree structure that completely specifies the content of a virtual universe, and how it is to be rendered.

2.4.4 Java Speech API 1.0

The Java Speech API, developed by Sun Microsystems in cooperation with speech technology companies, defines a software interface that takes advantage of speech technology. It defines a standard, easy-to-use, cross-platform software interface to state-of-the-art speech technology (Sun Microsystems, 1998). Two core speech technologies are supported through the Java Speech API: speech recognition and speech synthesis. Speech recognition provides computers with the ability to listen to spoken language and to determine what has been said. In other words, it processes audio input containing speech by converting it to text. Speech synthesis provides the reverse process of producing synthetic speech from a text.

In order to develop an interface that process speech based instruction, Java Speech API is used as the library in the development of corresponding programming interface to speech recognizer and synthesis. In addition, the Java Speech API is an extension to the Java platform which allows us to extend the functionality of Java platform that used to develop the proposed system.

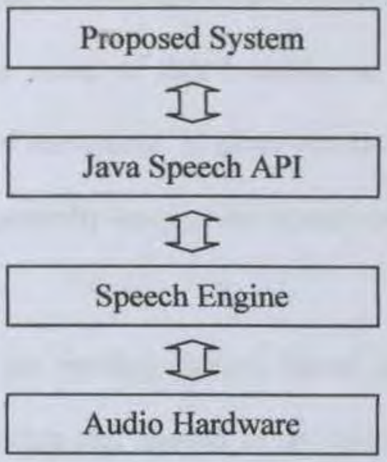


Figure 2.9: The Java Speech API Stack (Sun Microsystems, 1998)

Figure 2.9 shows the stack of the Java Speech API. At the bottom of the stack, it is the audio hardware that resides in our computer. The speech engine that sits on top of the stack interacts with the audio hardware. Meanwhile, the Java Speech API that sits on top of it provides a standard and consistent way to access the speech synthesis and speech recognition functionality provided by the speech engine

2.4.5 Speech Engine (Microsoft Speech SDK)

As displayed in Figure 2.9, speech engine is needed by the Java Speech API. Generally, the engine is used to interact with the audio hardware of machine. Microsoft Speech SDK which bundled with Microsoft advanced speech recognition engine and Microsoft concatenated speech synthesis engine is chosen as the speech engine. It is a free speech engine and downloadable from Microsoft website.

2.5 Conclusion

Reviewing on the user interface of existing system revealed that the reviewed systems are not elderly friendly since the using of user interface of such systems required vision focusing and certain degree of movement. In other words, such user interfaces are not so suitable for elderly who are generally having vision and mobility impairment.

Additionally, reviewing on the existing speech based application had enabled us to identify the software components that needed to be developed in our proposed system. Besides that, we have also successfully identified the necessary development tools namely, LinkGrammar parser, WordNet dictionary, Java3D API, Java Speech API, and Microsoft Speech Engine.

Chapter 3.0 Methodology

3.1 Introduction

A well-defined methodology is vital to the system development since the technique of methodology will affect the end result of the system. The main focus of this chapter is to outline the methodology undertaken in the development of the proposed NLIDS.

3.2 Research Methodology

As shown in Figure 3.1, the research methodology used in this project involves of three phases. In the first phase, literature review on the existing systems has been carried out. The purpose of the literature review is to review the current status and knowledge of existing systems. In addition, the approaches and state of the art techniques implemented in the existing system had also been reviewed. The outcome of the Literature Review phase is discussed in Chapter Two.

The second phase in research methodology is Tool development. A prototype system has been developed according to the methodology of software development described in the next section. The developed prototype is used as a tool to evaluate the acceptance as well as the feedback of the target audience. The detail explanation for building the prototype is discussed in the next few chapters.

The last phase is Evaluation. The objective of the research has been evaluated based on the result obtained in an evaluation session conducted for target audience. Besides that, the feedback on the prototype had also been gathered through a questionnaire. The evaluation's result is documented in the Chapter 7.

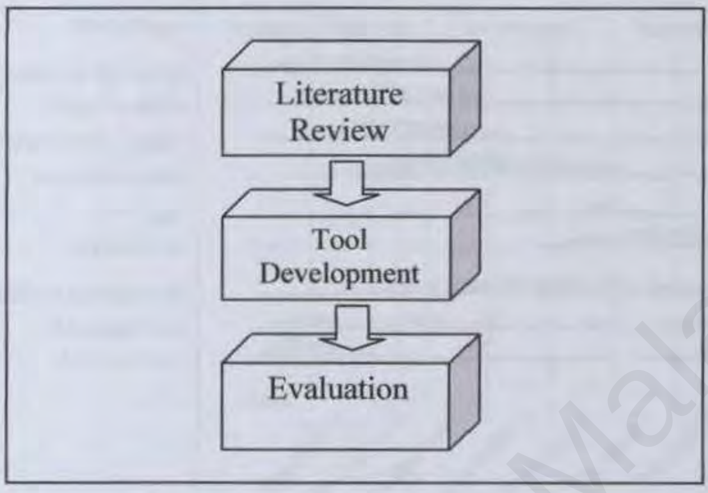


Figure 3.1: Research Methodology of NLIDS

3.3 Tool Development Methodology

NLIDS adopted Rational Unified Process as the tool development methodology. The methodology provides a disciplined approach to assigning tasks and responsibilities within a development organization (Kroll, 2005). Additionally, it is a guide for how to effectively use the Unified Modeling Language (UML). The UML is an industry-standard language that allows us to clearly communicate requirements, architectures and designs.

As shown in Figure 3.2, there are four phases in the Rational Unified Process, namely Inception Phase, Elaboration Phase, Construction Phase, and Transition Phase. In our research, five relevant workflows namely Requirements, Analysis and Design, Implementation, Test, and Development across each phase have been adopted in the development of NLIDS. The tasks performed in each workflow are discussed in the following section.

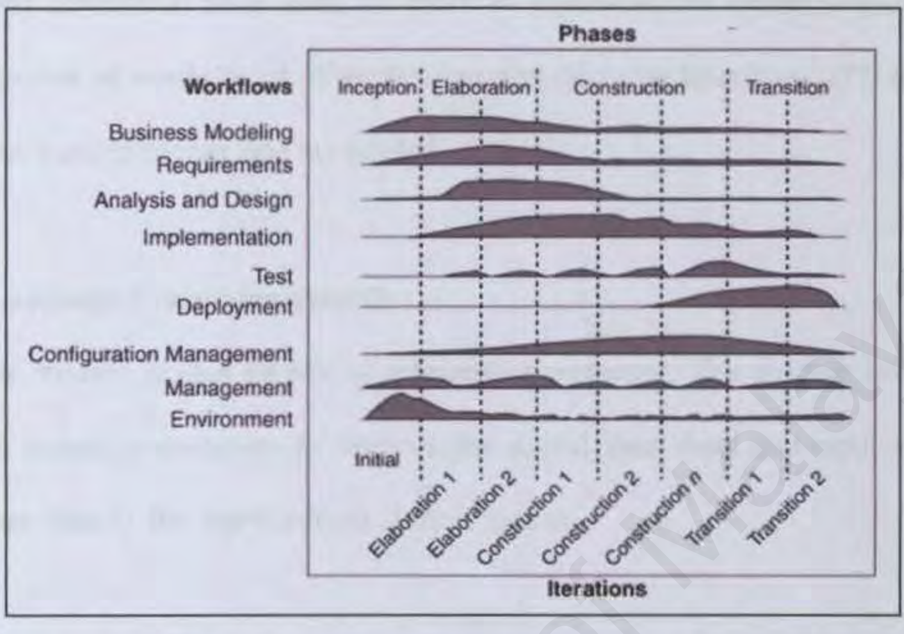


Figure 3.2: The Rational Unified Process Lifecycle Phases (Kroll, 2005)

3.3.1 Requirements

The first workflow carried out is requirement gathering. It is the basis of the development of proposed system since it provides us a clear view of what is needed to be achieved. There were two main objectives during requirement gathering, which were to understand the domain of the NLIDS and to identify the requirement of the proposed system.

In order to accomplish the first objective, reviewing on the user interface of existing system have been carried out. In general, as discussed in the Chapter Two, the processing flows involved in the reviewed systems are as following:

i. Input Receiving Module

In general, a stable speech recognizer engine will be employed to receive speech based instruction from user. In order to transform the speech instruction into a sequence of words, a set of application programming interface (API) together with speech and grammar rule are needed.

ii. Language Processing Module

This module is also known as information extractor. The module uses linguistic and semantic analyzers to retrieve the useful data from user input and converts these data to the standardized desired output..

Other than syntactic parser, the module also requires a lexical database to reference the domain meaning for the data that is parsed by syntactic parser. As reviewed in Chapter Two, WordNet is a lexical reference that is widely used in natural language research. It is designed in English and is able to separate words into nouns, verbs and adjectives.

iii. Visualization Module

This module serves as the output or response module to user. Generally, it is responsible to render the virtual environment operation according to the processed input from language processing module.

In order to accomplish second objective, the user requirement has been identified. As stated in the earlier section, the proposed system is intended to develop a speech based user interface in Smart Home system for elderly to overcome the difficulty in accessing such system due to mobility and visually impaired. Therefore, the proposed system must at least to be implemented in the following manner:

- English is defined as "natural language" and input to the system.
- The system must perform according to the allowed input.
- The output should be represented in graphical or animation form.

As shown in Figure 3.2, the requirements have also been redefined during the later phases such as Elaboration and Construction phase which the implementation work has already been started.

3.3.2 Analysis and Design

After gather the initial requirement, we have started the Analysis and Design workflow. During this workflow, we have performed the analysis on the requirement gathered and produced the architecture of the proposed system that illustrates the major modules.

As shown in the Figure 3.3, the proposed system is divides into three modules: Input module, Language Processing module, and Visualization module. In general, Input module receives input from the user and formalizes it before sending it to the language processing module. It includes the design for system behaviours such as data structure for temporal planner and techniques to model the parsed input.

The second module, Language Processing module, is the "brain" for the proposed system. At the earlier stage, the input data from Input module would be parsed into nouns, verbs and adjectives. The parsed output would then be processed with domain specify logic in the same module at later stage.

The last module in NLIDS is the visualization module that has been implemented using Java3D API. The output of an instruction would be rendered in animation form.

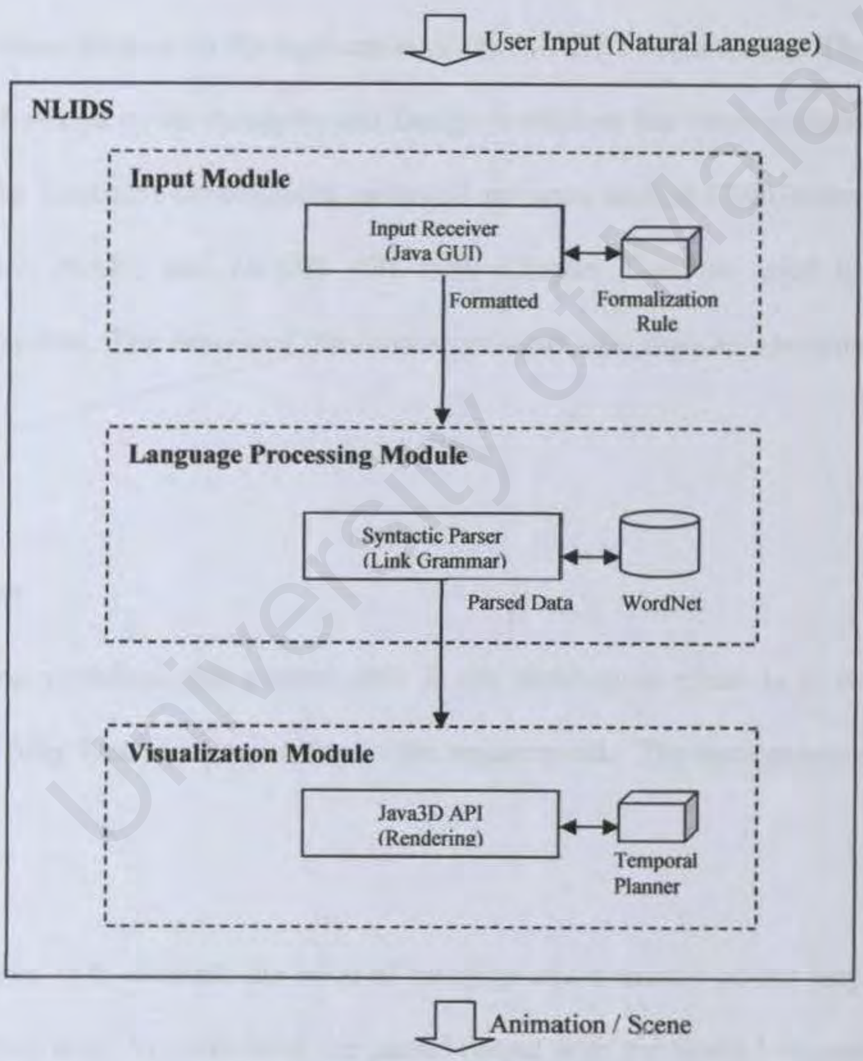


Figure 3.3: Architectural Diagram of NLIDS

In the workflow, several artifacts have been produced. For instances, Use Case diagram that used to present a graphical overview of the functionality, Sequence diagram that used to show the interaction between software components, and Class diagram that captured the design of proposed system's class. The detail of the tasks carried out in the Analysis and Design workflow are discussed in the Chapter Four and Chapter Five.

3.3.3 Implementation

This workflow focuses on the realization of the NLIDS's requirement. The architectural diagram produced in the Analysis and Design workflow has been realized into the real system. The identified development tools and software such as LinkGrammar, WordNet, J2SDK1.4.2, JSAPI, and Java3D API from Chapter Two are used to develop the proposed system. The details of the Implementation workflow are described in Chapter Six.

3.3.4 Test

The Testing workflow that started early in the elaboration phase is to ensure that the NLIDS is fully functional according to the requirement. The testing was carried out in two steps.

The first step is to evaluate the level of integrity and accuracy of the proposed system. This has been done by comparing the parsed output with the desired outcome based on a set of predefined instruction. While at the second step, a Questionnaire form has been passed to the user. This step is to gather the user feedback on the proposed system. The details of the Testing workflow are addressed in Chapter Seven.

3.3.5 Deployment

Defects and errors that are not been found during testing workflow may be discovered during the Deployment workflow. Besides, tailoring on the existing implementation is possible during the workflow also. Since the proposed system is using the popular platform and open source programming language, the maintenance of the system is possible, easy and transparent. The future enhancement possibilities are discussed in the Chapter Eight.

3.4 Conclusion

The LinkGrammar, WordNet, J2SDK 1.4.2, JSAPI, and Java3D API are among the implementation tools that are been used in developing the NLIDS. The tool development methodology adopted is the Rational Unified Process which consists of four phases: Inception Phase, Elaboration Phase, Construction Phase, and Transition Phase. This chapter briefly introduced the tasks that been carried out in each workflows.

Chapter 4.0 System Analysis

4.1 Introduction

"System analysis enables transforming a problem definition from a fuzzy set of facts and myths into a coherent statement of a system's requirements." (Bahrami, 1999) Therefore, the main purpose of the analysis is to examine the necessary requirement as well as the functionality for NLIDS.

4.2 Research Findings

In order to understand the domain of the NLIDS, related study was carried out on the two key specifications which are the LinkGrammar parser and the SmartHome system. The results of the study are summarized below.

i. Parser – LinkGrammar Analysis

Parsing is the principal stage in many natural language processing (NLP) systems. A good parser is expected to return an accurate syntactic structure of a sentence. This structure is typically forwarded to other modules so that they can work with unambiguous and well-defined structures representing the sentences. It is to be expected that the performance of an NLP system is degraded if the parsing system returns incorrect syntactic structures.

LinkGrammar (Sleator and Temperley, 1991) is employed as the natural language parser in the proposed system. Generally, LinkGrammar is a syntactic parser of English, based on an original theory of English syntax. Given a sentence, the system assigns to it a syntactic structure, which consists of a set of labeled links connecting pairs of words. The parser also produces a constituent representation of a sentence which shows noun phrases, and verb phrases. Sleator and Temperley (1991) came up with an algorithm used for parsing with LinkGrammar that is commonly used. In general, the algorithm proceeds by constructing links in a top-down fashion.

LinkGrammar is an elegant context free parser and has a unique combination of useful properties (Moll´a et al., 2003) as below:

Direction of dependency: LinkGrammar's 'links', although similar to true dependencies, do not state which participant is the head and which is the dependent. However, LinkGrammar uses different link types for head-right links and head-left links, so this information can be recovered.

Clausal heads: LinkGrammar generally chooses the front-most element to be the head of a clause, rather than the main verb.

Graph structures: LinkGrammar's links combine dependencies at the surface-syntactic and deep-syntactic levels. The resulting structures are graphs rather than trees.

Dependency types: LinkGrammar uses a set of about 90 link types and many subtypes, which address very specific syntactic constructions.

ii. **Smart Home System Analysis**

As discussed in Chapter Two, most of the existing systems are using predefined instruction or also known as computer based command as an input to the system. For example, EyStar SmartHome Management System (SDA, 2008) which was developed by Secured Digital Applications works with a central processing unit that controls with a graphical iconic based interface to provide control of security systems, control of temperature and lights, remote monitoring and a digital telephone.

In addition to that, users of the system are only able to monitor security systems such as motion detectors, provided remote viewing capability and notify up to eight telephone numbers of an intrusion or abnormal temperatures due to a heating system failure via the same kind of user interface. As mentioned earlier, such computer based interface is an obstacle for elderly whom are mostly having vision or motor impairment in accessing the smart home system.

4.3 Analysis of Proposed System

Unified Modelling Language (UML) in the object-oriented analysis process (OOA) is used to represent the user requirements and to generate a model that represents the entire concept of the system. Throughout the OOA process, business model, use case model, sequence diagrams and object model will be used to illustrate and document the whole development process. In general, the requirement analysis for the NLIDS is divided into three different sections: user identification, business model, and use case model.

4.3.1 User Identification

In this project, as mentioned earlier, one of the objectives is to explore the possibilities of using speech user interface in smart home system. Therefore, the proposed system must allow user control on predefined electronic appliances such as fan, lights, TV and radio by using speech instructions. Additionally, English is the type of language used in the system. The user requirements defined are as below:

- **Instruction language**

  The targeted users would be elderly that are English literacy since the the instruction language used is in English only.

- **Feedback**

  The system would focus on providing immediate feedback to the user's interaction. A positive response will be given to the user if the system understands the instruction or vice versa where a negative response shall be given.

- **Functionality**

  The proposed systems will response in real time to the user instruction. For example, the fan or light in the virtual environment will turn on immediately if the user request to do so.

4.3.2 Business Model

Business model is used to layout the behavior or business requirement of a system. Figure 4.1 shows the business model of NLIDS. As shown in the figure, NLIDS is ready to receive speech instruction once it has been started. The speech instruction will be submitted automatically when the end of speech is detected.

The submitted instruction will then be parsed by the system parser and follows by validation against business rule and system's dictionary. If the instruction is invalid or unsupported, the system shall reject the instruction and notify the user regards the failure. In other cases, if the instruction is valid, the instruction will then be piped to the respective device module based on the targeted device, for example, the Fan Toggle module for fan. The respective device module is used to verify the current state of the represented device with the intended state. For instance, if the user requests to switch on a light which is currently on, the respective device module shall cease the instruction from being further process and notify the user regarding the outcome. Otherwise, the desire state of a device should be forwarded to the Operation Visualization module.

The Operation Visualization module will then generate corresponding animation according to the provided device's state. Again, we would like to stress that the NLIDS is just to demonstrate the possibility of implementing natural language processing in user interface of smart home system. Hence, the result of a valid instruction will only be presented in the computer animation form but no operation on any physical device is involved.

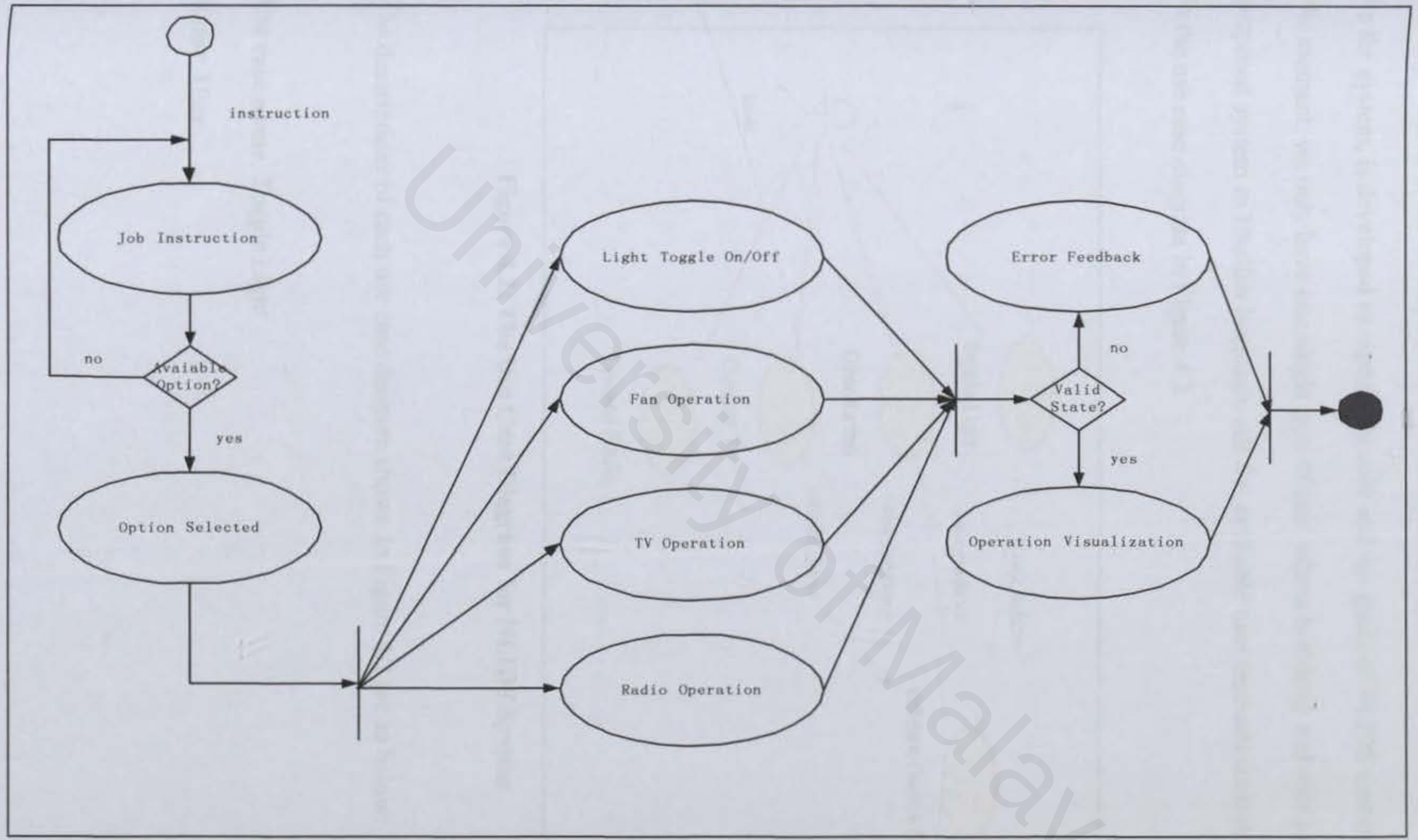


Figure 4.1: Business Model of NLIDS

4.3.3 Use Case Model

Use case which is the description of a potential business situation that may be supported by the system, is developed to capture the role and the goals of NLIDS system's users. At the moment, we only have one single type of user whom is elderly and able to instruct the proposed system in English language. All the available user instructions/tasks are shown in the use case diagram in Figure 4.2.

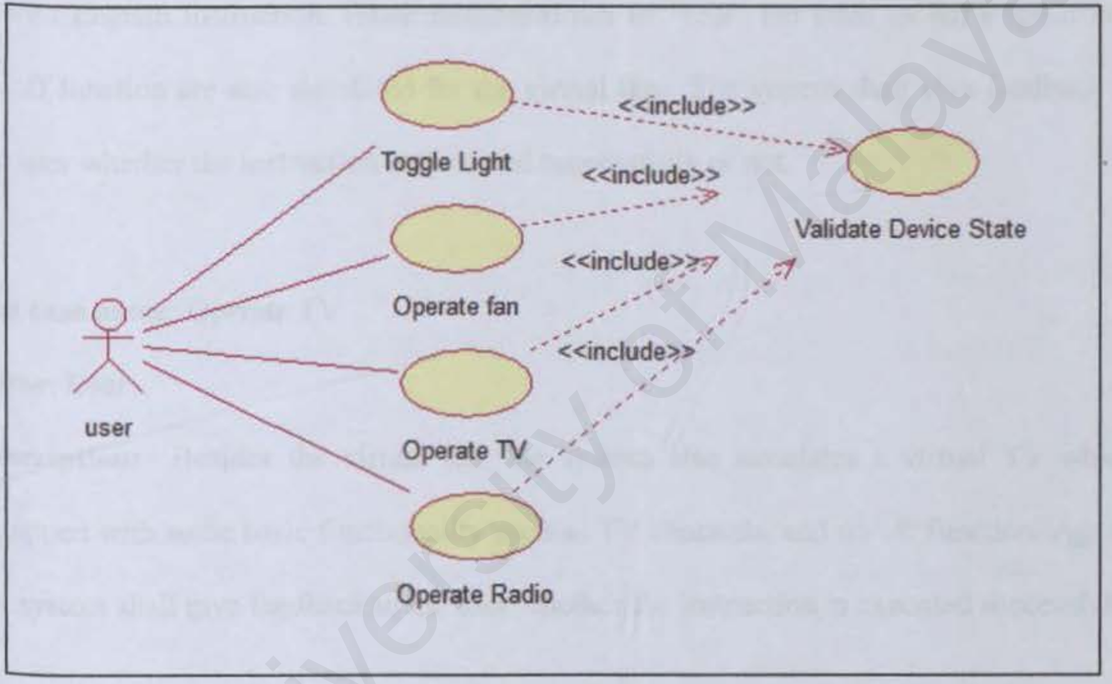


Figure 4.2: The Use Case Diagram for NLIDS System

The descriptions of each use case diagram shown in Figure 4.2 are as below:

Use case name: Toggle Light

Actor: User

Description: This use case is responsible for toggle on/off to the light that is the setup in the virtual environment. The user shall give the instruction in English language while the system shall give feedback to the user whether an instruction is executed successfully.

Use case name: Operate Fan

Actor: User

Description: There is a Fan simulated in the virtual environment where user can operate it with English instruction. Basic functionalities of "real" fan such as fan's speed and on/off function are also simulated for the virtual fan. The system shall give feedback to the user whether the instruction is executed successfully or not.

Use case name: Operate TV

Actor: User

Description: Besides the virtual fan, the system also simulates a virtual TV which equipped with some basic functionality such as TV channels, and on/off function. Again, the system shall give feedback to the user whether the instruction is executed successfully.

Use case name: Operate Radio

Actor: User

Description: User also will be able to operate the virtual Radio that setup in the virtual environment. There is some basic functionality of radio such as radio channel and on/off function that is provided to the user. The system shall also give feedback to the user whether the instruction is executed successfully.

Use case name: Validate Device State

Actor: Use Cases

Description: This use case provides the functionality for checking the current state of a particular device. For an instance, if the user toggles on the light, the *toggle light* use case will check whether the light is currently on or off by triggering this use case. The appropriate message to the user will be responded to user depending on the output of this use case.

4.3.4 Sequence Diagram

UML sequence diagrams are used to model the flow of logic in a visual manner in order to validate the logic as well as identifying the behaviours of NLIDS. It is used for both analysis and design purposes.

The sequence diagrams for each corresponding use case are enclosed in Appendix A of this report.

4.3.5 System Requirement

The system requirements ensure that the system is function at optimum level as well as fulfilled the user requirements. Additionally, it must be complied with what in order to make sure that the system operates properly. The system requirements for the NLIDS are stated below.

4.3.5.1 Software Requirement

- *LinkGrammar Parser (Version 4.1 or later)*

  It is a syntactic parser of English, based on LinkGrammar, an original theory of English syntax. Given a sentence, the system assigns to it a syntactic structure, which consists of a set of labeled links connecting to pairs of words.

- *LinkGrammar–WN (Version 1.0)*

  This is a lexicon expansion for the LinkGrammar Parser. It contains nearly 15,000 new word forms that are not available for the original LGP lexicon.

- *JSDK (Version 1.42 or later) and Java 3D API (Version 1.4.1 or later)*

  The Java compiler as well hierarchy of Java classes which serves as the interface to a sophisticated three dimensional graphics rendering and sound rendering system which is used to develop the system's core function and visualization module.

- Java Speech API 1.0

  This serves as the application programming interface for speech synthesis and recognition module. It is used to processes audio input containing speech by converting it to text. Besides that, the API is also used to synthesize response to user.

- Microsoft Speech SDK (Version 5.1 or later)

  This is the speech engine that interacts between the audio hardware of the machine and Java Speech API.

- *Windows*

Windows acts as the platform for the software mentioned earlier. Windows' various versions of 95/98/Me and NT/2000/XP could be used as the operating system.

4.3.5.2 Hardware Requirement

- A computer with minimum specification equipped with at least Pentium II, 128 MB memory and 400 MB free disk space, which is also the minimum requirement of JSDK.

4.4 Conclusion

In general, the system analysis which involves identifying the user requirements, system requirements and hardware requirements have enabled us to understand the domain of the NLIDS more accurately. Throughout the chapter, we have analyzed all the necessary aspects before proceed to the system design.

Chapter 5.0 System Design

5.1 Introduction

The design process involves developing several models of the system at different levels. Through the design process, any errors and omissions in the earlier stage can be discovered and rectified. This chapter will summarize the design of the proposed system into three sections. The first section shows the layout of the high level view of the system and the interfacing among each system's component while the following section will describe the system's functionality in detail using UML class diagram. Lastly, we will look into system interface design by designing a paper prototype.

5.2 UML Class Diagram

The functionality of NLIDS is organized into three layers according respective functionality in order to simplifying the designing process. Each layer represents in a respective class diagram namely as business layer class diagram, access layer class diagram, and view layer class diagram.

5.2.1 Business Layer Class Diagram

Business layer class diagram is used for modeling the business logic of objects and the interaction between classes that accomplish the NLIDS core business process functionality. Figure 5.1 represents the business layer class diagram of NLIDS.

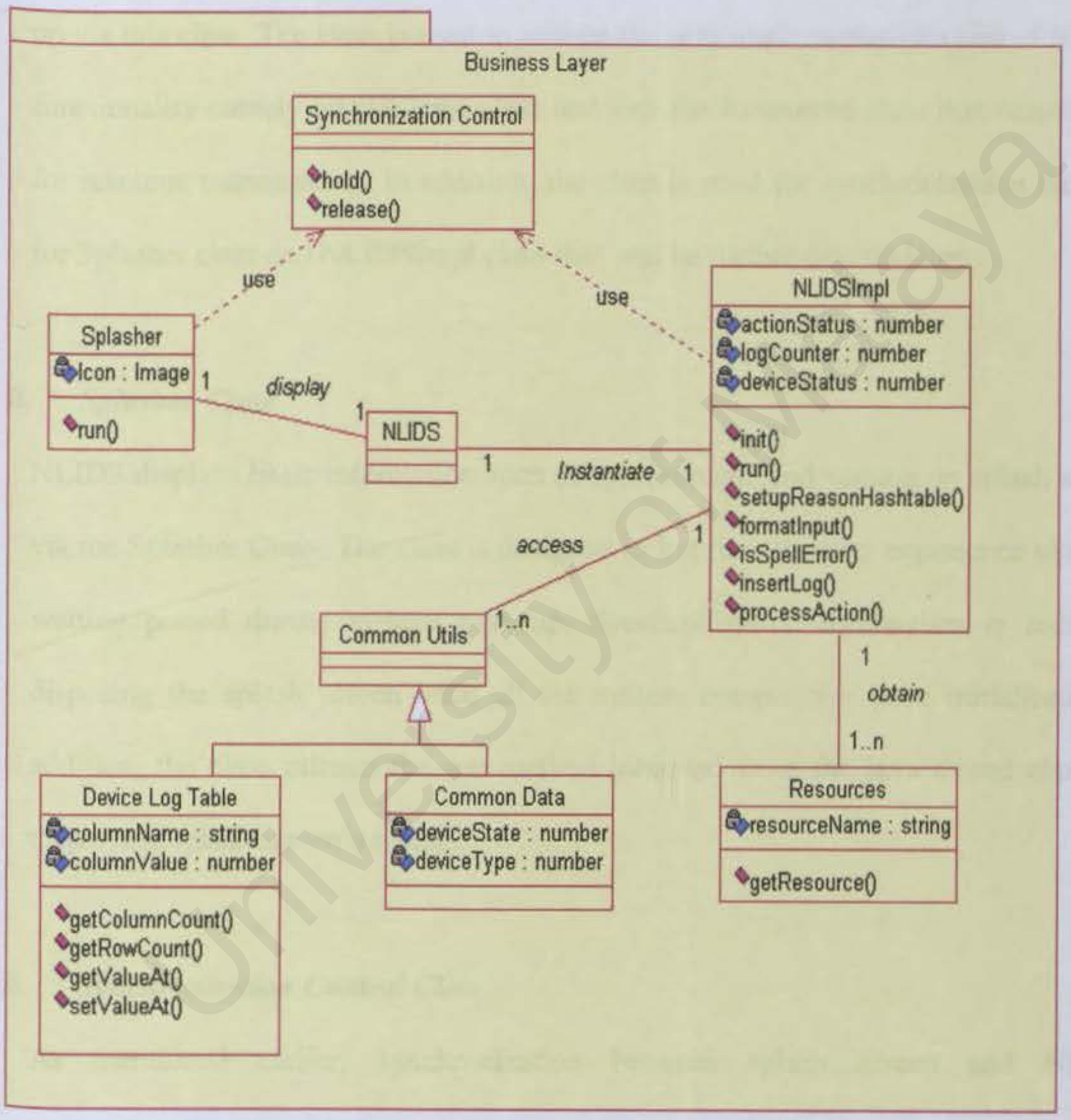


Figure 5.1: Business Layer Class Diagram of NLIDS

The figure shows the classes that created at the business layer to perform core processing functionality of NLIDS. These classes are:

i. ***NLIDS Class***

This is the main class of the entire system where the system can be launched and start up via this class. The class is used to initiate the core implementation class of NLIDS functionality namely NLIDSImpl class and link the Resources class that responsible for resource management. In addition, the class is used for synchronization medium for Splasher class and NLIDSImpl class that will be further discuss later.

ii. ***Splasher Class***

NLIDS displays basic information such as system name and version on splash screen via the Splasher Class. The class is designed to improve the user experience over the waiting period during system start up. Synchronization mechanism is used for disposing the splash screen once all the system components have initialized. In addition, the class utilizes the run method inherited from the java thread class for displaying splash screen's image.

iii. ***Synchronization Control Class***

As mentioned earlier, synchronization between splash screen and NLIDS Implementation class is needed to create a synchronized behavior for starting the implementation class once all the necessary objects are initialized and disposed the splash screen. The Synchronization Control Class is created to cater this purpose where Splasher class and NLIDSImpl class will be hold and release by using common mutual exclusive lock.

iv. NLIDSImpl Class

NLIDSImpl Class is the core implementation class for the entire system and there are various numbers of methods defined for this class including init, run, formatInput, processAction, and others. The functions are used to define the business logic as well as to control the system flow. Resources such as image and audio are obtained through the Resource class whereby the class utilized the common data structured provided by Common Utils Class to store information.

v. Resource Class

This is the helper class which provides the convenient for accessing the system resource such as image and audio. It is only used by NLIDSImpl class.

vi. Common Utils Class: Device Log Table and Common Data

The Common Utils Class consists of two subclasses namely Device Log Table class and Common Data class. The main purpose for the class is to provide a convenient data structure as well as storage utilities for proposed system. Several access functions such as setValue and getValue are created for this purpose.

5.2.2 Access Layer Class

This class consists of classes that act as wrapper classes and access classes to the existing packages. For example, LinkGrammar class is created to provide the accessibility to org.apache.commons.text.link package. Figure 5.2 shows the relationship between wrapper classes and their corresponding packages.

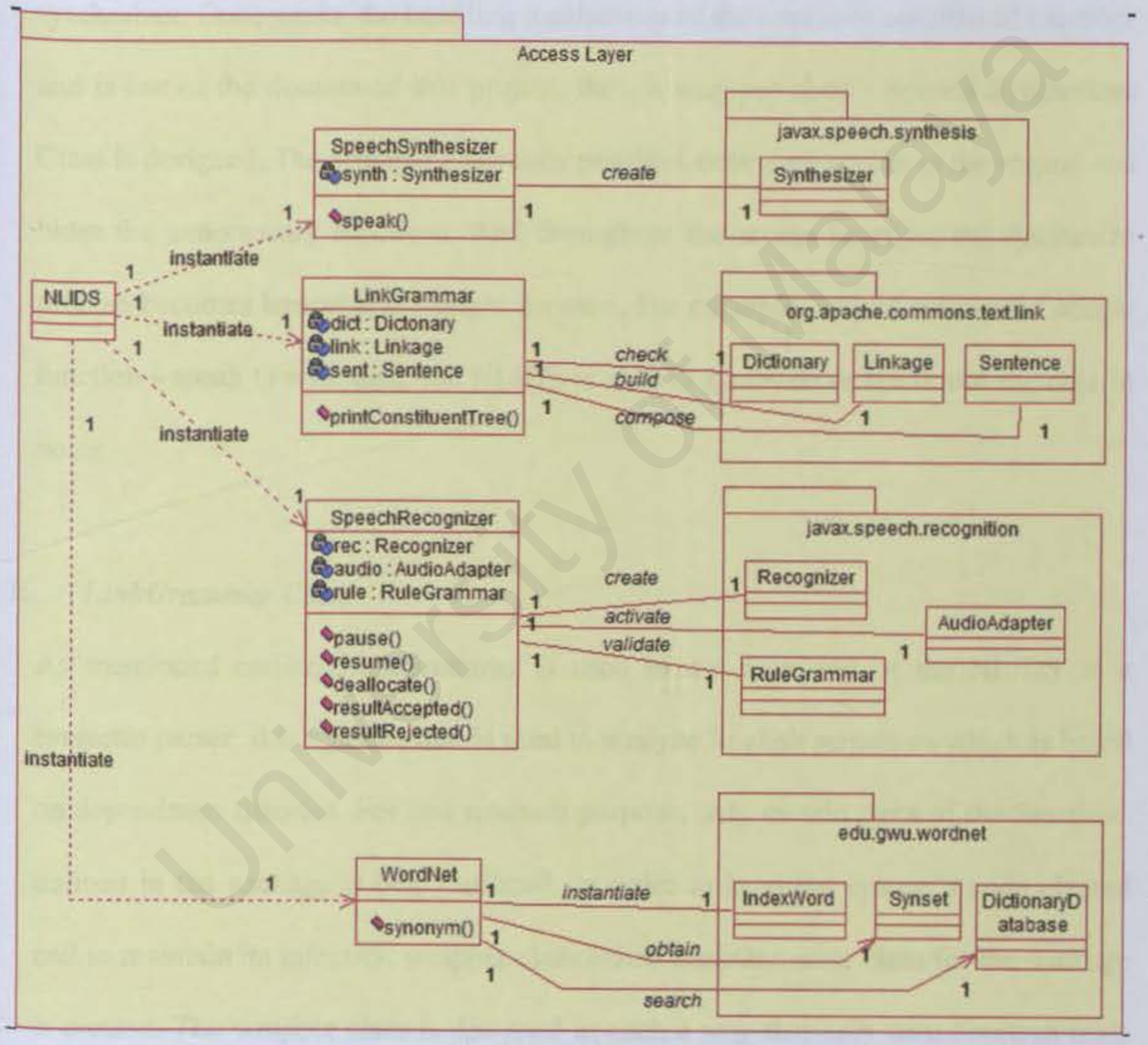


Figure 5.2: Access Layer Class Diagram of NLIDS

The descriptions of these classes are as follows:

i. Speech Synthesizer Class

A speech synthesizer engine is employed to make the NLIDS be more users friendly. The engine is expected to synthesize the respond to the user if the synthesize function is activated. An open source package - javax.speech.synthesize is chosen as the synthesizer. Once again, the handling mechanism of the engine is considered complex and is out of the domain of this project, thus, a wrapper class - Speech Synthesizer Class is designed. The wrapper class only provides necessary access to the engine and hides the unnecessary functions. And throughout the access function, the synthesize process becomes lenient and straight forward. For example, by just calling the access function - speak () with data, the NLIDS is able to synthesis or speak out the data in voice.

ii. LinkGrammar Class

As mentioned earlier, LinkGrammar is used in the back end of the NLIDS as a syntactic parser. It is a parser that is used to analyze English sentences which is based on dependency theories. For this research purpose, only certain parts of the functions defined in the package is being utilized. In order to keep the system's code cleared and to maintain its integrity, wrapper class called LinkGrammar class for the package is created. The wrapper class is designed in such a way that only core function from the package is provided with a compatible access function. Therefore, it will achieve the encapsulation for the rest of functions in the package.

iii. Speech Recognizer Class

In order to recognize the speech instruction from a user, a speech recognizer engine has been employed in the development of NDLIS. Basically, the engine functions as a recognizer of speech instruction and transforms the instruction to text form follows by forward the transformed instruction to the NLIDS parser. Similar to the speech synthesizer, a wrapper class has been designed to provide the accessibility to the functions of the recognizer engine. The wrapper class will be run as a thread so that NILDS is always ready to receive new speech instruction. Some important functions such as function to allocate systems resource (allocate ()) and function to check for validity of speech instruction (resultAccepted ()) are provided.

iv. WordNet Class

WordNet class is used to act as a lexical reference to identify the synonym of a particular word in NLIDS. Obviously, we need some levels of access to the class in order to make the class to be function as a lexical reference. A wrapper to the class is required to cater this requirement. The wrapper class is used to initiate the class during system start up, searching the synonym in the WordNet class's database, as well as obtaining the possible synonym set of a word. Again, with the employment of wrapper class, the real implementation of WordNet class is encapsulated.

v. javax.speech.synthesis Package

The package consists of speech synthesis functions which convert test based output into speech output which also known as text-to-speech (TTS) conversion package. This is one of the core technologies that defined in Java Speech API. When accessing the speech synthesizer wrapper class, NLIDS is capable to create synthesize or voice out a speech.

vi. ***org.apache.comon.text.link Package***

This package is defined by apache organization for providing functionality to one of the famous grammar linker – LinkGrammar. Its component classes are assessable via the LinkGrammar wrapper class. For example, the wrapper class is able to check the dictionary defined in the package, build the linkage between grammars by using functions in the linkage class, and compose a sentence with the Sentence class.

vii. ***javax.speech.recognition Package***

Apart from the synthesize package, this is another core technology supported in the Java Speech API. It is the package that defines functions for analyzing the spectrum (frequency) characteristics of the incoming audio, and comparing the sequence of likely phonemes against the words and patterns of words specified by the grammar rule. Once again, wrapper class implemented in NLIDS is functional as an assessor to the package. By creating a recognizer class, this follows by activating the audio device class and validating the input with grammar rule class. The speech instruction from a user will be converted to a text and forward to the parser of NLIDS.

viii. ***edu.gwu.wordnet Package***

The package is created by George Washington University for providing WordNet functionality in Java Language (Fellbaum, 1998). As summary, the package provides access to English language word database that defines relationships between words in a comparable method to the way humans think of related words. By calling the function in the WordNet wrapper class, NLIDS is able to assess the defined function in the package to determine the synonym of a word.

5.2.3 View layer Class

View layer class contains the visualization of the proposed system to user. Basically, it captures the design of proposed system interface.

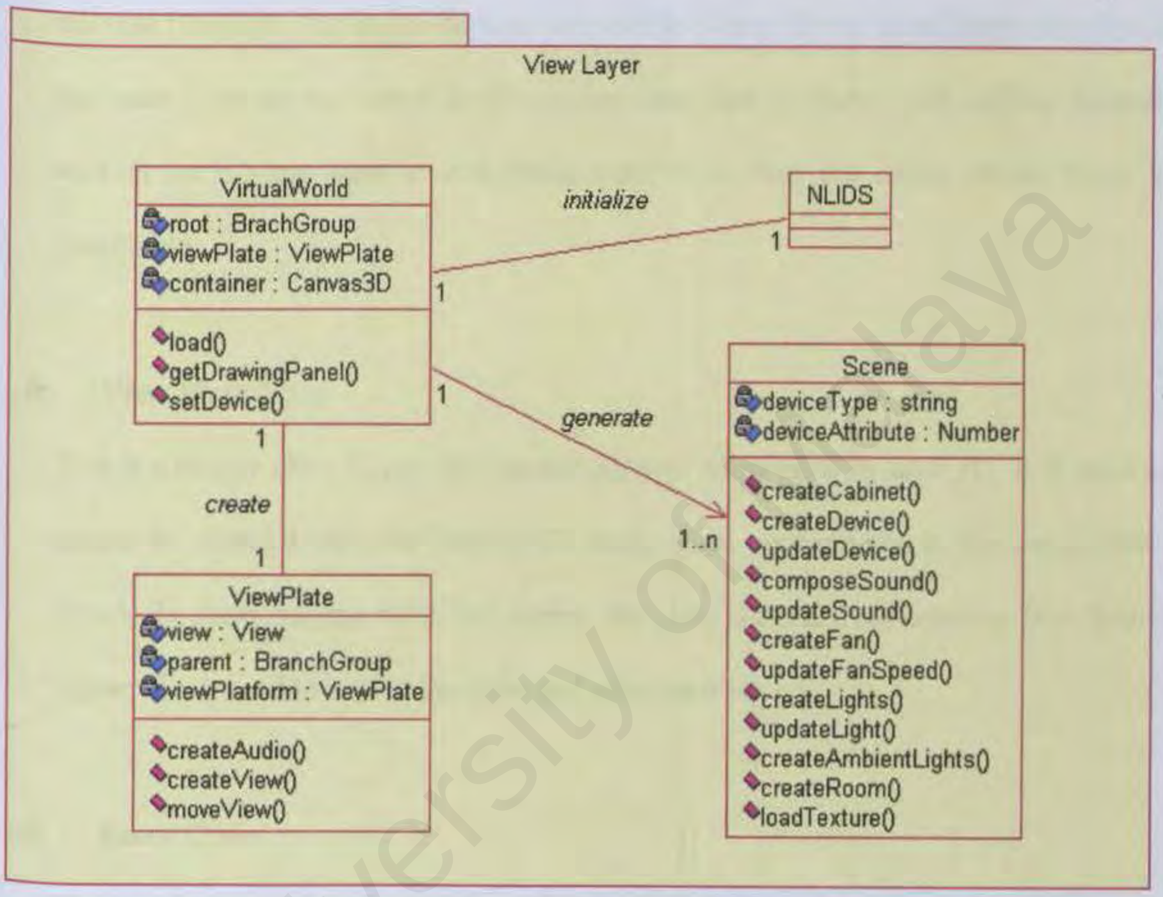


Figure 5.3: View Layer Class Diagram of NLIDS

Figure 5.3 shows the classes that were designed for the view layer. Basically we have created three classes: Virtual World class, View Plate class, and Scene class for rendering the virtual home in NLIDS. Each of the classes is describe as follow:

i. **Virtual World Class**

The class acts as the main class for visualization module which is called by NLIDS to generate the virtual scene. During system start-up or initialization, the class will create all the virtual devices such as light, fan, television, and radio by calling relevant functions that create devices provided in Scene Class. In addition, the class is also used to create the virtual home environment such as render wall, ceiling, floor as well as set a view plate to a suitable distance so that the entire virtual home is observable.

ii. **View Plate Class**

This is a simple class to provide functionality of view plate in Java 3D. It is used to ensure the visual devices in front of the image plate are rendered to the image plate. Beside the functionality described earlier, the class is needed for creating Java Sound Mixer which used to render the radio and television sound.

iii. **Scene Class**

This is the core functional class that is heavily based on Java3D API in the visualization module. The class is designed to provide function for rendering devices, function for generating environment such as wall and ceiling, function for providing real time rendering or animation such as changing fan speed and other necessary rendering related functionality. Besides the rendering functions mentioned, it is also uses Java Media API for composing sound for radio and television.

5.3 Paper Prototype of User Interface

Paper prototyping is a widely used method for designing, testing, and refining the user interface. It is started in the mid 1980s and then became popular in the mid 1990s when companies such as IBM, Honeywell, Microsoft, and others started using the technique in developing their products (Snyder, 2003). According to Bailey (2008), the benefits of using paper prototyping includes allowing rapid externalization of design ideas with low investment and iteration on a design many times prior to committing to the implementation.

These section discuses the layout of the user interface of the proposed system using paper prototyping. There is only two screens are designed for the proposed system since the proposed system is just a prototype system which is used to demonstrate the usage of speech as an input medium in Smart Home system. The two user screens are main screen that provides system information and feedback screen that prompting alert to user if any unexpected exception occurs.

i. Main Screen

The main screen of NLIDS provides all the necessary information as displayed in Figure 5.4 to increase user experience. The screen consists of the following Java components:

- Menu Bar – provides some basic system navigation capabilities such as reloading the initialization file.

- Visualization Screen – use for rendering supported virtual devices in the virtual environment.
- Input Field – receives user's input as well as displaying the voice input.
- Usage Log – shows the history of user input.
- Device Status Field – shows the current status of virtual devices.

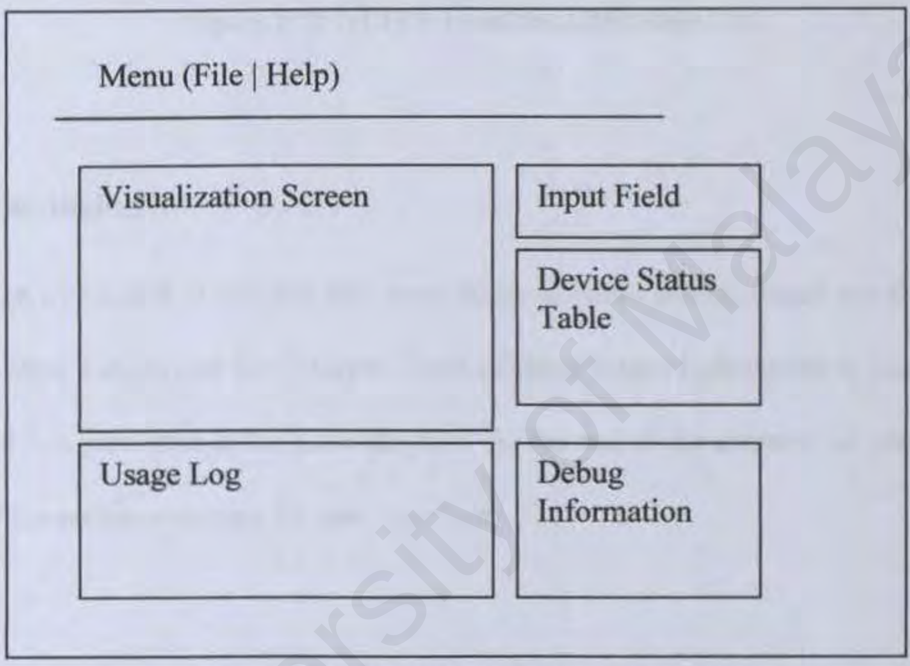


Figure 5.4: NLIDS Main Screen Layout

ii. Feedback Message Box

The paper prototype of the feedback message box that used to prompt the input violation and constraints messages to user is shows in Figure 5.5. There are two un-editable fields that show the user input and its corresponding error messages. As suggested by Dmitry (2009), different color code has been used for different type of message error.

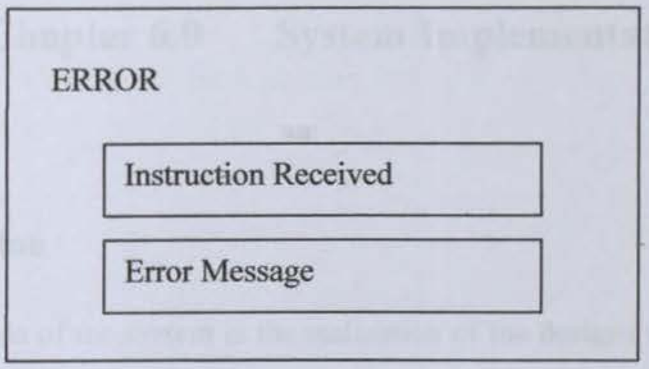


Figure 5.5: NLIDS Feedback Message Box

5.4 Conclusion

The design of NLIDS is divided into three main different layers, which are the Business Layer, Access Layer, and View Layer. Each of these layers is designed to cater different needs and is represented in the class diagram. At the end of the chapter, we also provide a layout of the paper prototype for user interface.

Chapter 6.0 System Implementation

6.1 Introduction

The implementation of the system is the realization of the designs that were proposed in Chapter Five. The NLIDS functionalities such as input parsing, parsed input formatting, and visualization generation are implemented using Java language with several libraries discussed in the earlier chapter. In addition, this chapter also analyzes on the system configuration and installation of the required software.

6.2 Implementation of System Module

As discussed in Chapter Five, the three main modules which are the Input & Response Module, Language Processing Module, and Visualization module are expected to handle for separate processing flow, correlate the input from other modules, and generate certain output to other module or user. In order to minimize the complexity during the development phase especially in the coding phase, we have divided the implementation of the NLIDS into several modules according to their specific functionality.

6.2.1 Implementation of Input & Response Module

The discussion of implementation of NLIDS's interface is divided into two sections, namely Input Module and Response Module as following:

6.2.1.1 Input Module

This module is responsible for recognize the instruction in speech form and format the instruction before relaying the instruction to Language Processing Module. It is a preliminary processor that is implemented for filtering invalid input from being further processed in the system. Figure 6.1 shows the activity diagram of the input module.

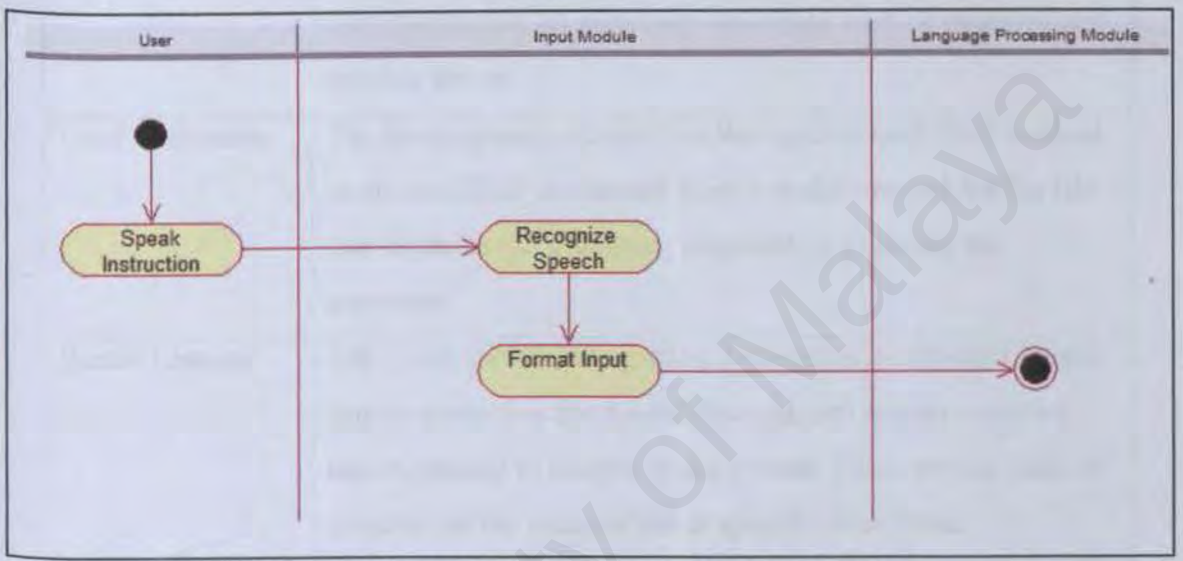


Figure 6.1: Activity Diagram of Input Module

To realize the activities shown in the Figure 6.1, two functional units have created. The description of the functional units is as below:

i. Speech Recognizer

Instruction is received and recognized by the speech recognizer of NLIDS. Basically, the recognizer runs on Microsoft Speech Engine via Java Speech API. NLIDS is capable to convert speech instruction into text form by handling the state generated by the recognizer. Table 6.1 shows the processing state of the recognizer.

Table 6.1: Processing State of Speech Recognizer

State	Handle
Create	The javax.speech.createRecognizer function is called to obtain a speech recognizer. The EngineModeDesc function uses Locale.ENGLISH as an argument to request a recognizer that understands English.
Allocation	The javax.speech.recognition.Recognizer.allocate function is used to allocate all necessary resources such as memory and graphic driver.
Load Grammars	The javax.speech.recognition.Recognizer.loadJSGF method reads in a JSGF document from a reader created for the file that contains the grammar, followed by enabling the grammar.
Result Listener	The javax.speech.recognition.Recognizer.addResultListener function attaches the SpeechRecognizerListener interface that is created to receive result events. These events indicate progress as the recognition of speech takes place.
Committing	Grammars are committed by calling the javax.speech.recognition.Recognizer.commitChanges function. The commit is required for changes to affect the recognition process.
Focus and Resuming	The recognizer must be in the RESUMED state and must have the speech focus in order to recognize the grammar. The javax.speech.recognition.Recognizer.request.Focus and javax.speech.recognition.Recognizer.request.resume functions are used to achieve this function.
Result Acceptation	RESULT_ACCEPTED event is provided when the recognizer completes recognition of the input speech that matches an active grammar.
Result Rejection	RESULT_REJECTED event is provided when the

	recognizer can not match an active grammar with input speech.
De-allocation	The javax.speech.recognition.Recognizer.deallocate function is called to free up the recognizer's resources before the recognizer exits.

In addition to handle the mentioned engine's state, grammar is provided to the speech recognizer engine to define the words that a user can say, and the patterns in which those words can be spoken. The grammar is defined using the Java Speech Grammar Format. Figure 6.2 shows the sample of the speech rule used in NLIDS. The grammar has a single public rule called "Command". The rule accepts several parameters such as <Polite>, <Action>, <Adjective>, and <Object>. Each of the parameters defines a set of acceptable words.

```
#JSGF V1.0;
// Define the grammar name
grammar SimpleCommands;
// Define the rules
public <Command> = [<Polite>] <Action> <Adjective> <Object> (and
<Object>)*;
<Action> = turn | switch | dim;
<Adjective> = on | off;
<Object> = the light | the fan | the TV | the radio;
<Polite> = please|could you
```

Figure 6.2: Sample of Speech Recognizer's Grammar

ii. Input Formatter

The input formatter is responsible for replacing a predefined pattern found in the raw input to a more system friendly pattern. For example, the multiple spacing between words in the raw input will be replaced with single spacing; acronym will be replaced to the full word such as "TV" replaced with "television". Table 6.2 below shows the matching patterns with their respective replacing words in a regular expression that is used by the formatter.

Table 6.2: Regular Expression of Input Formatter

Matching Pattern	Replacing Word
\\s++	[[:SPACE:]]
\\p{Punct}	""
\\b[Tt][Vv]	Television
\bthe light|\\blight	the light
\bthe fan|\\bfan	the fan
\bthe television|\\btelevision	the television
\bthe radio|\\bradio	the radio

6.2.1.2 Response Module

System responses are returned to the user via this module. The module consists of several components which are:

- Command log area
- Debugging message area (designed for system debugging purpose)
- Device status table
- Error message prompt
- Voice synthesizer

Figure 6.3 outlines the activity diagram of the response module. The processed input from Language Processing Module would be directed to response components which are Log area, voice synthesizer, debugging area, and error message prompt, and device status table.

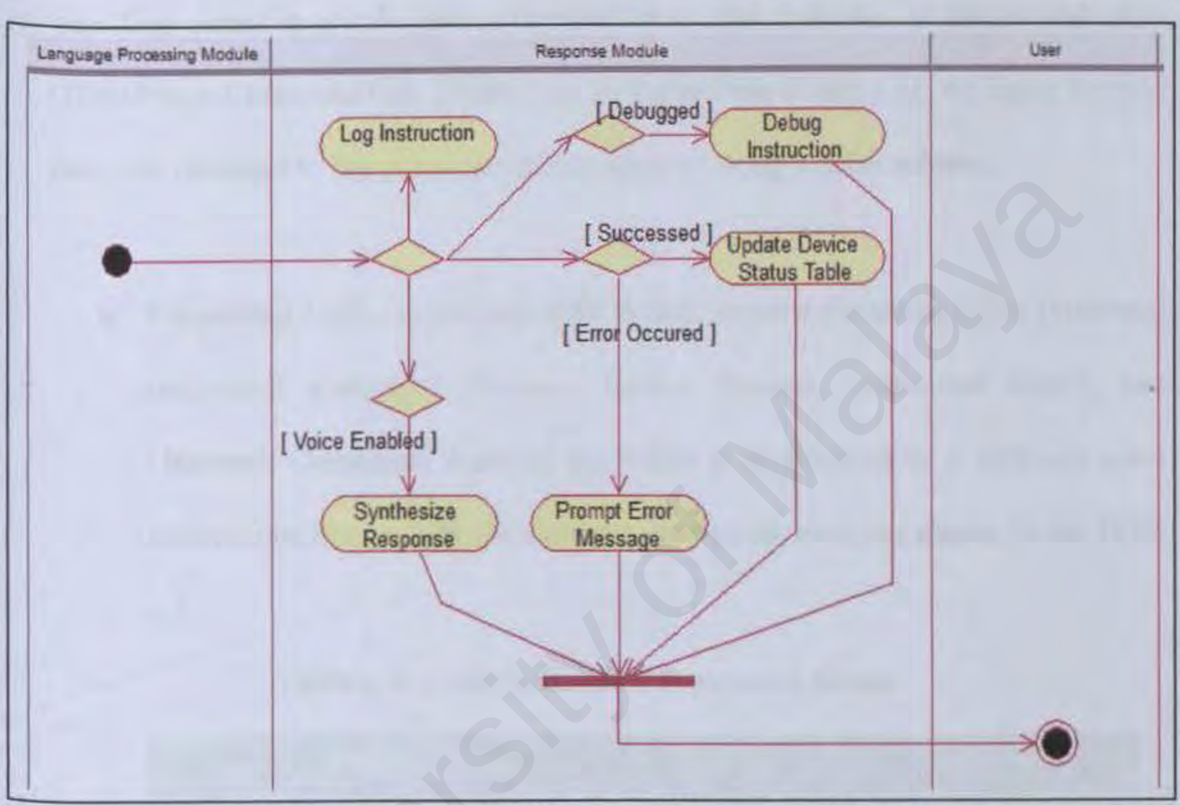


Figure 6.3: Activity Diagram of Response Module

The Log area is designed for logging the commands and also showing processing detail of commands given by the user for tracking purposes. Meanwhile, the voice synthesizer is developed using Java Speech Synthesizer API for converting command responses to speech. Other than the components, we also incorporate error message prompts to display error information, device's status table to show the current state of the implemented virtual device, and also an optional debugging area that is used to show the debug information. The details of the implementation of each component are as follow:

i. Log Area

The Log Area is a tab pane (JTabbedPane) that consists of three text pane (JTextPane): Command Log, Detail Log, and also Parsed Text Log. By using the text pane, we managed to log necessary information by using a color scheme.

a. **Command Log** – This component is used to show the status of the processed command: Command Success, Partial Success, Command Failed, and Unknown Command. Each of the status is highlighted in a different color scheme in order to differentiate the nature of the status as shown in the Table 6.3.

Table 6.3: Color Scheme of Command Status

Status	Scheme	Example
Command Success (CmdSucc)	Blue	**CmdSucc:** Turn on the light and fan
Partial Success (CmdPSuc)	Orange	**CmdPSuc:** Turn on the TV and switch the light
Command Failed (CmdFail)	Red	**CmdFail:** Turn light and fan
Unknown Command (UnknCmd)	Red	**UnknCmd:** TV and Radio

b. **Detail Log** – Each command that is logged into the Command Log will have a corresponding log in the Detail Log component. The Detail Log component is used to display the detailed information of the command status. Table 6.4 shows the reason hash table that is used as a mapping table of command status while Table 6.5 shows the example of the command that logs into the Command Log and Detail Log.

Table 6.4: Reason Hash Table of Command Status

Status	Sub-status	Mapping Description
CmdSucc	CMD_ACT	Device is activated
	CMD_DEACT	Device is deactivated
	CMD_INSTATE	Device is already in the state
	CMD_SPEEDCHG	Speed is changed
	CMD_CHANNELCHG	Channel is changed
CmdPSuc	NIL	Command partial success
CmdFail	NIL	Not supported action
UnknCmd	NIL	Unknown Command

Table 6.5: Sample of Command Log and Detail Log

Command Log	Detail Log
CmdSucc: Turn on the light and fan	**Processed:** turn on light fan -Device is activated
CmdPSuc: Turn on the TV and switch the light	**Processed:** turn on television -Device is activated **Failed:** switch light -Not supported action - One attribute is needed. E.g. On

CmdFail: Turn light and fan	Failed: turn light fan -Not supported action - One attribute is needed. E.g. On
UnknCmd: TV and Radio	Failed: TV and Radio -Unknown Command

c. **Parsed Text Log** – The component is used to display the LinkGrammar's constituent tree of a command. There are up to five types of constituent trees that are possible to be displayed for each command. And in order to increase the readiness of the Log, only one constituent tree is displayed by default.

```
Command: Turn on the light and the fan
CT 1:
(S (VP turn
     (NP (PP (PP on)
           the light and
           (PP then))
        the fan)))
CT 2:
[S [VP turn [NP [PP [PP on PP] the light and [PP then PP] PP] the fan NP] VP] S]
CT 3:
(S (VP turn (NP (PP (PP on) the light and (PP then)) the fan)))
CT 4:
LEFT-WALL turn.v on the light.n and then.e the fan.n
CT 5:
( ( turn ( ( ( on ) the light and ( then ) ) the fan ) ) )
```

Figure 6.4: Constituent Tree of a Parsed Command

ii. Voice Synthesizer

The voice output is generated using a speech synthesizer engine of NLIDS. Basically, the speech synthesizer engine runs on the Microsoft Speech Engine and is accessed via Java Speech API which is similar to the speech recognizer mentioned earlier. By handling the generated synthesizer's state, NLIDS is capable of generating synthesized speech that is based on the reason hash table mentioned in Table 6.4. Table 6.6 shows the synthesizer's state that is implemented in NLIDS.

Table 6.6: Speech Synthesizer Processing State

State	Handle
Create	The javax.speech.Central.createSynthesizer function is triggered to obtain a speech synthesizer Since the domain language of NLIDS is in English, so we are using SynthesizerModeDesc= Locale.ENGLISH as an argument in order to request a synthesizer that manages to generate synthesized speech in the English Language.
Allocation	The javax.speech.synthesis.Synthesizer.allocate function is called to request the synthesizer to allocate all necessary resources.
Resuming	The synthesizer must be in the RESUMED state in order to generate synthesized speech. The javax.speech.synthesis.Synthesizer.resume function is used to achieve this.
Generation	The generation of synthesized speech from a string is triggered by calling the javax.speech.synthesis.Synthesizer. speakPlainText function.
De-allocation	The javax.speech.synthesis.Synthesizer.deallocate function is called to free up the synthesizer 's resources before the synthesizer exits.

iii. Debug Area

This component is a Java Text Area and used to show debug information. It will be ceased from the normal running mode of NLIDS unless the "-debug" flag is specified when starting the program. Several debug information such as the processing rule of an input and the breakdown of a complex input will be dumped into the component.

iv. Error Message Prompt

NLIDS will handle some known exceptions. The exceptions will be caught and the related information will be prompted to the user via the Error Message Prompt. Basically, the Error Message Prompt is just a Java Message Dialog (JOptionPane.showMessageDialog) with a set of default attributes. Users are required to response to the dialog box before they are allowed to trigger a new input if any exception occurs for the current input.

v. Device Status Table

In addition to the graphical visualization of the device's status, a table is created to display the current status of the devices in our system. By using the key-pair value strategy, we could easily identify the state of the desired device and also set the corresponding device's state to a particular value. The table is constructed using the Java Table Model and is predefined with three columns which are Device, State, and Speed/Channel accordingly. The table is initialized with a data structure which holds the initial status of devices as shown in the Table 6.7 below when NLIDS is starting-up.

Table 6.7: Device Status Table

Device	State	Speed/Channel
Television	OFF	1
Radio	OFF	2
Fan	OFF	3
Light	OFF	N/A

6.2.2 Implementation of Language Processing Module

The Language Processing Module is the "brain" of NLIDS that is responsible for user input parsing, grammar checking of parsed input, devices' status validation, devices' status updating, command log composition, visualization triggering, and response generation. Figure 6.5 shows the activity diagram of Language Processing Module.

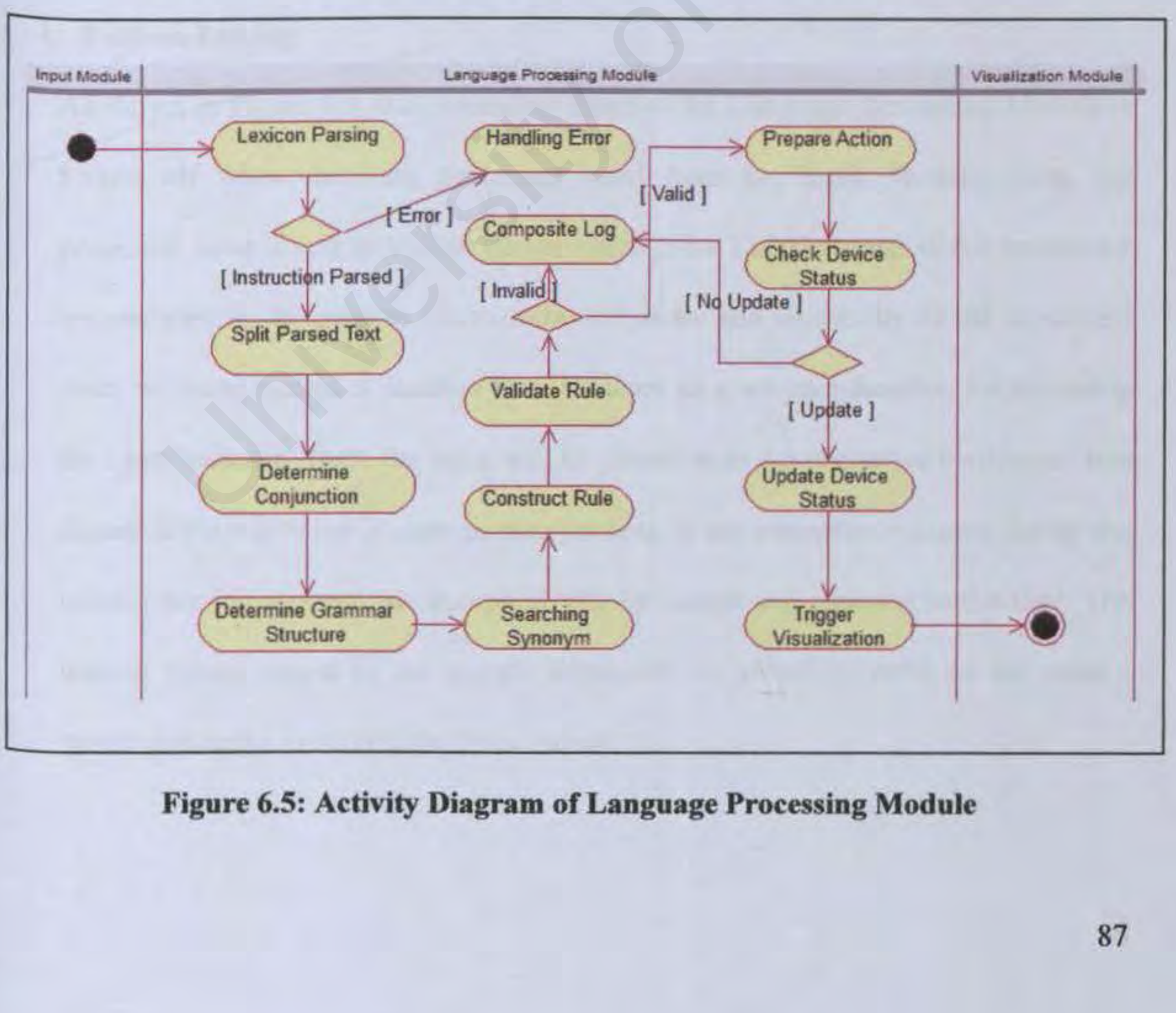


Figure 6.5: Activity Diagram of Language Processing Module

Generally, the processing steps in the Language Processing module could be divided into 3 main stages, namely preliminary stage, processing stage, and execution stage. In order to ease the understanding of processing stages implemented in the language processing module, a sample instruction: "switch on the radio and turn off the fan" is used to examine the output of each stage.

6.2.2.1 Preliminary Stage

Generally, the preliminary stage consists of several processes that work together to construct the processed input from the input module to a structured format that is required at the next stage in the Language Processing Module's processing chain.

i. Lexicon Parsing

As shown in Figure 6.5, the processing chain of the Language Processing Module is kicked off when receiving processed input from the Input Module. First, the processed input is sent to lexicon parser which is the LinkGrammar in our project for lexicon parsing. In order to increase the simplicity and reusability of the developed code, we have created a function that is known as a wrapper function for accessing the LinkGrammar. Then, the input will be parsed in to the respective constituent tree format at the end of the lexicon parsing process. If any exception occurred during the lexicon parsing process, the exception will be caught and reported to the user. The lexicon parsed output of the sample instruction is: switch.v (verb) on the radio.n (noun) and turn.v (verb) off the fan.n (noun).

ii. Parsed Text Splitting

Once the processed input has been parsed successfully, the corresponding constituent tree is then put back into the processing chain and sent to the sentence splitting process. The sentence splitter is responsible for splitting the complex sentences into a simple sentence based on the notation information provided in the LinkGrammar's output. This process is necessary in order to reduce the complexity of the parsed input so that the processing in the following processes is more effective as well as easy. The split sample instruction is as following:

```
(S (S (VP (ADVP turn)
      on
      (PP the
         (NP radio))))
and
(S (VP (ADVP then)
      turn
      (PP off
         (NP the fan)))))
```

*The S, VP, ADVP, PP, NP are the type of linkage defined by the LinkGrammar. The detail of each link type could be found at LinkGrammar's documentation (Sleator and Temperley, 1991).

6.2.2.2 Processing Stage

This stage is responsible for processing command requirement before generating the response back to user. The stage consists of Conjunction Determination, Grammar Structure Determination, Synonym Searching, Rules Construction, and Rules Validation.

i. Conjunction Determination

The split input will be processed at the conjunction determination process in order to determine the relationship of each single input of a complex input. For instances, the relationship is "and", "before", and after which requires the NLIDS to process the split commands in a specify order. The conjunction information will then be piped to grammar structure determination process for further processing. The determined conjunction for the split sample instruction is "and".

ii. Grammar's Structure Determination

At the grammar structure determination process, the NLIDS will identify the grammar of each lexicon in the parsed input. For instance, a command "please turn on the light", the grammar of each lexicon is as follows: please is an adverb, turn is a verb, and light is a noun. The process is crucial for the NLIDS as the following processes in the language processing module such as synonym searching and rules construction depends on the grammar information for processing. The grammar of the split sample instruction would be evaluate based on the suffix added to each lexicon in the earlier stage.

iii. Synonym Searching

The language processing chain continues at the synonym searching process for searching lexicons that are unknown to the NLIDS. This process is required as NLIDS needs to "understand" the lexicons that are not hard-coded into the system but have similar meaning to the system. In addition, the hard-coded lexicon is kept as minimum as possible so that the system is more intelligent in that sense. The unknown lexicon will be searched for its meaning and compared with the set of hard-coded lexicon in the system. And if the searching is positive, the "understandable" lexicon will be passed to the rule construction's function. NLIDS uses the WordNet discussed in Chapter Two as the search reference. At this stage, the synonym of each lexicon in the split sample instruction would be validated against the hard-coded lexicon.

iv. Rule Construction & Rule Validation

Once we have determined the synonym of the parsed input, the processing chain will be continued at the rule construction process. This process is dedicated to construct a rule for parsed input that is passed from the lexicon parsing function. The constructed rule will be cross-checked with the predefined rule and the entire language processing chain will only be continued if the cross-checking result is positive. On the other hand, if the constructed rule does not map to any predefined rules, the language processing chain will be stopped and the corresponding input will be logged into the command log area and the unknown command error message and its details are shown to the user. The synonym validated sample instruction would be cross-checked against the predefined rule at this stage.

6.2.2.3 Execution Stage

The execution stage consists of Action Preparation, Device Status Checking, Device Status Updating, and Visualization Triggering. The function of this stage is to execute users' commands that have been structured in the earlier stage.

i. Action Preparation

The processing chain is continued at the Action Preparation process that is responsible for composing the action string. The action string consists of four parts which are verb, attribute, target, and parameter. Each part represents an action against a device, what is going on against the device's state, what is the device, and additional parameters for devices. For instance, a user's command: "Please switch the TV channel to channel 1" will be converted to an action string that consists of verb: switch; attribute: channel 1: target: TV; and parameter: channel. The composed action string enables the functionality of the next process - device status checking process in the language processing module.

ii. Device Status Checking & Device Status Updating

The device status checking process is responsible for checking the current status of a device that is specified in an action string, before updating the device status occurs, to avoid unnecessary action to a device. For instance, if a device is in the "on" mode, and if a command requests the device to be turned on again, the NLIDS shall response to the user that the device is already "on". Thus, the "on" action will be avoided to be sent to the consequence process. Although sending a request that updates a device status to the current status causes no harm in NLIDS, but it should be avoided in a real world as this might cause a real device becomes malfunction.

A particular device's status is determined by referring to the device's status table that is discussed in the response module section. If the action string requires a status change of a device, the device status updating function will update the device status table and send the new status to the visualization triggering process to render the corresponding visual in the NLIDS. Otherwise, if updating a device status is unnecessary, the action string will only be logged to the command log with the respective details and the processing of the input command is considered completed at this point.

iii. Visualization Triggering

This process acts as the interface with the visualization module. It is responsible for converting the action string to the visualization command for the rendering of the action to be possible.

6.2.3 Implementation of Visualization Module

The Java 3D recipe for writing Java 3D programs is the rule of thumb for the implementation of visualization module. In addition to that, NLIDS has provided a Status

Updating functionality in the visualization module for updating device's status. Figure 6.6 shows the activity diagram Visualization module.

The processes in the Visualization Module could be divided into pre-processing processes and real-time processes according to the process's life span in the NLIDS. The pre-processing processes such as Canvas3D Creation, Simple Universe Creation, Content Branch Creation, Content Branch Compilation, and Content Branch Integration are used for visual environment initialization. The processes are executed only when NLIDS is initializing. On the other hand, the real-time processes – Status Updating is responsible as an interface for the Language Processing Module to update a particular device's status. Both pre-processing processes and real time processes are requested for the visualization rendering process to render visualization and display the visualization to the user.

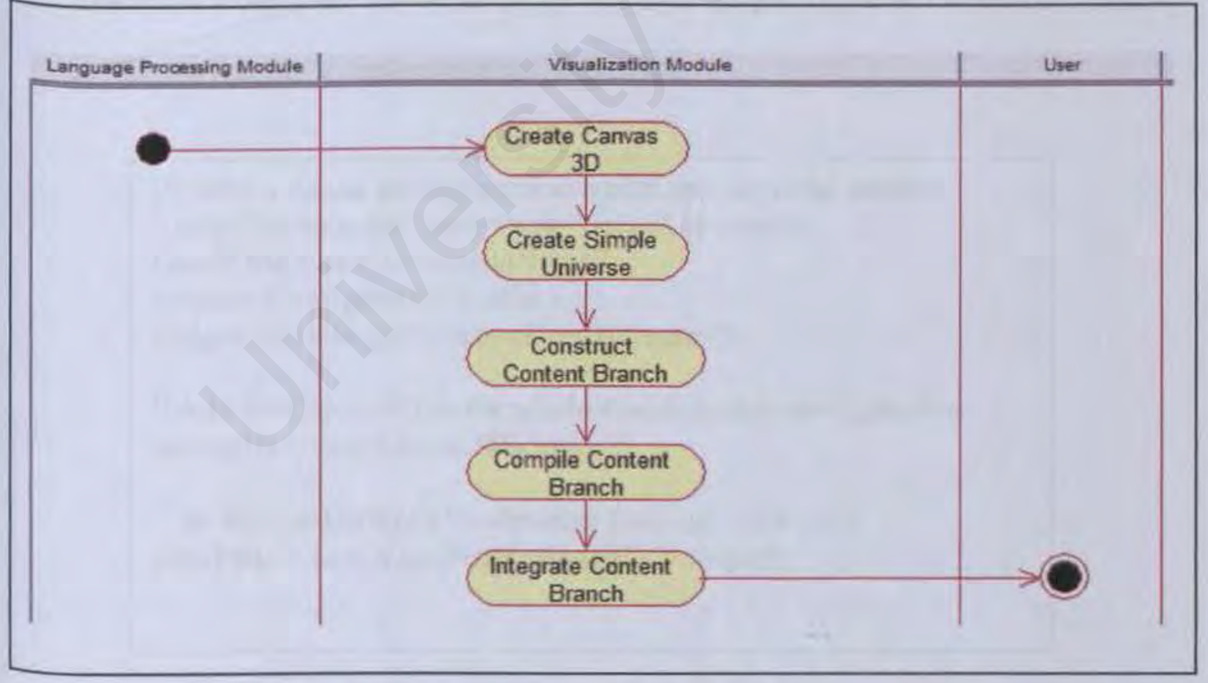


Figure 6.6: Activity Diagram of Visualization Module

6.2.3.1 Pre-processing Process

In this process, we would render the virtual devices such as light, fan, TV, and radio in a virtual space. As mentioned, the rendering process is constructed using the Java 3D recipe's approach. The following section describes the action taken for each step of the recipe.

i. Canvas 3D Creation

The process is to create a container for holding all virtual objects which includes the virtual devices and also the environment details of NLIDS. This is a simple but important process for Java 3D engine to identify objects that need to be rendered in the virtual environment. The canvas 3D is created with default configuration to define the relationship with view plate of NLIDS. In addition, the canvas 3D serves as a reference in the view branch graph of the scene graph. Figure 6.7 shows the snapshot of the Canvas 3D's implementation in NLIDS.

```
//Create a virtual environment as a platform for virtul objects.
VirtualUniverse universe = new VirtualUniverse();
Locale loc = new Locale(universe);
GraphicsConfiguration config =
SimpleUniverse.getPreferredConfiguration();

// Add the Canvas3D to the window with default configuration
canvas3D = new Canvas3D(config);

// set the Canvas3D to be viewable from our view plate.
viewPlate = new ViewPlate(canvas3D, viewport);
```

Figure 6.7: Implementations of Canvas3D and Simple Universe in NLIDS

ii. Simple Universe Creation

NLIDS uses the SimpleUniverse class as a single step to perform the several necessity tasks such as creating the virtual universe object, creating a locale object and attaching to the virtual universe, and constructing the view object when creating a view branch graph structure. Figure 6.7 shows the snapshot of the implementation of Simple Universe in NLIDS.

iii. Content Branch Construction

All virtual devices created in NLIDS need to have a corresponding branch group according to Java 3D specifications. We have created several helper functions for creating the branch group for virtual devices such as light, TV, radio, and fan. Besides that, we are also required to create branch groups for virtual environment details such as ambient lights and sound. Figure 6.8 shows the snapshot of the implementation of Branch Group Construction in NLIDS.

```
BranchGroup viewport = new BranchGroup();
viewPlate = new ViewPlate(canvas3D, viewport);
loc.addBranchGraph(viewport);

rootBG.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
rootBG.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
scene = new Scene(250.0f, 0.0f, 100.0f, 130.0f, 40.0f, 65.0f);

// Add virtual objects brach group to the root branch group
rootBG.addChild(scene.createRoom(0.0f,0.0f,0.0f,500.0f,300.0f,700.0f));
rootBG.addChild(scene.createFan(250.0f, 250.0f, 250.0f, 40.0f, 20.0f));
rootBG.addChild(scene.createDevice(80.0f, 45.0f, CommonData.TV));
rootBG.addChild(scene.createDevice(50.0f, 25.0f, CommonData.RADIO));
rootBG.addChild(scene.createLights(15.0f, 250.0f, 120.0f, 80.0f));
```

Figure 6.8: Implementations of Branch Groups in NLIDS

iv. Content Branch Compilation

The Branch Group object has a method, namely compile(). This method is used to compile objects that rendered under the branch group and create as well as cached a compiled scene graph. The scene graph is kind of internal representation of rendering attribute in Java 3D. Figure 6.9 shows the snapshot of the implementation of Content Branch Group Construction in NLIDS.

```
Locale loc = new Locale(universe);
BranchGroup viewport = new BranchGroup();
viewPlate = new ViewPlate(canvas3D, viewport);
loc.addBranchGraph(viewport);
 // Compile and add the rootBG
rootBG.compile();
//Combine the content Brach Group into virtual space.
loc.addBranchGraph(rootBG);
```

Figure 6.9: Implementations of Content Branch Group Compilation in NLIDS

v. Content Branch Integration

This step is needed to integrate the created virtual environment with the main user interface of NLIDS. The Canvas 3D that holds the virtual environment detail will be put into a Java Panel for displaying purpose. Figure 6.10 shows the snapshot of the implementation of Content Branch Integration in NLIDS.

```
 private JPanel drawingPanel;
drawingPanel = new javax.swing.JPanel();
drawingPanel.setLayout(new java.awt.BorderLayout());
drawingPanel.setPreferredSize(new java.awt.Dimension(500, 400));
drawingPanel.add(canvas3D, java.awt.BorderLayout.CENTER);
```

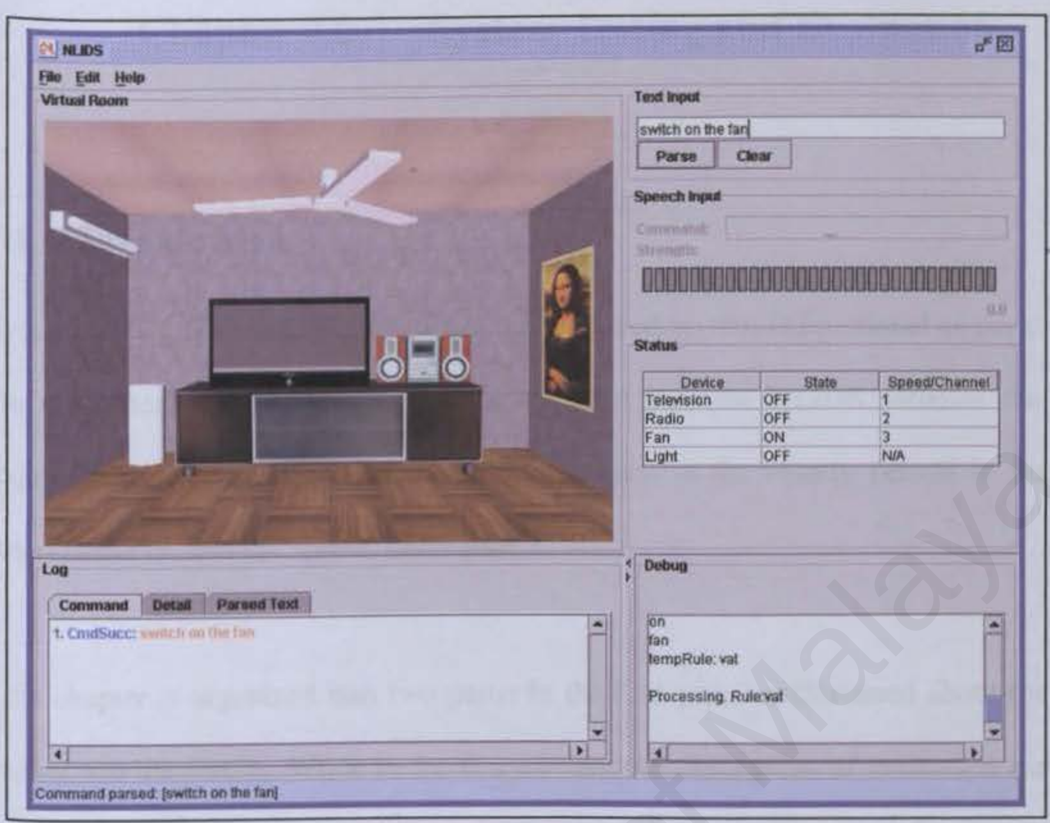
Figure 6.10: Implementations of Content Branch Group Integration in NLIDS

6.2.3.2 Real Time Process

The details of the rendered virtual environment in the Pre-processing Process will be updated according to the user's command. The status updating function and Java built-in visualization thread are responsible for the updating. The updating function is created as generic as possible so that only one function is needed for updating all virtual objects. Figure 6.11 shows the snapshot of the implementation of the status updating function while Figure 6.12 shows the screen shot generated by the virtualization module.

```
public boolean updateDeviceStatus(int targetDevice, int state, int behavior){
    switch (targetDevice)
    {
        case CommonData.TV:
        case CommonData.RADIO:
              scene.updateDevice(targetDevice, state, behavior);
              break;
        case CommonData.FAN:
              scene.updateFanSpeed(state, behavior);
              break;
        case CommonData.LIGHT:
              scene.updateLight(state);
              break;
        default:
              System.out.println("Invalid target!");
    }
    return CommonData.SUCCESS;
}
```

Figure 6.11: Implementations of Status Updating in NLIDS



switch on the fan

Device	State	Speed/Channel
Television	OFF	1
Radio	OFF	2
Fan	ON	3
Light	OFF	N/A

```
1. CmdSucc: switch on the fan
```

```
on
fan
tempRule: vat

Processing. Rule:vat
```

```
Command parsed: [switch on the fan]
```

Figure 6.12: Screenshot of NLIDS

6.3 Conclusion

This chapter discussed the techniques used in the implementation of the three modules of NLIDS, namely Input & Response Module, Language Processing Module, and Visualization Module. Subsequently, the configuration of the system also has been discussed.

Chapter 7.0 System Testing and Evaluation

7.1 Introduction

System testing for NLIDS is to ensure the proposed system is functional as per designed. On the other hand, system evaluation certifies that the NLIDS satisfies the system requirement where it should provide the facilities to the elderly person in instructing smart home system via speech instruction.

This chapter is organized into two parts. In the first part, we discussed about the system testing and the results. While in the second part, the description of evaluation carried out on selected user group is discussed.

7.2 System Testing

System error could reside without being noticed during the design phase and still undiscovered in system implementation phase. The system testing is conducted to expose the system errors before it causes trouble in real-world program execution. The system testing has been carried out in two phases, namely unit & module testing, and functional testing. Generally, unit & module testing is to test on each unit and the interfacing between the modules, while the functional testing is to trace back the system requirements defined in the use cases shown in Figure 4.2. Details of these testing are discussed in the following sections.

7.2.1 Unit & Module Testing

Unit testing and module testing have been combined and done in parallel for the three modules of NLIDS. In this regards, each module has been feed with predefined instructions and verified whether the module is producing the expected result. The predefined input data for each test case are listed in Appendix B. As shown in Table 7.1, the unit testing was carried out on each of the functional unit in different modules. The results obtained from each functional unit testing in the table are explained at the following section.

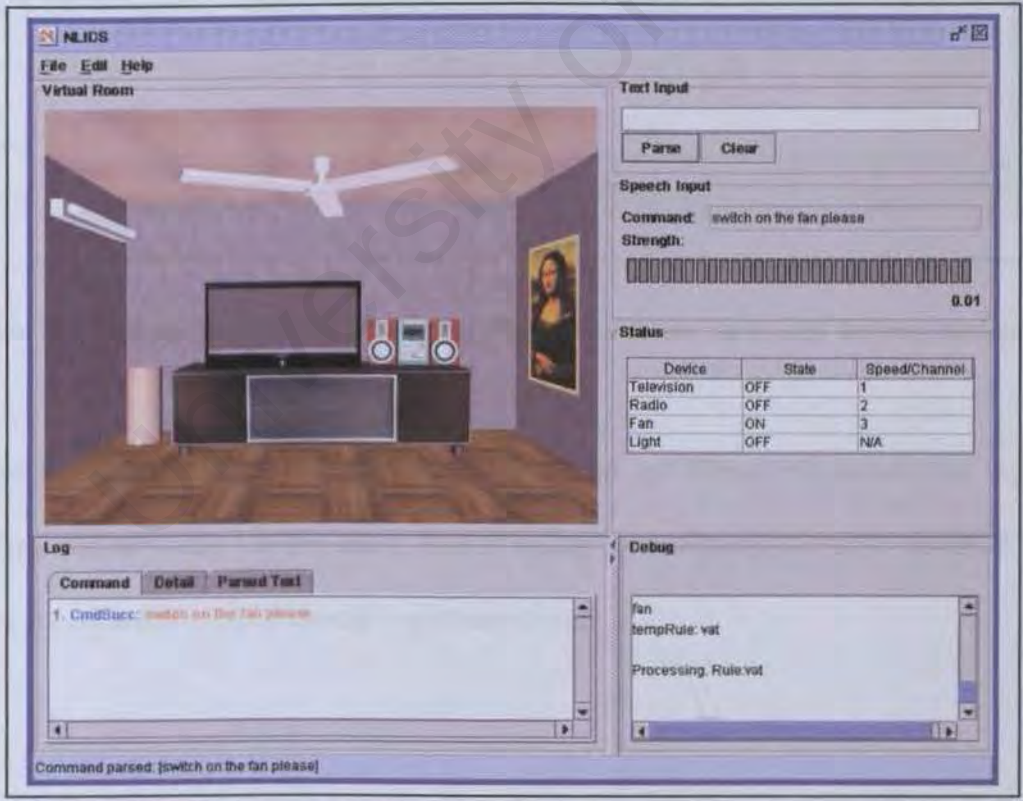
Table 7.1: Result of Unit & Module Testing of NLIDS

Module	Unit	Test Case	Test Procedure	Test Result
Input & Response	Speech Recognition	1.1a	Read domain related sentences.	Speech recognized. Respective input shown.
		1.1b	Read domain unrelated sentences.	"Unrecognized speech" shown.
	Input Formatter	1.2	Read domain related sentences.	Formatted input according Table 6.1 shown.
	Log Area	1.3	Read text input.	Input is logged.
	Voice Synthesizer	1.4	Read domain related sentences.	Respective sentence was synthesized by system.
		1.5	Read domain unrelated sentences.	"Unrecognized speech" synthesized by system.
	Debugging Area	1.6	Read domain related sentences.	Input was parsed into selected constituent tree.
	Device Status Table	1.7	Read domain related sentences.	Respective device's state has been updated.

Language Processing	Lexicon Parsing & Grammar Structure Determination	2.1	Read domain related sentences.	The parsed input shows in "Parsed Text Tab".
	Parsed Text Splitting & Conjunction Determination	2.2	Read domain related complex sentences.	The complex input is split into single input and shows in "Detail Tab".
	Rule Construction & Validation	2.3	Read domain related sentences.	The constructed and validated rule is shows in "Debug Tab"
	Device Status Checking & Updating	2.4	Read domain related sentences.	The respective device's state is updated.
	Action Preparation & Visualization Triggering	2.5	Read domain related sentences.	Prepared action sent to visualization module for processing.
Visualization	Pre-processing State	3.1	Start NLIDS.	Virtual environment rendered once NLIDS started.
	Real Time State	3.2	Read domain related sentences.	Respective device's animation is rendered.

7.2.1.1 Input & Response Module

For Input & Response module, as shown in the Table 7.1, the functional units namely Speech Recognizer, Input Formatter, Log Area, Voice Synthesizer, Debugging Area, and Device Status Table were tested. For example, a test case - "switch on fan please", is recognized by the Speech recognizer and the recognized input was shown in the "Speech Input text field" as in Figure 7.1. Additionally, the Input Formatter mapped the noun "fan" to "the fan" per the regular expression rule defined in the Table 6.2. Besides that, the formatted input from "Input Formatter" was displayed in the "Log Area". "Device Status" table and the "Debugging Area" were updated accordingly. The response from the proposed system was synthesized to the user to indicate that the command has been successfully performed.



Command: switch on the fan please
Strength:
0.01

Device	State	Speed/Channel
Television	OFF	1
Radio	OFF	2
Fan	ON	3
Light	OFF	N/A

```
1. CmdSucc: switch on the fan please
```

```
fan
tempRule: vat

Processing. Rule:vat
```

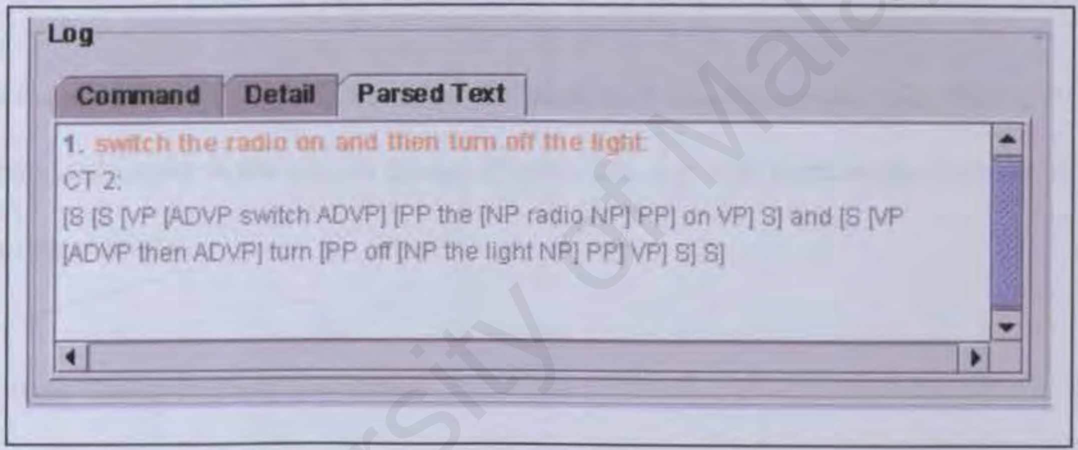
```
Command parsed: [switch on the fan please]
```

Figure 7.1: Sample Output of Unit & Module Testing for Input & Response Module

7.2.1.2 Language Processing Module

For the Language processing Module, as shown in the Figure 6.6, five main functional units, namely Lexicon Parsing & Grammar Structure Determination, Parsed Text Splitting & Conjunction Determination, Rule Construction & Validation, Device Status Checking & Updating, and Action Preparation & Visualization Triggering have been tested with respective test cases. For example, the output of each functional unit for test case: "switch the radio on and turn off the light" are as below:

i. Lexicon Parsing & Grammar Structure Determination

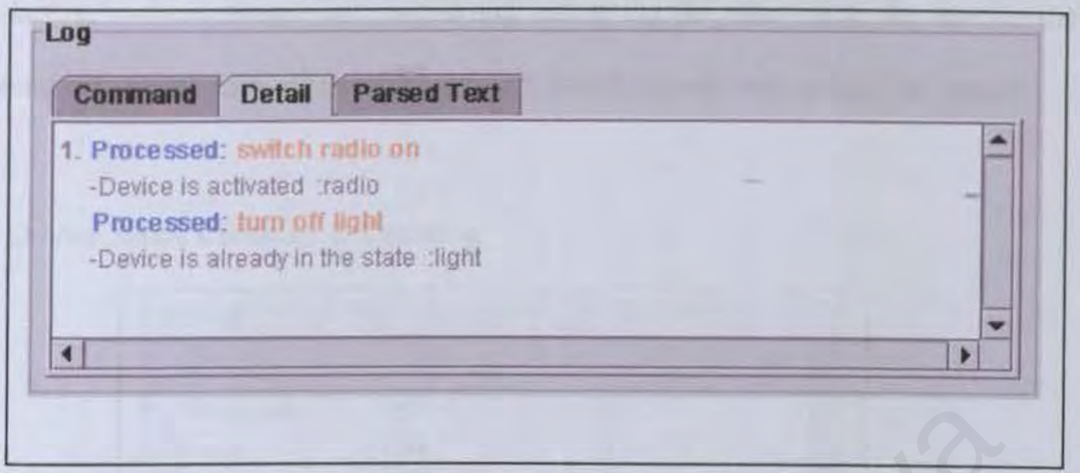


```
1. switch the radio on and then turn off the light
CT 2:
[S [S [VP [ADVP switch ADVP] [PP the [NP radio NP] PP] on VP] S] and [S [VP
[ADVP then ADVP] turn [PP off [NP the light NP] PP] VP] S] S]
```

Figure 7.2: Sample Output of Lexicon Parsing & Grammar Structure Determination Unit

As shown in the Figure 7.2, the unit produced constituent representation assigned syntactic structure for the input. The meaning for assigned syntactic structure could be found at the LinkGrammar's documentation.

ii. Parsed Text Splitting & Conjunction Determination

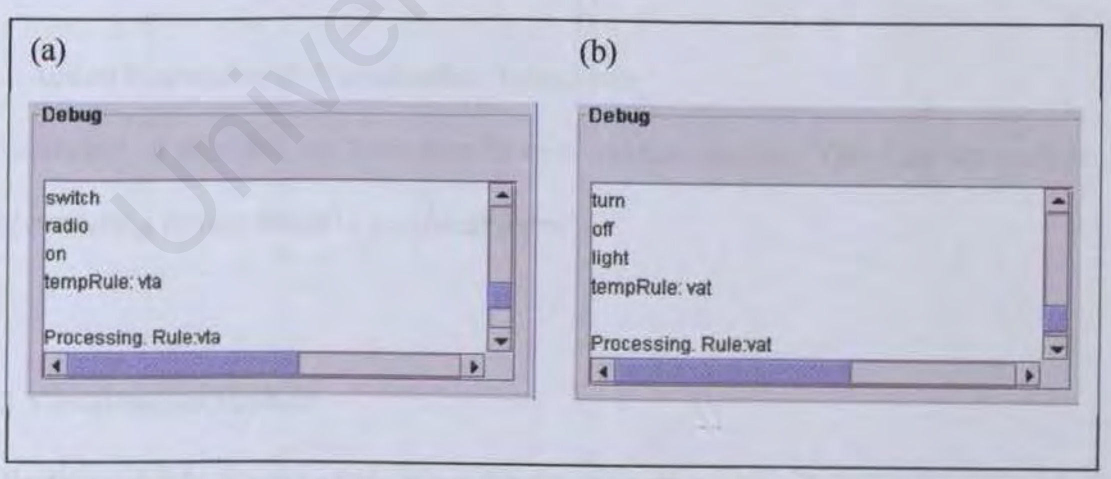


```
1. Processed: switch radio on
   -Device is activated :radio
   Processed: turn off light
   -Device is already in the state :light
```

Figure 7.3: Sample Output of Parsed Text Splitting & Conjunction Determination Unit

As shown in the Figure 7.3, the input has been split into two single instructions. As mentioned earlier in the system design chapter, this step is to increase the integrity of parsing outcome.

iii. Rule Construction & Validation



```
switch
radio
on
tempRule: vta

Processing. Rule:vta
```

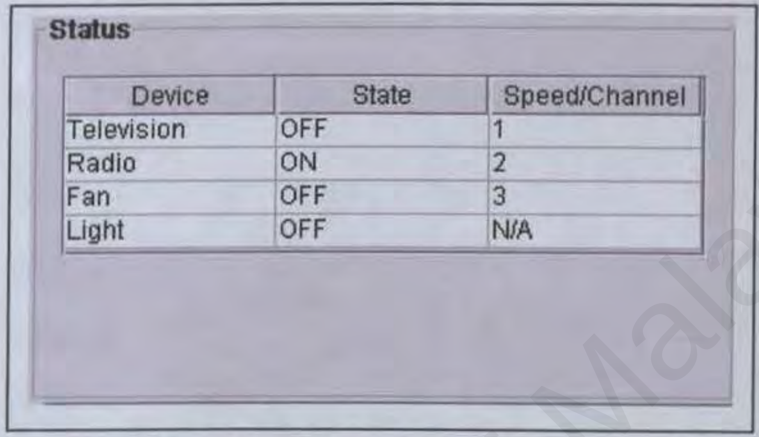
```
turn
off
light
tempRule: vat

Processing. Rule:vat
```

Figure 7.4: Sample Output of Rule Construction & Validation Unit

As in Figure 7.4(a), the rule constructed for first split single input: switch the radio on is "vta", which means verb, target and action. On the other hand, the rule for the second split input: turn off the light is "vat", which means verb, action, and target.

iv. Device Status Checking & Updating



Device	State	Speed/Channel
Television	OFF	1
Radio	ON	2
Fan	OFF	3
Light	OFF	N/A

Figure 7.5: Sample Output of Device Status Checking & Updating Unit

Figure 7.5 shows that the instruction has been processed successfully where radio is turned on while light is turned accordingly.

v. Action Preparation & Visualization Triggering

The output of this unit has been sent to next module, namely Visualization module for rendering device status in graphical form.

7.2.1.3 Visualization Module

Visualization module is responsible to render the respective action in animation form for an instruction. The animation is rendered at the "Virtual Room" section of proposed system. For example, as shown in Figure 7.6, the module rendered an animation with radio is turned on while light is turned off for test case: "switch the radio on and turn off the light".

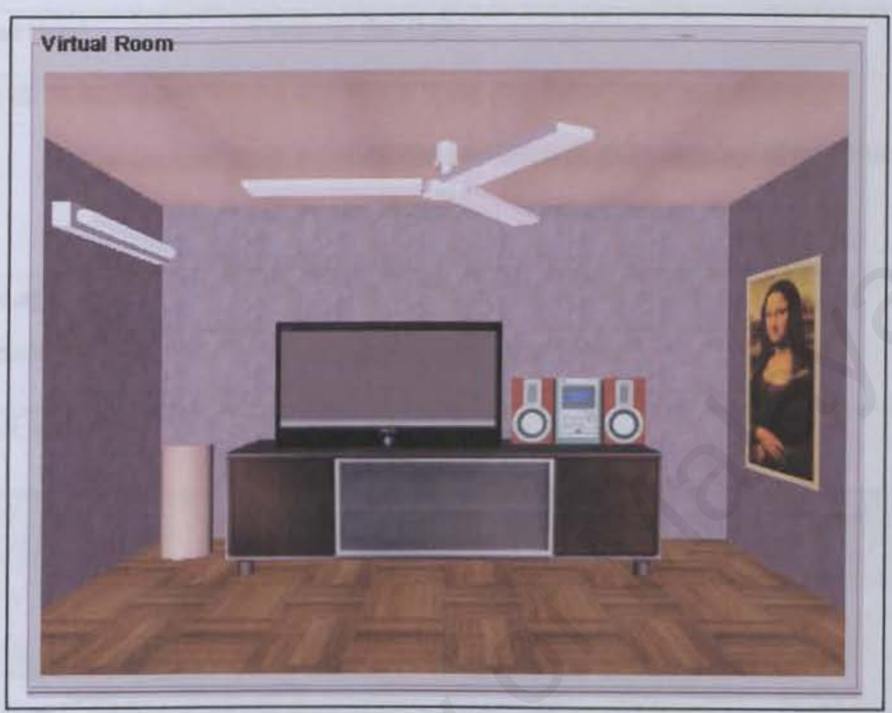
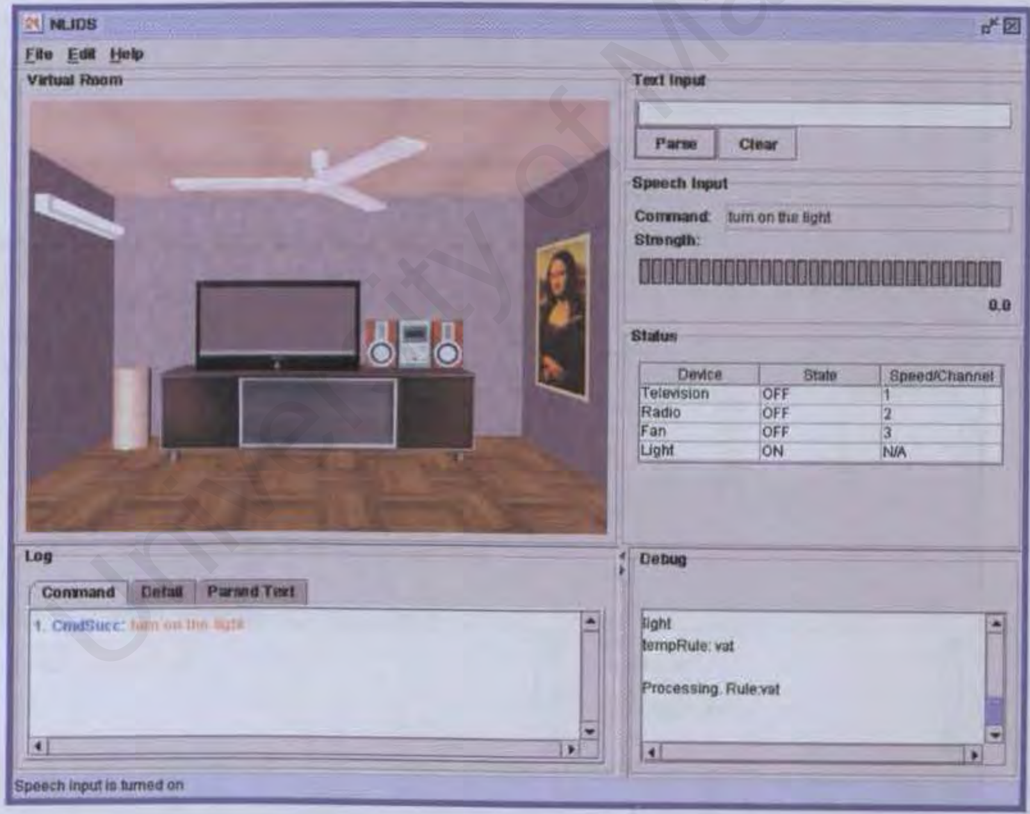


Figure 7.6: Sample Output of Visualization Module

7.2.2 Functional Testing

Functional testing is carried out to test the functionality of the proposed system defined in the use case shown in Figure 4.3. In this regards, several instructions are created as test cases to validate the functionality mentioned in the each use case. The functional test cases and their results are shown in Table 7.2 (a - d) below.

Table 7.2a: Result of Functional Testing for Toggle Light Use Case

1.	**Scenario:**	Instruct proposed system to switch on the light.
	Use case:	Toggle light
	Instruction:	Turn on the light.
	Input & Response:	• Input field displaying the instruction. • Formatted instruction is display at command tab • Notification synthesized to user.
	Language Processing:	• Parsed input is shows at parsed text tab. • Status table is updated.
	Visualization:	• Light is turned on. 

NLIDS

Command: turn on the light
Strength: 0.0

Status

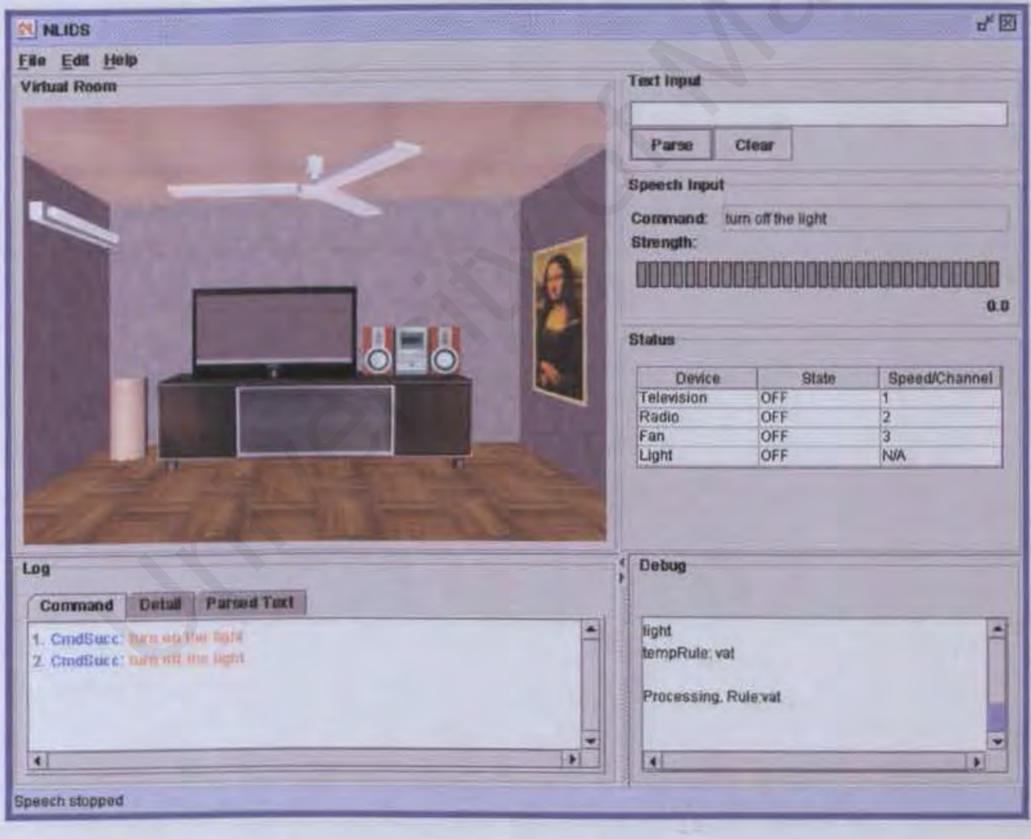
Device	State	Speed/Channel
Television	OFF	1
Radio	OFF	2
Fan	OFF	3
Light	ON	N/A

```
1. CmdSucc: turn on the light
```

```
light
tempRule: vat

Processing. Rule:vat
```

Speech input is turned on

2.	**Scenario:**	Instruct proposed system to switch off the light
	Use case:	Toggle light
	Instruction:	Turn off the light.
	Input & Response:	• Input field displaying the instruction. • Formatted instruction is display at command tab • Notification synthesized to user.
	Language Processing:	• Parsed input is shows at parsed text tab. • Status table is updated.
	Visualization:	• Light is turned off. 

Command: turn off the light

Device	State	Speed/Channel
Television	OFF	1
Radio	OFF	2
Fan	OFF	3
Light	OFF	N/A

```
1. CmdSucc: turn on the light
2. CmdSucc: turn off the light
```

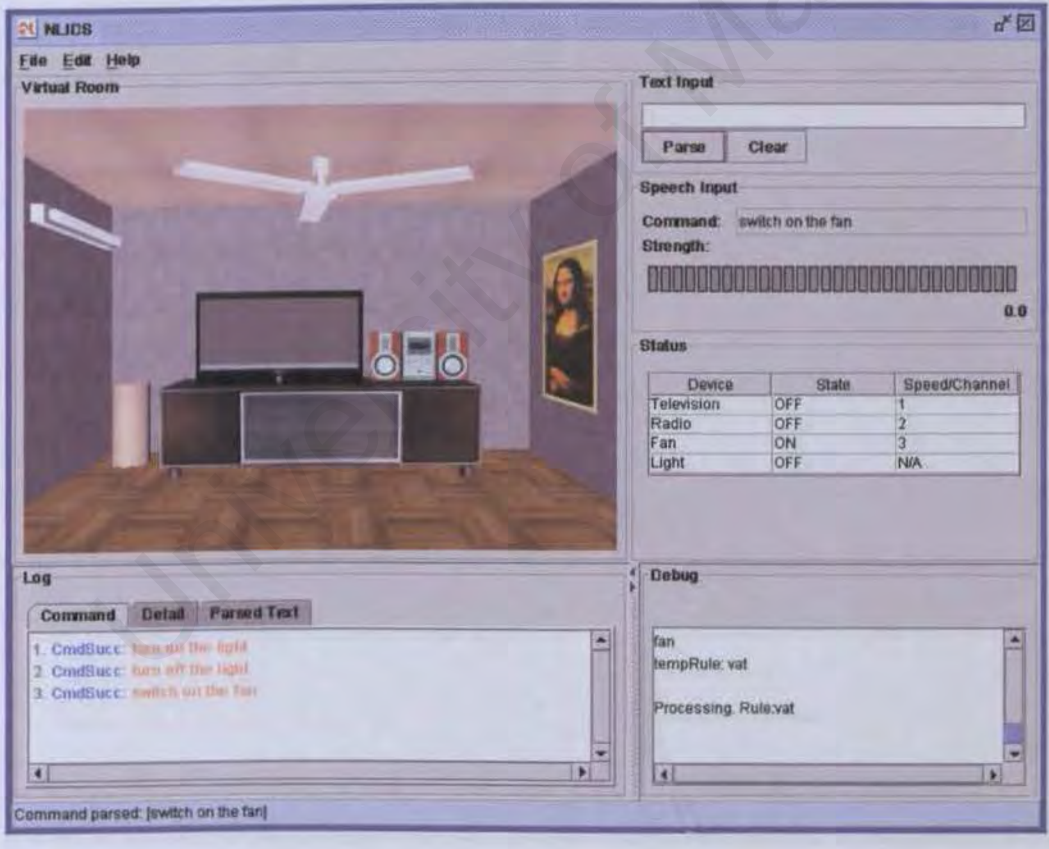
```
light
tempRule: vat

Processing. Rule:vat
```

Speech stopped

Table 7.2b: Result of Functional Testing for Operate Fan Use Case

1.	**Scenario:**	Instruct proposed system to switch on the fan
	Use case:	Operate fan
	Instruction:	Switch on the fan.
	Input & Response:	• Input field displaying the instruction. • Formatted instruction is display at command tab • Notification synthesized to user.
	Language Processing:	• Parsed input is shows at parsed text tab. • Status table is updated.
	Visualization:	• Fan is turned on.



Command: switch on the fan
Strength:
0.0

Device	State	Speed/Channel
Television	OFF	1
Radio	OFF	2
Fan	ON	3
Light	OFF	N/A

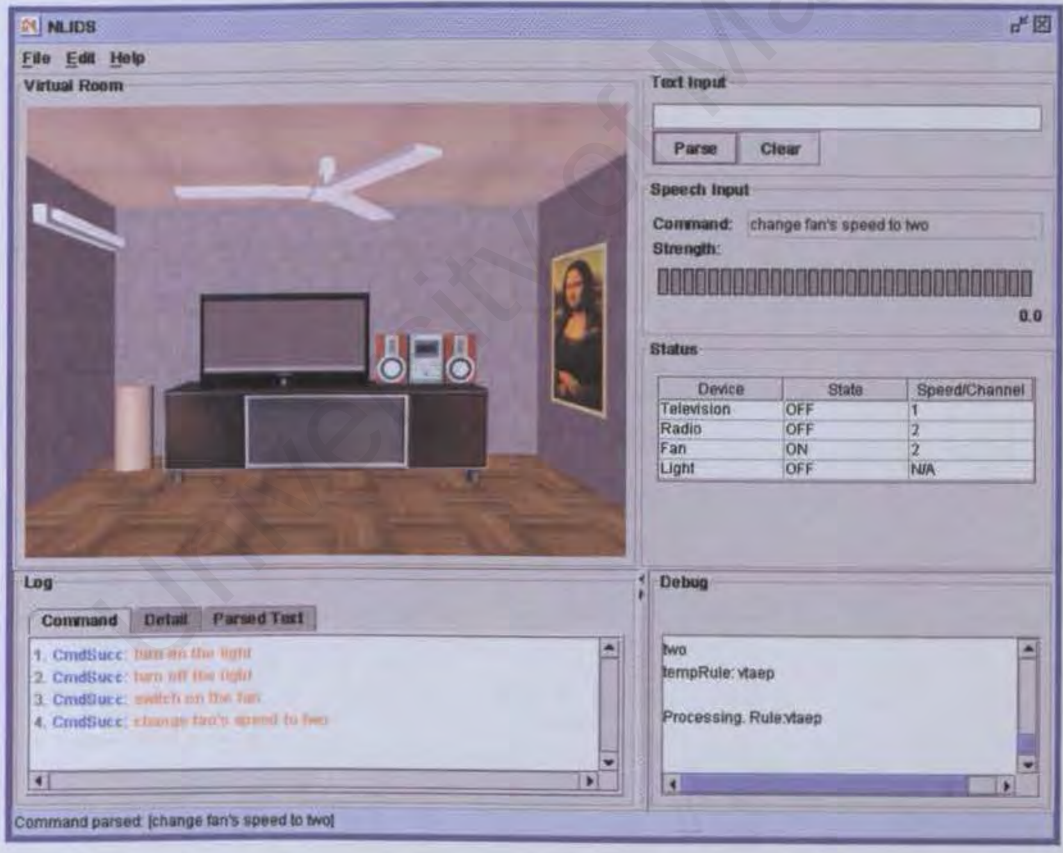
```
1. CmdSucc: turn on the light
2. CmdSucc: turn off the light
3. CmdSucc: switch on the fan
```

```
fan
tempRule: vat

Processing. Rule:vat
```

Command parsed: [switch on the fan]

2.		
	Scenario:	Instruct proposed system to switch fan's speed to two
	Use case:	Operate fan
	Instruction:	Change fan's speed to two.
	Input & Response:	• Input field displaying the instruction. • Formatted instruction is display at command tab • Notification synthesized to user.
	Language Processing:	• Parsed input is shows at parsed text tab. • Status table is updated.
	Visualization:	• Fan's speed is switched to two.



Command: change fan's speed to two
Strength:
0.0

Status

Device	State	Speed/Channel
Television	OFF	1
Radio	OFF	2
Fan	ON	2
Light	OFF	N/A

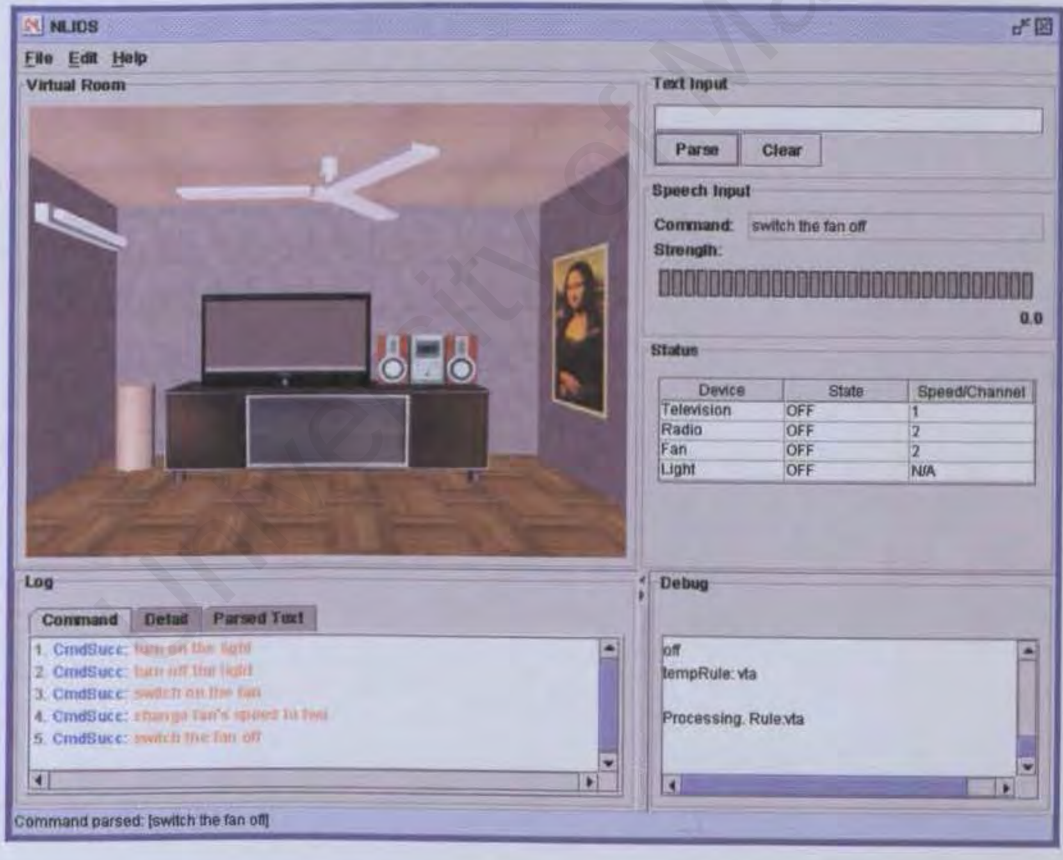
```
1. CmdSucc: turn on the light
2. CmdSucc: turn off the light
3. CmdSucc: switch on the fan
4. CmdSucc: change fan's speed to two
```

```
two
tempRule: vtaep

Processing. Rule:vtaep
```

Command parsed: [change fan's speed to two]

3.	**Scenario:**	Instruct proposed system to switch off the fan
	Use case:	Operate fan
	Instruction:	Switch the fan off
	Input & Response:	• Input field displaying the instruction. • Formatted instruction is display at command tab • Notification synthesized to user.
	Language Processing:	• Parsed input is shows at parsed text tab. • Status table is updated.
	Visualization:	• Fan is switched off.



Command: switch the fan off
Strength:
0.0

Status

Device	State	Speed/Channel
Television	OFF	1
Radio	OFF	2
Fan	OFF	2
Light	OFF	N/A

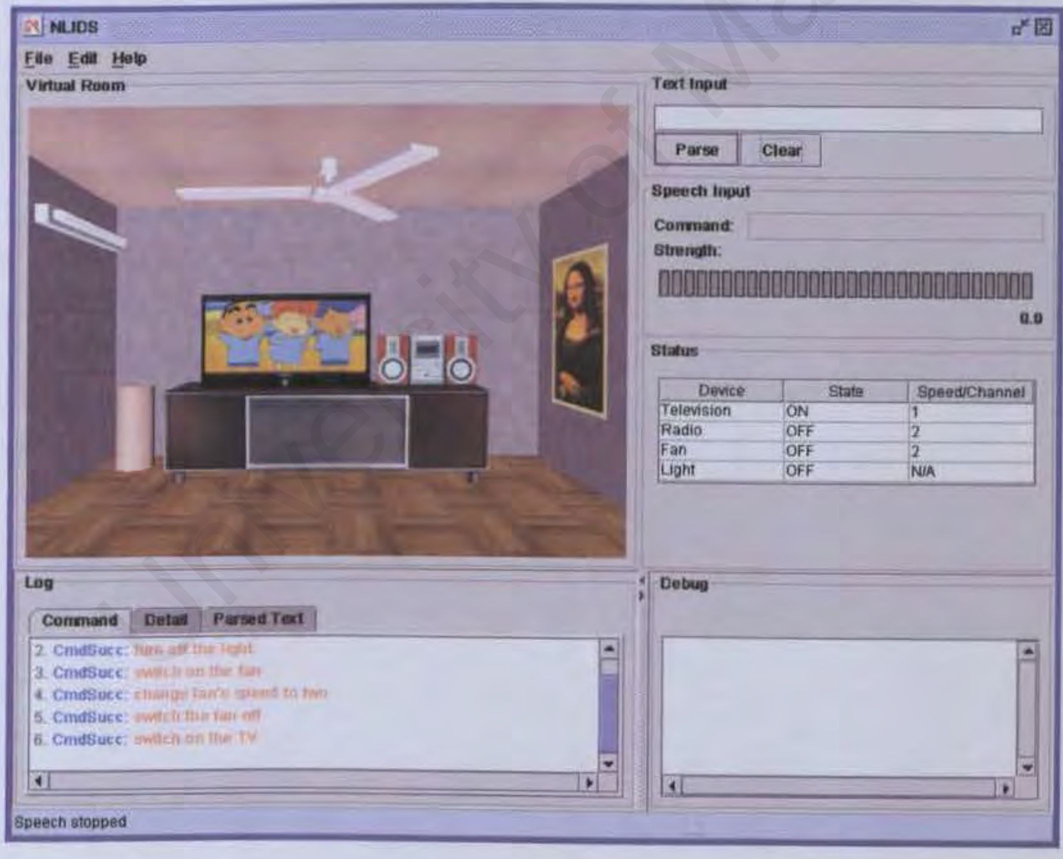
```
1. CmdSucc: turn on the light
2. CmdSucc: turn off the light
3. CmdSucc: switch on the fan
4. CmdSucc: change fan's speed to two
5. CmdSucc: switch the fan off
```

```
off
tempRule: vta

Processing. Rule:vta
```

Command parsed: [switch the fan off]

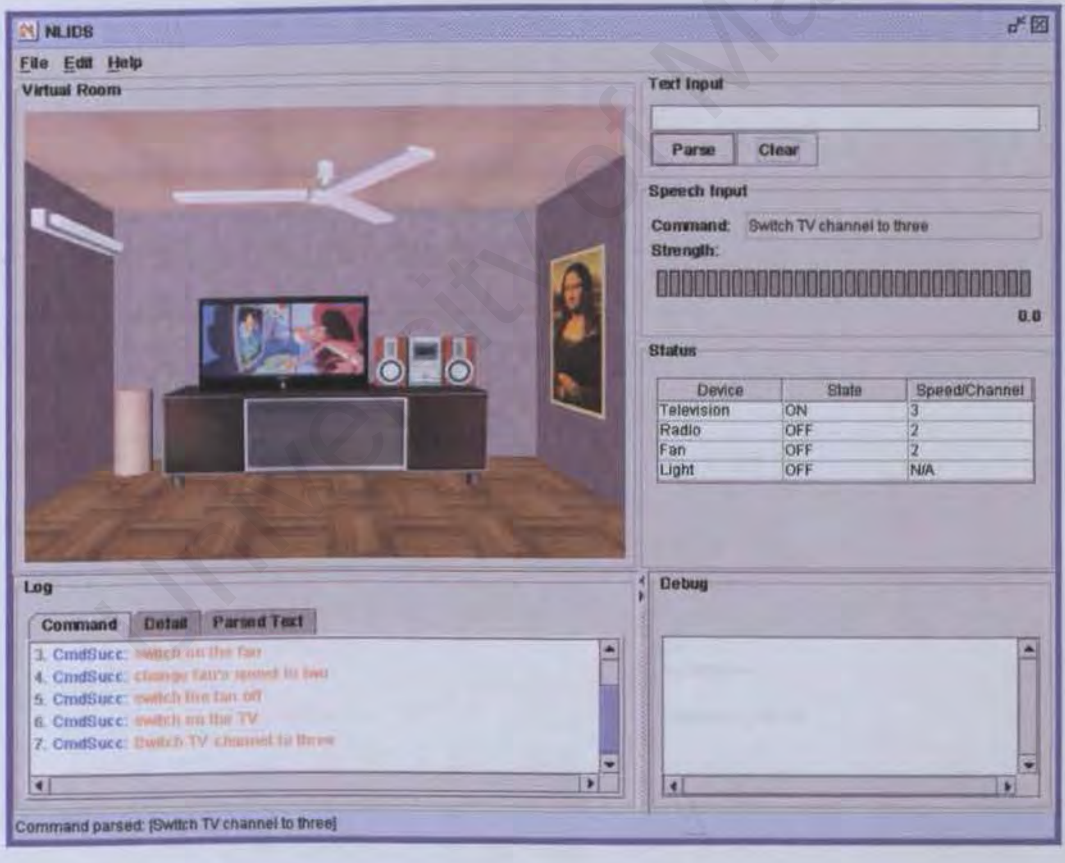
Table 7.2c: Result of Functional Testing for Operate TV Use Case

1.	**Scenario:**	Instruct proposed system to switch on the TV
	Use case:	Operate TV
	Instruction:	Switch on the TV
	Input & Response:	• Input field displaying the instruction. • Formatted instruction is display at command tab • Notification synthesized to user.
	Language Processing:	• Parsed input is shows at parsed text tab. • Status table is updated.
	Visualization:	• TV is switched on. 

Device	State	Speed/Channel
Television	ON	1
Radio	OFF	2
Fan	OFF	2
Light	OFF	N/A

```
2. CmdSucc: turn off the light
3. CmdSucc: switch on the fan
4. CmdSucc: change fan's speed to two
5. CmdSucc: switch the fan off
6. CmdSucc: switch on the TV
```

Speech stopped

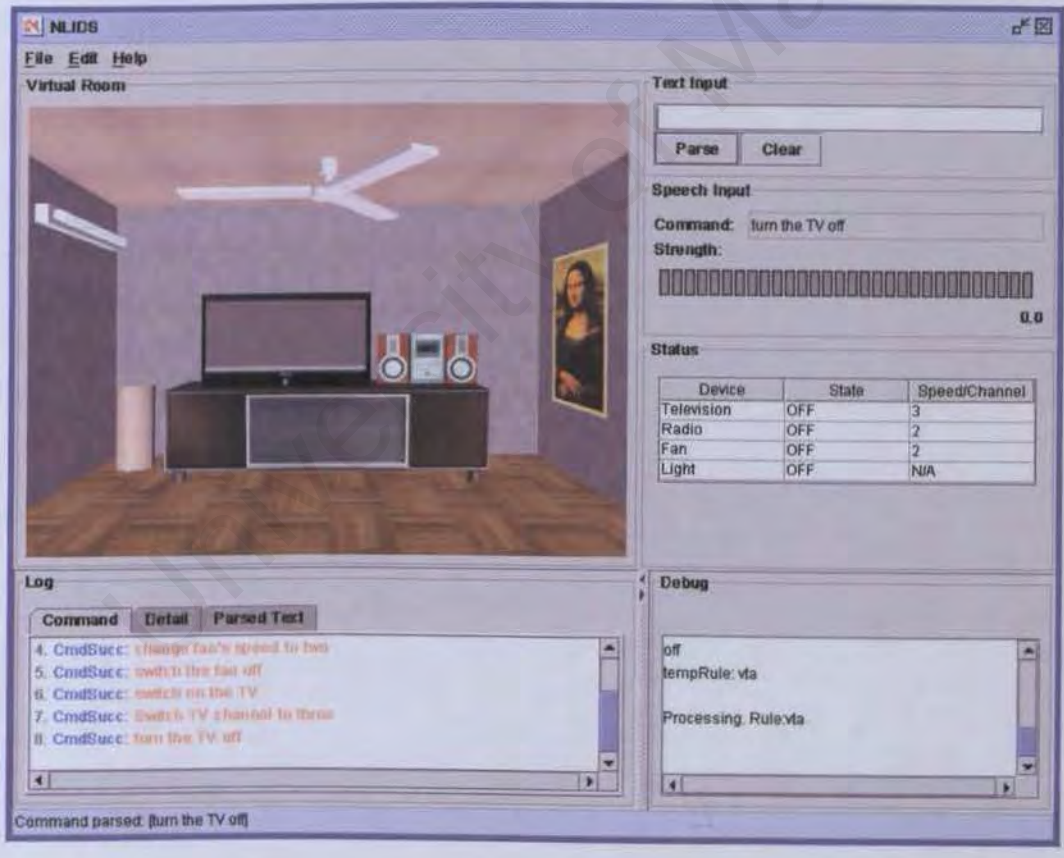
2.	Scenario:	Instruct proposed system to switch TV's channel to three
	Use case:	Operate TV
	Instruction:	Switch TV's channel to three
	Input & Response:	• Input field displaying the instruction. • Formatted instruction is display at command tab • Notification synthesized to user.
	Language Processing:	• Parsed input is shows at parsed text tab. • Status table is updated.
	Visualization:	• TV's channel is switched to three. 

Command: Switch TV channel to three

Device	State	Speed/Channel
Television	ON	3
Radio	OFF	2
Fan	OFF	2
Light	OFF	N/A

```
3. CmdSucc: switch on the fan
4. CmdSucc: change fan's speed to two
5. CmdSucc: switch the fan off
6. CmdSucc: switch on the TV
7. CmdSucc: Switch TV channel to three
```

Command parsed: [Switch TV channel to three]

3.	**Scenario:**	Instruct proposed system to turn off the TV
	Use case:	Operate TV
	Instruction:	Turn the TV off
	Input & Response:	• Input field displaying the instruction. • Formatted instruction is display at command tab • Notification synthesized to user.
	Language Processing:	• Parsed input is shows at parsed text tab. • Status table is updated.
	Visualization:	• TV is turned off. 

Command: turn the TV off
Strength: 0.0

Device	State	Speed/Channel
Television	OFF	3
Radio	OFF	2
Fan	OFF	2
Light	OFF	N/A

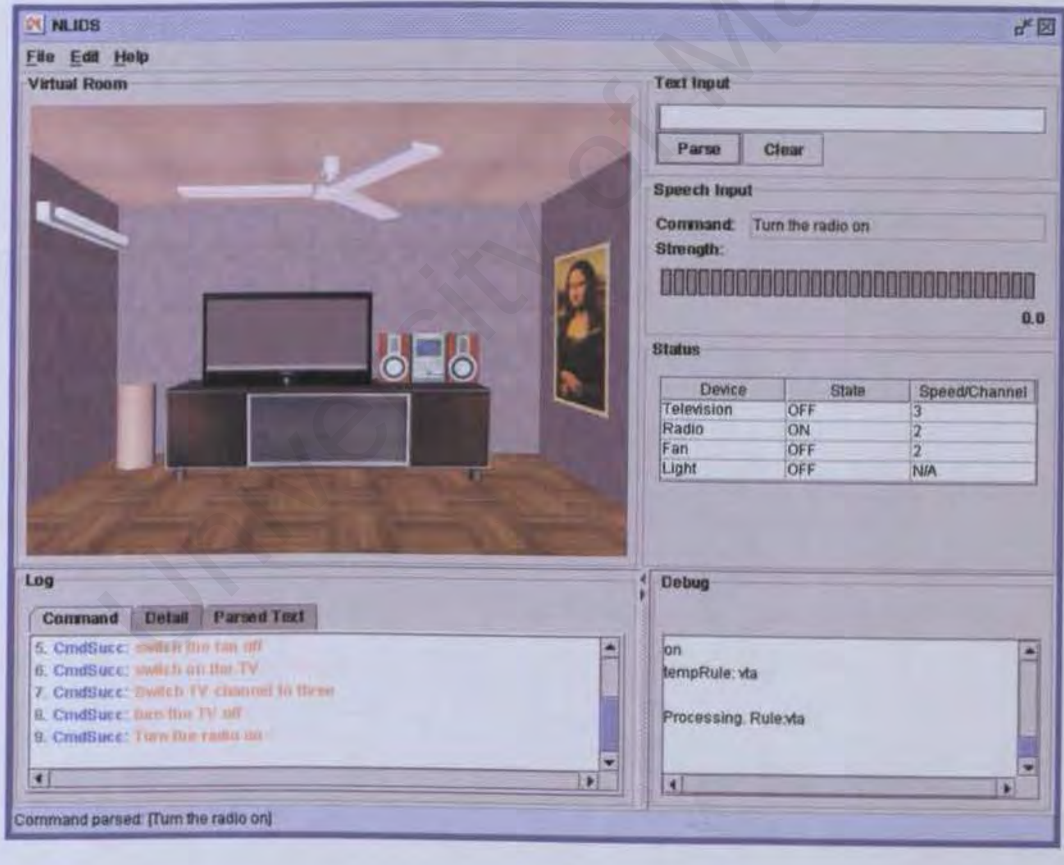
```
4. CmdSucc: change fan's speed to two
5. CmdSucc: switch the fan off
6. CmdSucc: switch on the TV
7. CmdSucc: Switch TV channel to three
8. CmdSucc: turn the TV off
```

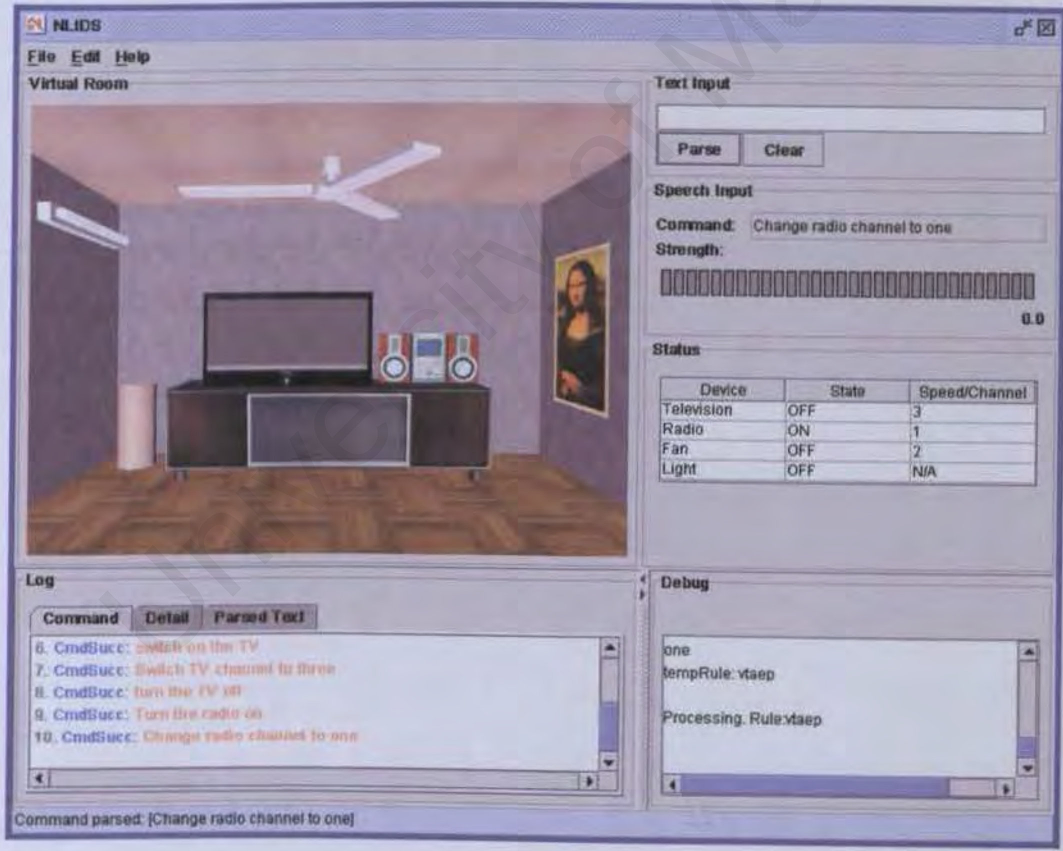
```
off
tempRule: vta

Processing. Rule:vta
```

Command parsed: [turn the TV off]

Table 7.2d: Result of Functional Testing for Operate Radio Use Case

1.		
	Scenario:	Instruct proposed system to turn on the radio
	Use case:	Operate Radio
	Instruction:	Turn the radio on
	Input & Response:	• Input field displaying the instruction. • Formatted instruction is display at command tab • Notification synthesized to user.
	Language Processing:	• Parsed input is shows at parsed text tab. • Status table is updated.
	Visualization:	• Radio is turned on.  NLIDS Command: Turn the radio on Strength: 0.0 Device / State / Speed/Channel Television / OFF / 3 Radio / ON / 2 Fan / OFF / 2 Light / OFF / N/A 5. CmdSucc: switch the fan off 6. CmdSucc: switch on the TV 7. CmdSucc: Switch TV channel to three 8. CmdSucc: turn the TV off 9. CmdSucc: Turn the radio on on tempRule: vta Processing. Rule:vta Command parsed: [Turn the radio on]

2.		
	Scenario:	Instruct proposed system to change radio's channel to one
	Use case:	Operate Radio
	Instruction:	Change radio's channel to one
	Input & Response:	• Input field displaying the instruction. • Formatted instruction is display at command tab • Notification synthesized to user.
	Language Processing:	• Parsed input is shows at parsed text tab. • Status table is updated.
	Visualization:	• Radio's channel is changed to one. 

Command: Change radio channel to one

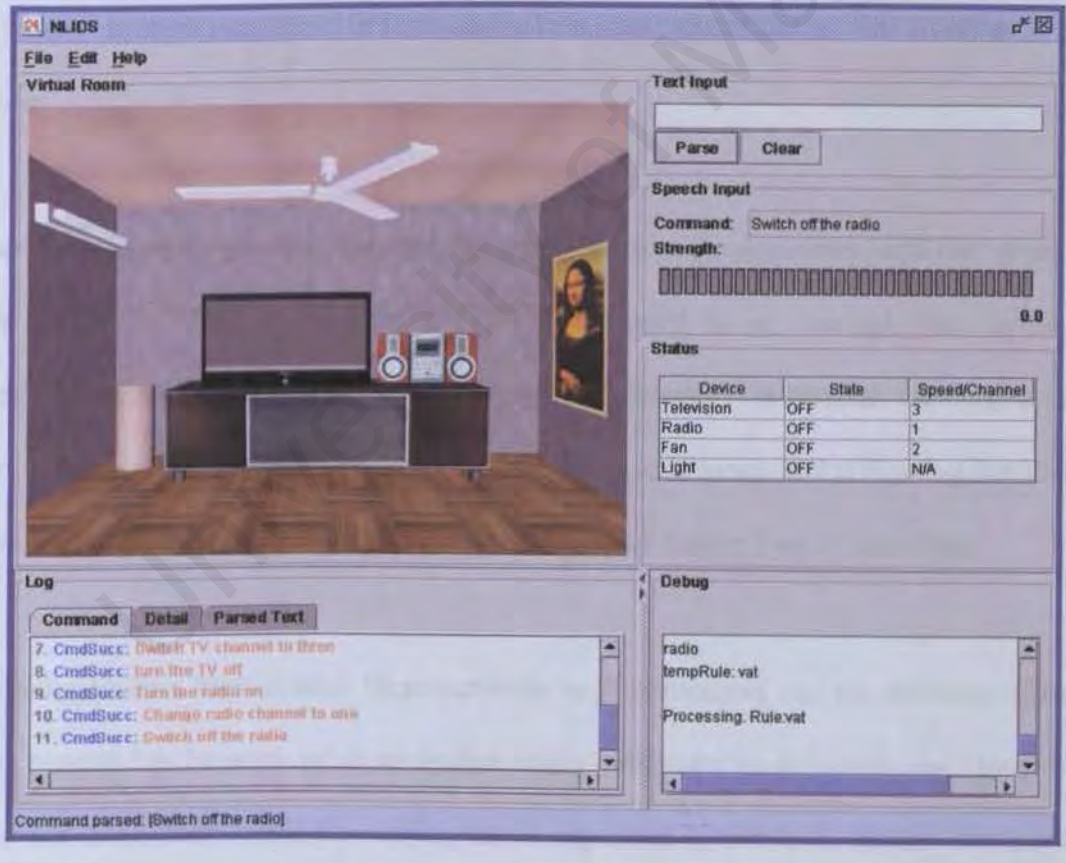
Device	State	Speed/Channel
Television	OFF	3
Radio	ON	1
Fan	OFF	2
Light	OFF	N/A

```
6. CmdSucc: switch on the TV
7. CmdSucc: Switch TV channel to three
8. CmdSucc: turn the TV off
9. CmdSucc: Turn the radio on
10. CmdSucc: Change radio channel to one
```

```
one
tempRule: vtaep

Processing. Rule:vtaep
```

Command parsed: [Change radio channel to one]

3.	Scenario:	Instruct proposed system to switch off the radio
	Use case:	Operate Radio
	Instruction:	Switch off the radio
	Input & Response:	• Input field displaying the instruction. • Formatted instruction is display at command tab • Notification synthesized to user.
	Language Processing:	• Parsed input is shows at parsed text tab. • Status table is updated.
	Visualization:	• Radio is switched off.  NLIDS Speech Input — Command: Switch off the radio Status Device | State | Speed/Channel Television | OFF | 3 Radio | OFF | 1 Fan | OFF | 2 Light | OFF | N/A Log (Command) 7. CmdSucc: Switch TV channel to three 8. CmdSucc: turn the TV off 9. CmdSucc: Turn the radio on 10. CmdSucc: Change radio channel to one 11. CmdSucc: Switch off the radio Debug radio tempRule: vat Processing. Rule:vat Command parsed: [Switch off the radio]

7.3 System Evaluation

The purpose of system evaluation is to evaluate the accuracy of the developed system with response to user's instruction and gather feedback on the user acceptance of the speech interface. For achieve these purposes, two sessions namely accuracy evaluation session and feedback gathering session have been carried out.

7.3.1 Evaluation Session Setup

Eight users whom are more than 65 years old are invited as the participant for both sessions. The participant has been told that the action triggered for each valid instruction given is only through visualization form on the computer screen but no real world activity involved.

In order to increase the level of comfortable and to acquire the necessary skill for "speak" speech instruction, each participant has been requested to go through the "Speech Recognition Voice Training" program developed by Microsoft. In general, the program is used to improve the computer's ability to understand user's voice and it is being installed automatically when the speech engine mentioned in the Chapter Two is installing.

The training process took around 10 minutes for each participant and the outcome of the training process has been saved as an unique voice profile for each participant. Once the training has been completed, the participant is requested to continue on the evaluation sessions described at the following sections.

7.3.2 Accuracy Evaluation Session

The main objective of this session is to evaluate the accuracy of the proposed system with the response to speech instruction. The accuracy evaluation session was carried out in two rounds. In each round, every participant was given the same set of test input that used in the unit & module testing. Generally, the test input consists of 20 instructions. In other words, there is total 160 instructions have been tested in each round. The participants were asked to read each instruction until the correct response was generated by NLIDS or until each instruction has been repeat read up to six times. The number of reading for each instruction was then being taken.

A test case is considered fail if NLIDS does not generate respective response after an instruction has been read up to six times. On the other hand, a test case is considered passed if an instruction is recognized by NLIDS after it has been read six times or less. The total number of recognized instruction was used to calculate the recognition rate of NLIDS.

As mentioned earlier, the accuracy monitoring session was conducted in two rounds. In the first round, the default voice profile for the speech recognition engine employed is being used. While in the second round, the own voice profile that saved during training session that mentioned in the evaluation setup session is being used. The results of the evaluation are summarized in Figure 7.7.

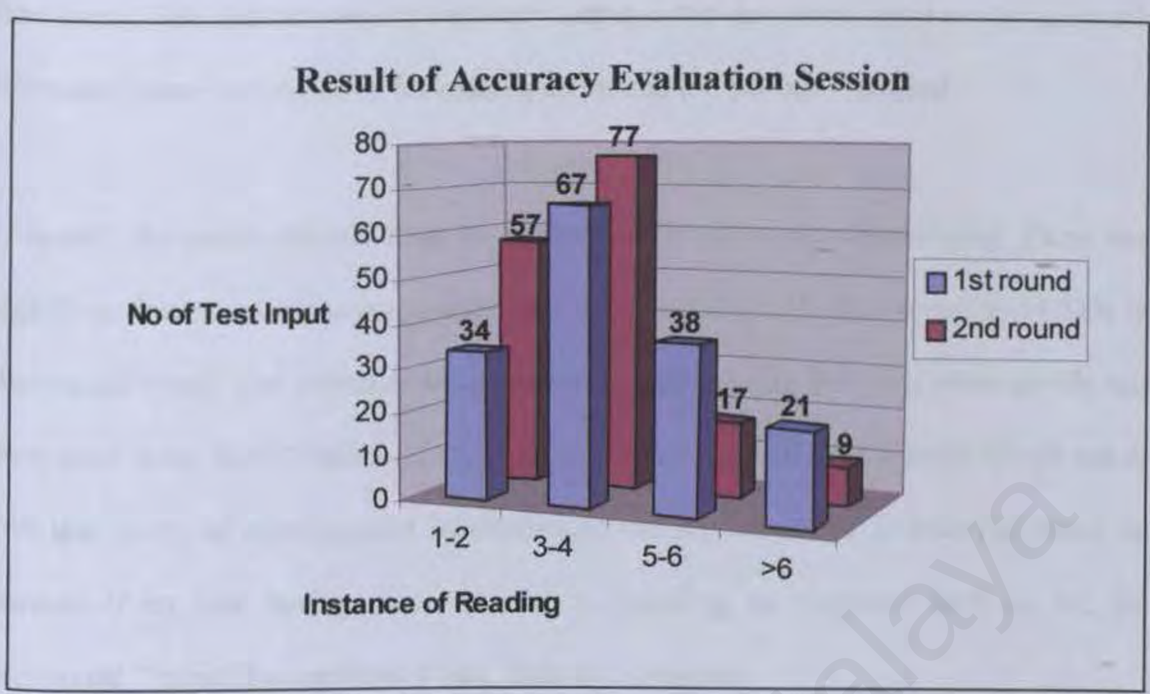


Figure 7.7: Result of Accuracy Evaluation Session

Figure 7.1 reveals that the proposed system recognized an instruction mostly when an instruction was read repeated three to four times. In the first round, the instance of reading accounted is about 42% (67 out of 160 test cases) while in the second round, the instances of reading was slightly increased to almost half of all counted instances. Besides that, the percentage for instruction were read one to two times is increased 10.6% to 35.6% (57 out of 160 test cases) in the second round compare to the corresponding percentage in the first round.

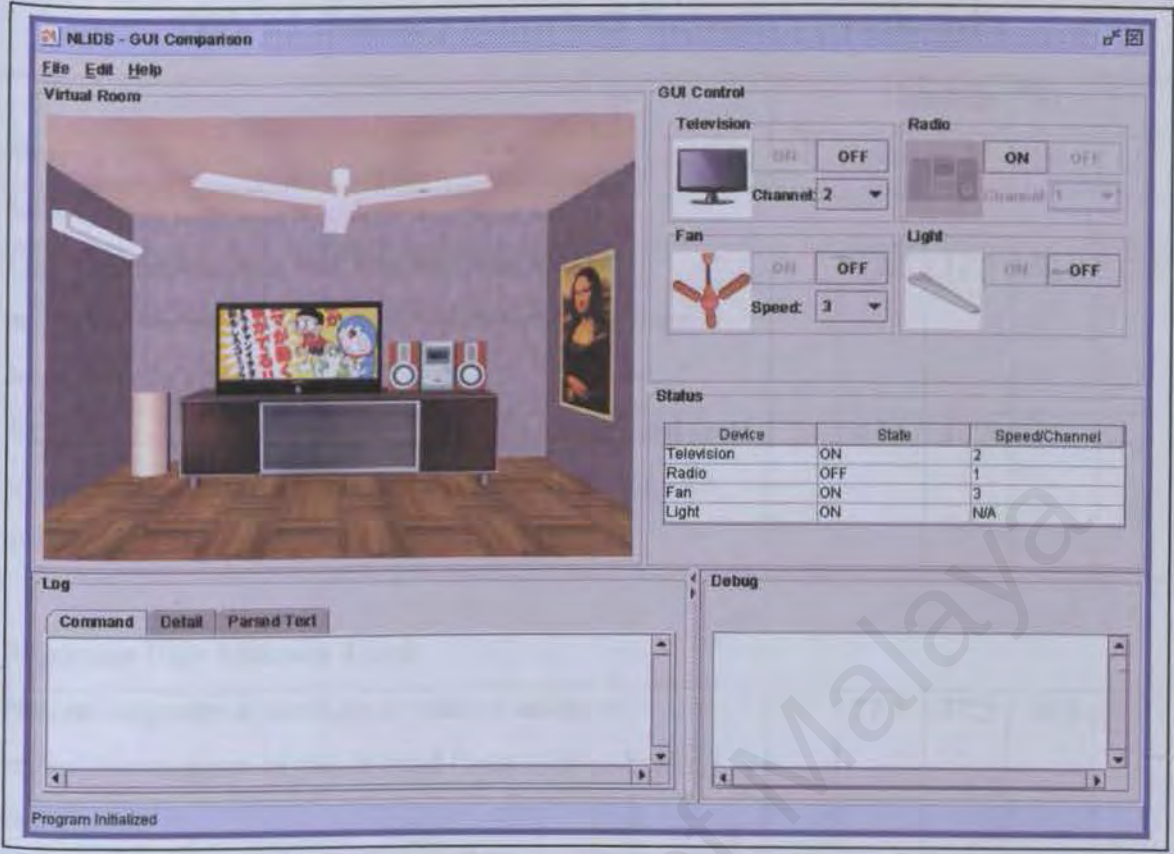
From the figure, we could see the dramatic increment from 63.13% in the first round to 83.75% in the second round for the total percentage of instructions were read less than four times. This situation would definitely increase the user satisfaction on using speech instruction since the repetition for reading an instruction has been reduced.

Generally, the result obtained from the evaluation session seems encouraging. There was significant increment in the recognition rate from 86.88% in the first round to 94.38% in the second round. The reason of such increment might due to that own voice profile has been used in the second round. In addition, the negative result, which is the 6% (9 out of 160 test cases) of unrecognized instruction in the second round, is believed could be reduced if an user having more training in speaking to computer such as via the mentioned "Speech Recognition Voice Training" program.

7.3.3 Feedback Gathering Session

The purpose of this session is to gather the user's experience and opinion for accessing speech instruction in smart home system.

In this session, a modified version of the proposed system is provided to the participant. As shown in the Figure 7.8, the speech instruction interface has been replaced with popular graphical user interface (GUI) components such as graphical button and drop down list. The participant was then be requested to perform the necessary action such as clicking the graphical button and select particular state in a drop down list to achieve the same outcome for each test cases used in the earlier session. The purpose of accessing such GUI component is to allow the participant to experience the differences between speech based inputs versus GUI component in instructing smart home's appliance.



Device	State	Speed/Channel
Television	ON	2
Radio	OFF	1
Fan	ON	3
Light	ON	N/A

Figure 7.8: NLIDS in GUI Comparison Mode

A questionnaire has been provided to participant once they have completed all the test case in GUI's method. The questionnaire is shows in the Appendix C of this report. Table 7.3 shows the average of rating (percentage) for eight participants in each question. The rating used in the table are as follows:

- 1 – Worst / Strongly Disagree
- 2. – Poor / Disagree
- 3 – Average / Neutral
- 4 – Good / Agree
- 5 – Excellent / Strongly Agree

Table 7.3: Summary of Feedback Gathering's Questionnaire

Description	Ratings (%)				
	1	2	3	4	5
Literacy					
Only **minimum** computer literacy is required for accessing Smart Home System via Natural Language based user interface.		12.5	12.5	50.0	25.0
Experience in using Smart Home System is required prior to use of Natural Language based user interface in such system		87.5	12.5		
Minimize User Memory Load					
Natural language instruction is relative easier to remember compare to pre-defined (mouse click/button) instruction.		12.5	37.5	50.0	
Usability					
Natural language instruction is relative easier to use compare to pre-defined (mouse click/button) instruction.		12.5	37.5	50.0	
Training					
Natural language instruction is relative easier to learn compare to pre-defined (mouse click/button) instruction.			25.0	62.5	12.5
Response					
Response is better in synthesized form (voice) rather than displayed (text) at screen.		50.0	50.0		
Integrity/Accuracy					
The given instruction is always understood and		50.0	25.0	25.0	

performed correctly by NLIDS					
Overall					
Natural language instruction is believed will help elderly in accessing smart Home System.		12.5	25.0	50.0	12.5
Natural language instruction is efficient as predefined instruction.		12.5	50.0	37.5	
Natural language instruction is **MORE** efficient as predefined instruction.		50.0	50.0		
Natural language instruction should be widely use in other instruction driven systems such as driving a car.			50.0	50.0	

As in table 7.3, the feedback are gathered according to the seven categories namely Literacy, Minimized User Memory Load, Usability, Training, Response, Integrity/Accuracy, and Overall. For the purpose of analyzing feedback gathered, the feedback gathered has been grouped into two categories which are non-agreeable (rating 1-2) feedback and agreeable (rating 3-5) feedback.

Based on the table, 7 out of 8 (87.5%) participants agreed that only minimum computer literacy is required for using natural language based user interface. Additionally, there was same number of participant (7 out of 8) feedback that experience in accessing Smart Home System is not required for using the speech user interface. The result seems encouraging as it suggests that the technology is possible to reach more users including those with limited computer literacy and not prior experience in using similar system.

Besides, most of the participants (7 out of 8 or 87.5%) responded that the natural language based user interface is easy or easier to use compare to existing GUI. Again, there was same number of participants (87.5%) responded that the speech based user interface is easier to remember than GUI. In addition, all of the participants (100%) believed that the learning of using the proposed user interface should be easy. The results suggest that the technology could be adopted easily by elderly. Elderly people often have difficulties interacting through unfamiliar means, such as keyboards and computer screens Roy et al. (2000).

However, despite the encouraging result obtained as stated above, there were half of participants (50%) are still lacking of confident in the integrity of outcome of natural language instruction. They were worry that the system might not understand completely about the given instruction and therefore produce unexpected outcome. The concern arise might due to the high unrecognition rate in the first round of accuracy evaluation session mentioned earlier. Anyway, advance in technology particular in speech recognizer related field should increase the recognition rate into an acceptable level.

In general, based on the feedback gathered, 7 out of 8 participants (87.5%) believed that the natural language based user interface would easy the elderly in accessing Smart Home. Same number of participant agreed that the technology is as efficient as GUI while the entire participant (100%) wished the technology could also be implemented into domain other than Smart Home.

7.4 Conclusion

Unit testing on the individual components, module testing on the each module, and integration testing that test the proposed system in whole were carried out sequentially. The problems encountered during each of the phase of testing have been rectified accordingly. Besides that, the evaluation carried out has pointed out that the participants appreciate on implementing natural language based user interface in smart home system and they believed such input method could be as efficient as common input method such as GUI.

Chapter 8.0 Conclusion and Future Works

8.1 Introduction

This chapter outlines the system constraints and bottlenecks, further work that can be carried out in the future, and conclusion of the NLIDS project.

8.2 Achievement

The objectives of this project were successfully accomplished, which includes:

- **Developed prototype of Speech Based Instruction Smart Home System**

  A prototype has been developed throughout this project. The prototype successfully demonstrates the possibility of using speech based user interface in Smart Home system. The analysis and design of the developed prototype was discussed in Chapter four and Chapter five.

- **Integrated State of the Art Natural Language Processing Technologies**

  Several state of the art tools and libraries such LinkGrammar library, Microsoft Speech SDK, Java Speech API and Java 3D API were successfully been integrated together in developing the proposed system. The integration and implementation of the tools was documented in Chapter Six.

- **Adopted Proven Methodology in Development of Proposed System**

Unified process was adopted in this project. The four phases, namely Inception phase, Elaboration phase, Construction phase, and Transition phase in the methodology have been studied and followed strictly during developing of the proposed system. The adoption of the methodology was discussed in Chapter three.

- **Tested and Evaluated Proposed System**

  Several testing and evaluation processes were performed to verify the proposed system has fulfilled the requirement specification. In addition, the feedback on the proposed system shows that speech recognition interface is value added functionality and relatively easier to use. The results of the testing and evaluation carried out were discussed in Chapter Seven.

8.3 System Constraints and Bottlenecks

This section is to identify the constraints and bottlenecks of the NLIDS system. By doing this, we managed to outline the future works to overcome the problems and also suggest the possible enhancement to the existing implementation. Among the identified constraints and bottlenecks of the NLIDS are stated below:

8.3.1 Language Constraints

As mentioned in the earlier chapter, the project used LinkGrammar, a parser that used for syntactic parsing. Since LinkGrammar is an English parser, our system can only 'understands' command in English language. In addition, the defined Java Speech Grammar Rules in English language is also limited the system to only accept English language's speech.

Besides that, the constructed command needs to comply with English grammar's rule. In other words, the constructed command must be a well defined and proper English sentence.

8.3.2 Processing Bottleneck

Generally, NLIDS is designed to process single type of LinkGrammar's constituent trees. Although this is sufficient for most of the user instructions, there are cases whereby the lexicon produced based on the single type of constituent tree is inaccurate and leads to undesirable systems output.

8.3.3 Parser Capability

The LinkGrammar used is a beta release (non-stable) version. It is a Java version that translated from the original C Language version. Hence, the produced parsed output might be different from the original C Language version of LinkGrammar. Moreover, the parsed output might be less detail and inaccurate since the porting of C Language version is yet to be completed.

8.3.4 Platform Dependency

Although there is no intention to limit the running platform for NLIDS, the system is still limited to the Microsoft Window. The limitation is due to the speech engine employed in the NLIDS only run on the Windows platform. Ideally, the system shall run on multiple platforms so that the system is easy to be customized and ported to other platform.

8.4 Future Works

In this project, we have shown that natural language instructions can be applied to Smart Home System. However, a few aspects were identified that has room for further improvements and enhancement as suggested at the following sections.

8.4.1 Parser Accuracy and Performance

Accuracy and performance in processing speed are always the key concerns when choosing a natural language parser. Unfortunately, the more accurate a parser is, the less performance in speed and vice versa. Therefore, a parser which well balances in all key aspect is preferable so that the time taken and quality of a response from a Natural Language Processing system is acceptable.

In order to increase the accuracy and performance of NLIDS, the existing beta version parser is suggested to be replaced with a production version parser. We would suggest to use an open source and Java based parser as replacing parser. The effort to be spent in the integration of new parser into the NLIDS could be minimized if the replacing parser is implemented in Java. In addition, an open source parse is relatively easier to be customized compare to the commercial parser. The Stanford University's Parser which is still yet to be released at the time the proposed system is developing could be a good option as the new parser since it is totally written in Java Language and platform independent.

8.4.2 Variety in Natural Language

The limitation of using English Language as the sole type of input language should be overcome in the future. There are certain aspects need to be focused for overcoming the limitation. Fist of all, we would need a new parser that support other type of languages. For instance, the Stanford Parser mentioned earlier is an option again. The parser is capable and has been adapted to work with other languages such as Chinese, German, and France.

Beside the parser, the speech synthesizer and recognition engine employed must able to support the desired language. In addition, the engine must be Speech Application Programming Interface version 4 (SAPI4) or SAPI5 compliant to fulfill the JSAPI requirement.

8.4.3 Cross Platform Support

The user experience is believed will be increased if NLIDS is able to run on multiple platforms. For instance, NLIDS can be installed on a mobile phone that running Linux operating system and the command to activate an electronic device can be send via the mobile phone rather than a personal computer.

The two key factors: language parser and the speech synthesizer/recognition engine need to be replaced in the effort to support multiple platforms. As mentioned in the earlier section, the platform limitation could be overcome if NLIDS employs a Java based and open source parser, such as The Stanford Parser. By using an open source parser, we are able to compile the new NLIDS together with the open source parser on multiple platforms. In addition to the parser, the speech synthesizer and recognition engine need to be switched to an open source SAPI4 and SAPI5 compliant speech engine such as Sphinx.

8.5 Conclusion

The development of the NLIDS involved many stages and processes. The problem statement and objective were firstly defined and set. After that, the NLIDS system was developed which consisted of analysis, design, implementation and testing and evaluation processes.

Many problems and challenges have been faced during the development process of the NLIDS. Nevertheless, more knowledge and experiences than expected have been gained throughout the research. These included learning to know the format of parsed output from natural language parser, to perceive the steps and phases of natural language processing, and to understand the grammar enforcing by speech API.

Lastly, we do hope that the NLIDS may serves as a stepping stone for future natural language driven system and a source of knowledge for the development for more advanced projects.

Appendix A Sequence Diagram

A1.0 Toggle Light

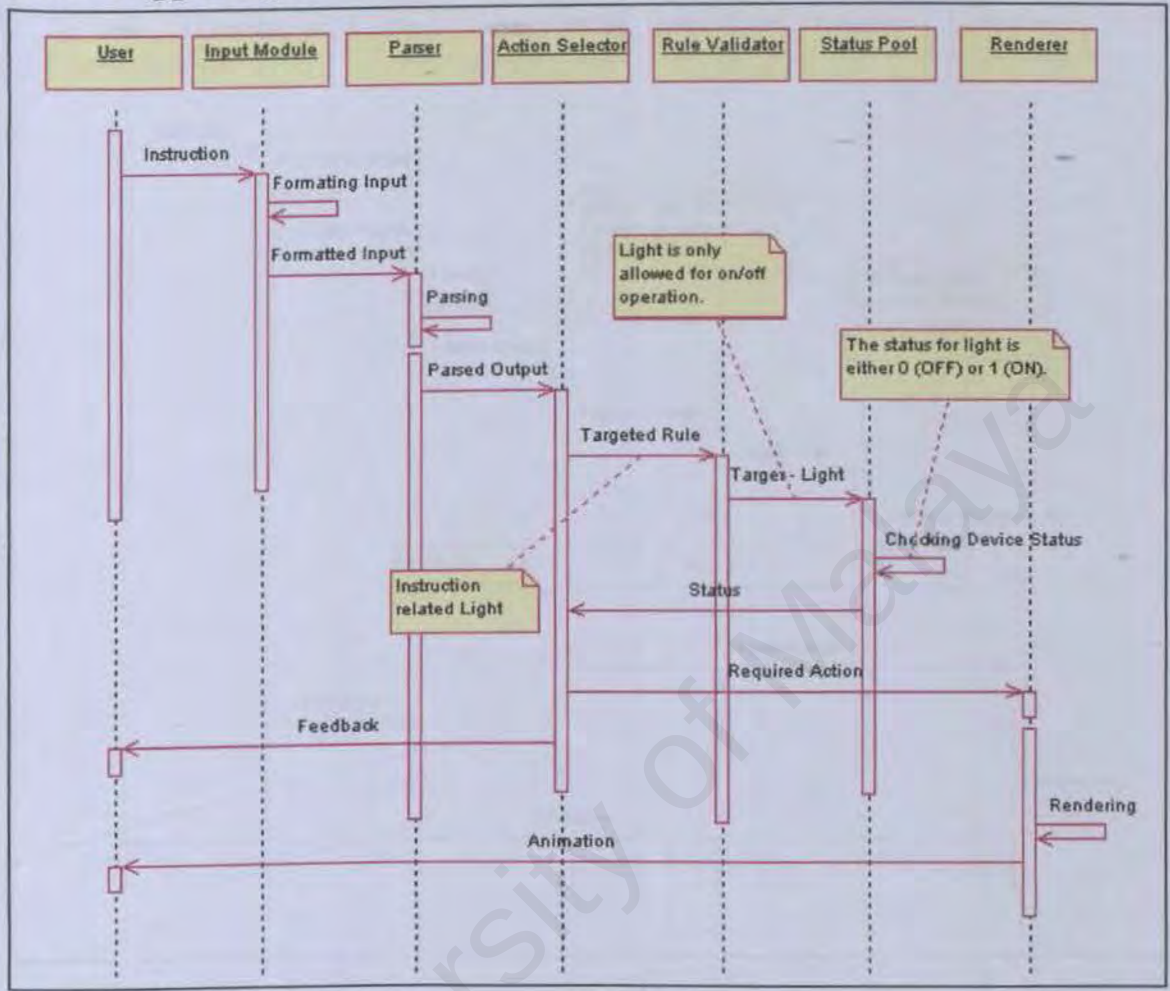


Figure A1.0: Toggle Light Sequence Diagram

User's instruction is passed to the system via Input Module. The input is in speech form. The input will be formatted before forward to the Parser for parsing. An output which refers as parsed output will be produced by the parser at the end of parsing stage. The parsed output is used as a key to determine the necessary rule and status that required for validating and checking. For Figure A1.0, the target is Light so the allowed action is toggle the light to ON or OFF state only. Once the Action selector receives the status from the Status Pool module, it will provide the feedback to user and so on trigger Render Module to generate respective animation.

A2.0 Operate Fan

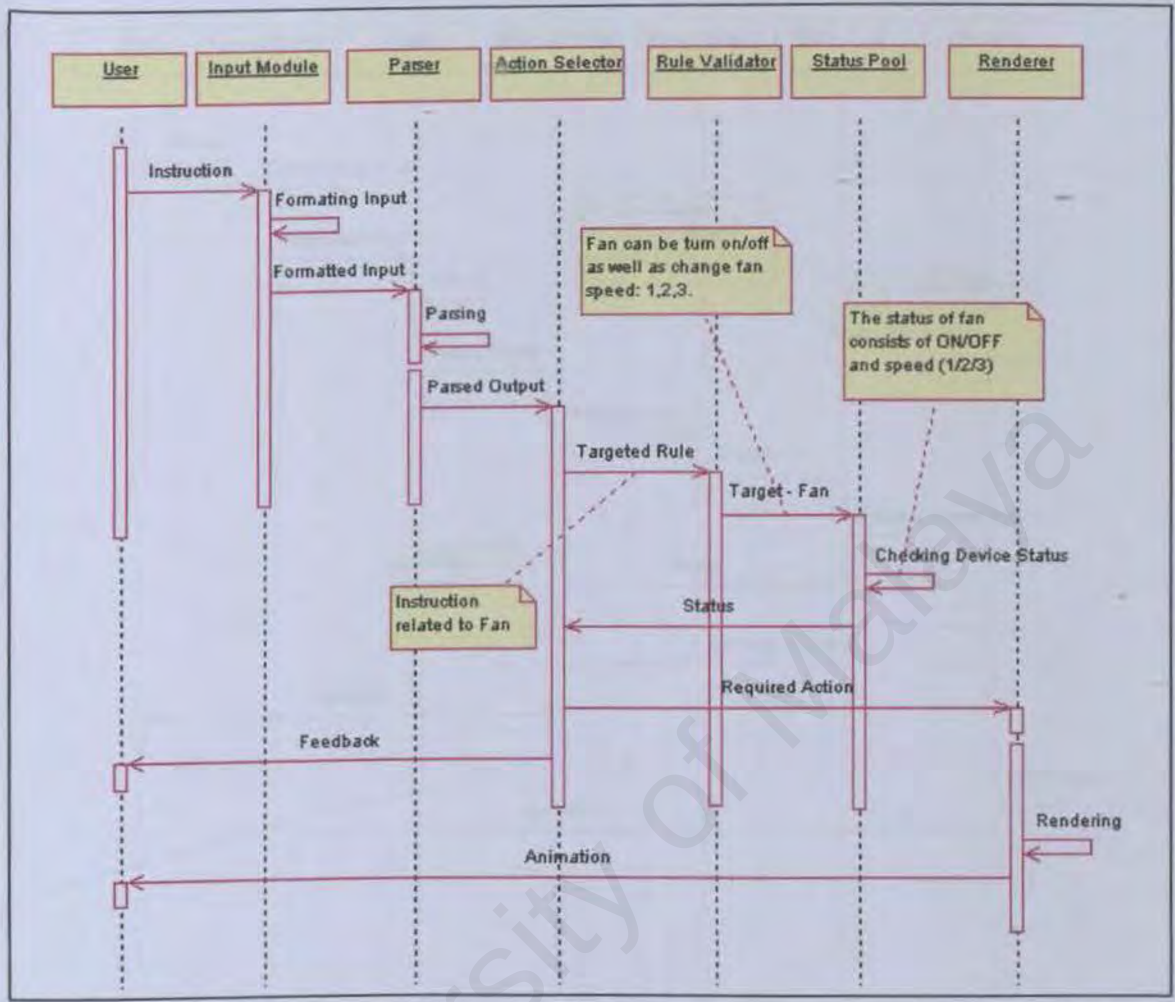


Figure A2.0: Operate Fan Sequence Diagram

User's instruction is passed to the system via Input Module and it will be formatted before forward to the Parser for parsing. An output which refers as parsed output will be produced by the parser at the end of the parsing stage. The parsed output is used as a key to determine the necessary rule and status that required for validating and checking. For target as Fan, it is allowed for switching ON and OFF as well as changing fan's speed. Once the Action selector receives the status from the Status Pool module, it will provide the feedback to user as well as trigger the Render Module to generate respective animation.

A3.0 Operate TV

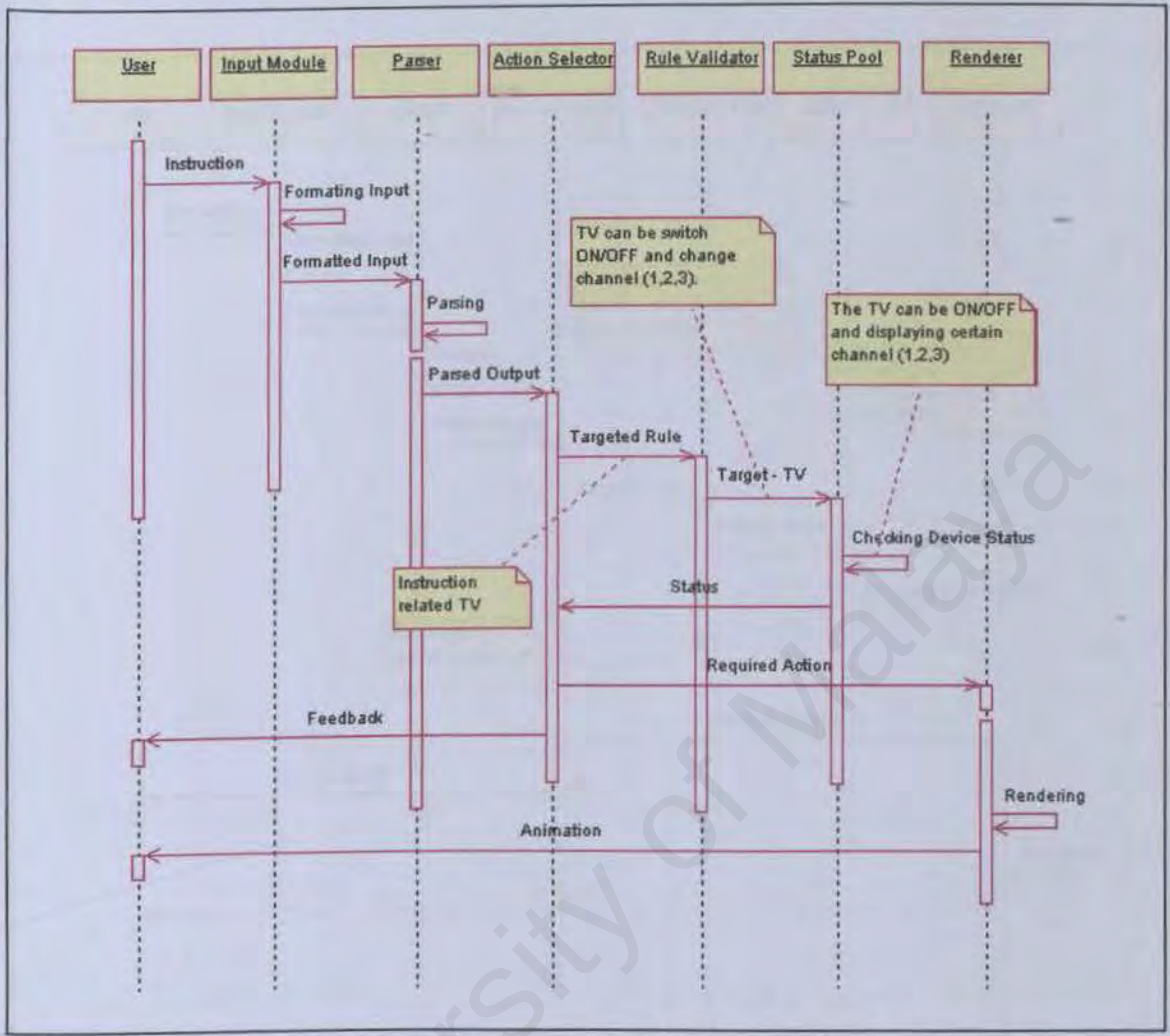


Figure A3.0: Operate TV Sequence Diagram

User's instruction is passed to the system via Input Module and it will be formatted before forward to the Parser for parsing. An output which refers as parsed output will be produced by the parser at the end of the parsing stage. The parsed output is used as a key to determine the necessary rule and status that required for validating and checking. For target as TV, it is allowed for turning ON and OFF as well as switching channel. Once the Action selector receives the status from the Status Pool module, it will provide the feedback to user as well as trigger the Render Module to generate respective animation.

A4.0 Operate Radio

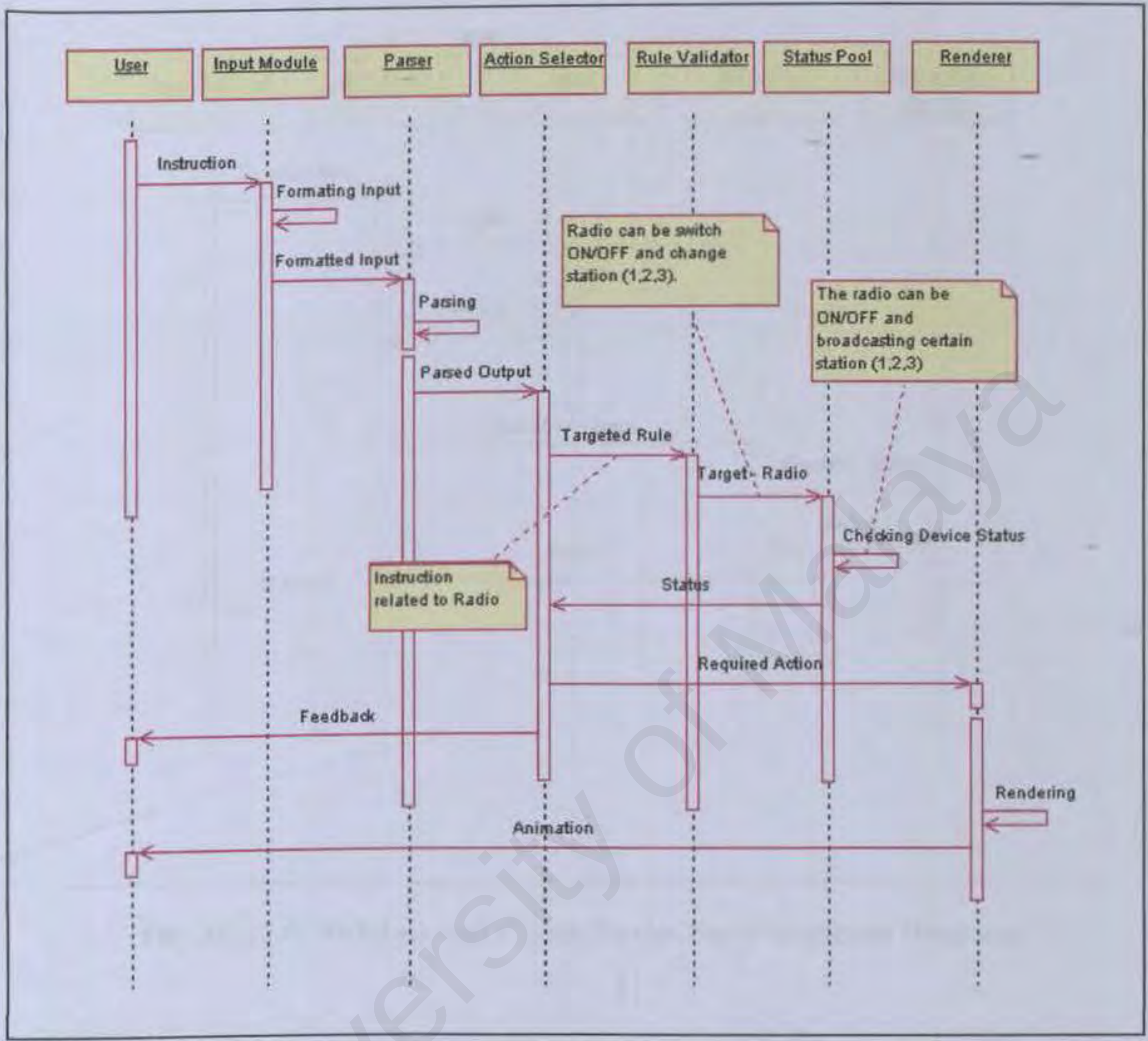


Figure A4.0: Operate Radio Sequence Diagram

User's instruction is passed to the system via Input Module and it will be formatted before forward to the Parser for parsing. An output which refers as parsed output will be produced by the parser at the end of the parsing stage. The parsed output is used as a key to determine the necessary rule and status that required for validating and checking. For target as Radio, it is allowed for turning ON and OFF as well as switching station. Once the Action selector receives the status from the Status Pool module, it will provide the feedback to user as well as trigger the Render Module to generate respective animation.

A5.0 Validate and Check Device State

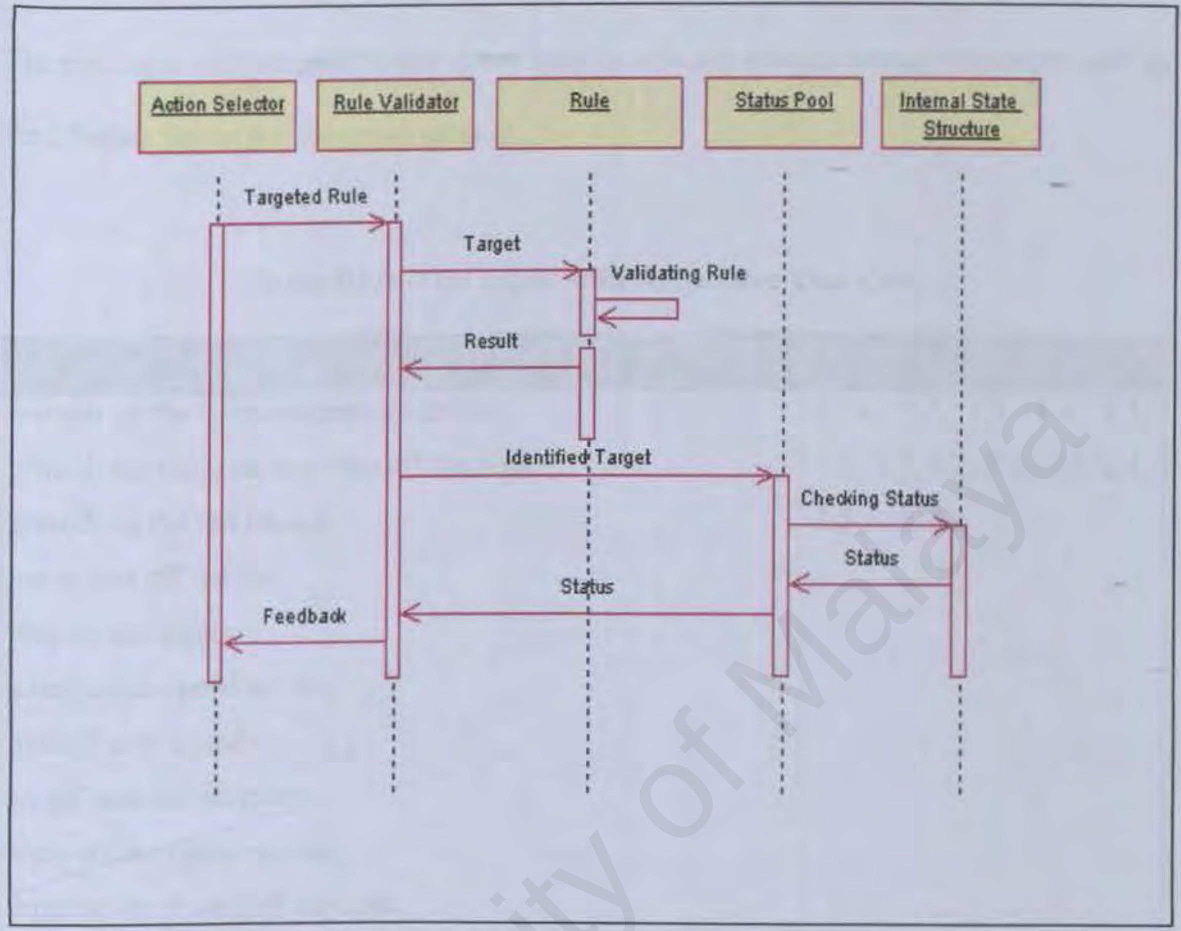


Figure A5.0: Validate and Check Device State Sequence Diagram

The parsed output that feed from the parsing stage will be used to validate device's rule as well as device's state according to the instruction's target. The rule validating engine will performing checking on device status by selecting the corresponding status from the status pool after validated the rules. If the validation of instruction is passed, the status will be feedback to the action selector module and so on returned to the earlier module.

Appendix B List of Test Data

The test input with respective test cases used in unit and module testing per mentioned in the Chapter Seven are shown in table B1.0.

Table B1.0: Test Input with Respective Test Case

Test Input	Test Cases
switch on the light and turn on radio switch the radio on and turn off the light switch on the fan please on tv and off the fan fan on and light on change fan speed to two fan off and on radio tv off and fan on please turn off the radio and fan turn on the tv and off the light switch tv channel to three switch tv channel to two please turn off the tv and on the fan switch on the light and turn on radio switch radio's channel to one turn off radio, fan and light turn on radio, fan and switch off the light switch off radio and fan and turn the light on	1.1a, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 2.1, 2.2, 2.3, 2.4, 2.5, 3.1, 3.2
Clean the TV and fan Tune radio's channel	1.1b, 1.7b

Appendix C Heuristic Questionnaire

HEURISTIC QUESTIONNIRES ON NLIDS

Objective:

To gather and analyze the preferences and feedback for accessing natural language instruction driven smart home system (NLIDS).

Please tick (√) in the box below:

1. Age group (years old):

   [ ] below 15 [ ] 16-35 [ ] 36-65 [ ] above 65

2. Gender:

   [ ] Male [ ] Female

Ratings

Worst/Strongly Disagree	Poor/Disagree	Average/Neutral	Good/Agree	Excellent/Strongly Agree
1	**2**	**3**	**4**	**5**

Based on the ratings table above, Please tick (√) on the appropriate fields.

Description	**Ratings**				
	1	2	3	4	5
Literally					
The using of Natural Language based user interface require only **minimum** computer Literacy					
Experience in using Smart Home System is required prior to use of Natural Language based user interface in such system					
Minimize User Memory Load					
Natural language instruction is relative easier to remember					

compare to pre-defined (mouse click/button) instruction.					
Usability					
Natural language instruction is relative easier to use compare to pre-defined (mouse click/button) instruction.					
Training					
Natural language instruction is relative easier to learn compare to pre-defined (mouse click/button) instruction.					
Response					
Response is better in synthesized form (voice) rather than displayed (text) at screen.					
Integrity/Accuracy					
The given instruction is always understood and performed correctly by NLIDS					
Overall					
Natural language instruction is believed will help elderly person for accessing smart Home System.					
Natural language instruction is efficient as predefined instruction.					
Natural language instruction is **MORE** efficient as predefined instruction.					
Natural language instruction should be widely use in other instruction driven systems such as driving a car.					

**************************** Thank You ********************************

Appendix D User Manual

In order to run the proposed system, the minimum hardware and software requirement listed below are required:

D1.1 Hardware Configuration

In order to run the required software, the system must be installed with Intel or 100% compatible processors and sufficient memory. For example, a system that is installed with Pentium III 667MHz processor and at least 256 MB RAM is recommended. Besides that, we would need a video card that supports DirectX for Java 3D rendering and also a sound card for audio play back.

D1.2 Software Configuration

This section mentions the installation steps for the software that is required by NLIDS.

D1.2.1 Java 2 SDK 1.4.2. (j2sdk 1.4.2_04)

The Java Virtual Machine is needed by any system that is developed using Java Language. Therefore, Java 2 SDK is used to create a Java virtual machine for NLIDS. The installation steps are shown as follows:

i. Download the SDK and check the downloaded file size.

The SDK can be obtained at

http://java.sun.com/products/archive/j2se/1.4.2_04/index.html.

ii. Run the Java 2 SDK installer.

Double-click on the downloaded file and follow the instructions provided by the installer. Note: Administrative permission is required to install the Java 2 SDK on Microsoft Windows 2000 and XP.

iii. Update the PATH variable.

Add the full path of the C:\j2sdk1.4.2_04\bin directory to the PATH variable. Set the PATH as follows, according to Microsoft Windows NT or 98/2000/ME.

- Microsoft 2000 and XP

Choose Start, Settings, Control Panel, and double-click System. Select the "Advanced" tab and then "Environment" Variables. Look for "Path" in the User Variables and System Variables. Add the full path to the right end of the "Path" in the User Variables.

- Microsoft Windows 98

Start the system editor. Choose "Start", "Run" and enter sysedit, then click OK. The system editor starts up with several windows showing. Go to the window that is displaying AUTOEXEC.BAT. Look for the PATH statement and add the full path to the right end of the PATH.

- Microsoft Windows ME:

From the start menu, choose programs, accessories, system tools, and system information. This brings up a window titled "Microsoft Help and Support". From here, choose the tools menu, and then select the system configuration utility. Click the environment tab, select PATH and press the edit button. Now add the

SDK path to PATH variable. Once the PATH has altered, save the changes and reboot machine when prompted.

D1.2.2 Java 3D 1.4.0

The Java 3D is used to render the virtual environment of NLIDS. The installation steps are shown as follows:

i. Download the Java3D and check the downloaded file size.

   The Java 3D can be obtained at

   http://java.sun.com/products/java-media/3D/1.4.0_01/download.html.

   Note: The 1.4.0 version of the Java 3D API runs on JDK version 1.4.2 and higher.

ii. Run the Java 3D installer.

   Double-click on the downloaded file and follow the instructions provided by the installer.

iii. Installation Verification.

   Verify that the j3dcore.jar, j3dutils.jar, and vecmath.jar files are in: "c:\Program Files\Java\jdk1.4.2_04\jre\lib\ext" and the j3d*.dll files are in: "c:\Program Files\Java\jdk1.4.2_04\jre\bin"

D1.2.3 Java Media Framework 2.1.1e

The software is needed for sound playback in NLIDS. The installation steps are shown as follows:

i. Download the framework and check the downloaded file size.

The JMF 2.1.1e can be obtained at

http://java.sun.com/products/java-media/jmf/2.1.1/download.html.

ii. Run the downloaded installer.

Run the executable by double-clicking on the jmf-2_1_1e-windows-i586.exe and follow the instructions provided by the installer.

D1.2.4 TalkingJavaSDK Version 163 (Java Speech API Implementation)

Java Speech API is used as an interface between NLIDS and speech engine for speech recognition and speech synthesizing in NLIDS. The installation steps are shown as follows:

i. Download the SDK and check the downloaded file size.

The TalkingJavaSDK Version 163 can be obtained at

http://www.cloudgarden.com/JSAPI/index.html.

ii. Run the downloaded installer.

Unpack TalkingJavaSDK-163.zip or TalkingJavaSDK-163.jar into a new folder, and double-click on Setup.exe. Then follow the instructions the installer provides to install the SDK and the Java Speech API files. Copy the cgjsapi163.dll to c:\Program Files\Java\jdk1.4.2_04\jre\bin once the installation completed.

iii. CLASSPATH variable setting.

Update the CLASSPATH variable. Add the full path of the cgjsapi.jar to the CLASSPATH variable. The steps are same as the installation for JDK 1.4.2 mentioned earlier.

iv. Installation verification.

Verify the installation by running the example embedded into the SDK. The version number should print out on the console window if the installation is correct.

D1.2.5 Microsoft Speech SDK 5.1

The Microsoft Speech SDK is the speech engine used in the NLIDS for providing speech recognition and synthesizing functionality. The installation steps are shown as follows:

i. Download the SDK and check the downloaded file size.

   The SDK can be obtained at http://www.microsoft.com/downloads/details.aspx?FamilyID=5e86ec97-40a7-453f-b0ee-6583171b4530&displaylang=en.

ii. Installing the SDK.

   Unpack the SpeechSDK51.exe and run the unpacked executable by double clicking on the Microsoft Speech SDK 5.1.msi. Then follow the instructions provided by the installer. When the installation is completed, delete the downloaded file as well as the unpacked files to recover disk space.

iii. Speech Recognition Training.

The system need to be trained to recognize the speech of the user. To do this, go to the control panel, and click on the "Speech" icon. At the "Speech Recognition" tab, click the "Train Profile" Button and follow the instructions prompted.

References

Abascal, J., Bonail, B., Marco, A., Casas, R., Sevillano, J.L. (2008*),* AmbienNet: an intelligent environment to support people with disabilities and elderly people, *In Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, ACM, New York, pp. 293-294.

Akeberg, O., Svensson, H., Schulz, B., Nugues, P. (2003), CarSim: an automatic 3D text-to-scene conversion system applied to road accident reports, *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, Volume 2, Association for Computational Linguistics, Morristown, pp. 191-194.

Aldrich, F.K. (2003), *Smart homes: Past present and future*, *Inside the Smart Home*, R. Harper (Ed.), Springer London, pp. 17-40.

Allen, J. (1995), *Natural Language Understanding* (2^{nd} ed.), Redwood City: The Benjamin/Cummings Publishing Company.

Bahrami, A (1999), *Object Oriented System Development*, Singapore: IrwinMcGraw-Hill.

Bailey, B. P., Biehl, J. T., Cook, D. J., Metcalf, H.E. (2008), Adapting paper prototyping for designing user interfaces for multiple display environments, *Special Issue: User-*

centred design and evaluation of ubiquitous groupware, Springer-Verlag, London, pp. 269-277.

Ball, T., Colby, C., Danielsen, P., Jagadeesan, L. J., Jagadeesan, R., Läufer, K., Mataga, P., Rehor, K. (2000), Sisl: Several Interfaces, Single Logic, *International Journal of Speech Technology*, Volume 3, Springer Netherlands, pp. 93-108.

Begg, RK, Hassan, R (2006) *Artificial neural networks in Smart Homes. In Designing Smart Homes: The Role of Artificial Intelligence*, Juan C. Augusto & Chris D. Nugent (eds), Springer, Chapter 9, pp. 146-164

Bouvier, Dennis J. (2000), *Getting Started with the Java3D API*, Sun Microsystem, Retrieve 10 April 2008, from http://java.sun.com/developer/onlineTraining/java3d/

Chan, M., Estève, D., Escriba, C., Campo, E. (2008), *A review of smart homes-Present state and future challenges*, Computer Methods and Programs in Biomedicine, Volume 91, Issue 1, pp. 55-81.

Cox, K., Grinter R. E., Mantilla, D. (2000), Using dialog and context in a speech-based interface for an information visualization environment, *In Proceedings of the working conference on Advanced visual interfaces*, ACM, New York, pp. 274-275.

Damper, R. I., Gladstone, K. (2006), Experiences of usability evaluation of the IMAGINE speech-based interaction system, *International Journal of Speech Technology*, volume 9, Springer Netherlands, pp. 41-50.

Daniel Jurafsky, James H. Martin (2000), *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall, New Jersey.

Department Of Statistics Malaysia, *Population And Housing Census 2000*, Retrieve 10 April 2008, from

http://www.statistics.gov.my/english/frameset_census.php?file=pressdemo

Dmitry, O. Briukhov, Leonid, A. Kalinichenko, Nikolay, A. Skvortsov (2001), Personalization through Specification Refinement and Composition, In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*.

Fellbaum, Christiane (editor) (1998), *WordNet: An Electronic Lexical Database*, MIT Press, Cambridge, MA.

Fraser, Norman M., Gilbert, Nigel G. (1991), Simulating Speech Systems, *Computer Speech and Language*, Vol. 5, Academic Press Limited.

Ghorbel, M., Segarra, M.-T., Kerdreux, J., Thepaut, A., and Mokhtari, M. (2004), Networking and communication in smart home for people with disabilities, *In Proceedings of the 9th International Conference on Computers Helping People with Special Needs*, Paris, France, pp. 937-944.

Gilbert, J.E. & Zhong, Y. (2003). Speech User Interfaces for Information Retrieval. *In Proceedings of 12th Annual ACM Conference on Information & Knowledge Management*, New Orleans, Louisiana, pp. 77-82.

Grasso, Michael A., Ebert, David S., Finin, Timothy W. (1998), The integrality of speech in multimodal interfaces, *ACM Transactions on Computer-Human Interaction (TOCHI)*, volume 5, ACM, New York, pp.303-325.

Hodas, J.S., Sundaresan, N., Jackson, J., Duncan, B. L., Nissen, W. I., Battista, J. (2001), NOVeLLA: A Multi-Modal Electronic-Book Reader With Visual and Auditory Interfaces, *International Journal of Speech Technology*, Volume 4, Springer Netherlands, pp. 269-284.

Hura, Susan (2008), Designing Usable Voice User Interfaces, *HCI Beyond the GUI: Design for Haptic, Speech, Olfactory and Other Nontraditional Interfaces*, Kortum P. (Ed.), Morgan Kaufmann Publishers, Burlington, MA.

IJM, *Bayswater Resort Condo*, Retrieve 12 April 2008, from http://www.ijmland.com/projectdetail.php?projectid=bayswater

iProperty, *Smart Living Call for Smart Planning*, Retrieve 12 April 2008, from http://www.iproperty.com.my/reviews/desamillennia/coverpage2.asp

Jini Organization (2006), *Jini Network Technology*, Retrieve 26 April 2008, from http://www.jini.org/wiki/Main_Page.

Kim, S. (2008), *A Literature Survey on the Design of Speech Interface to 3D Applications*, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Koskela, T., Väänänen-Vainio-Mattila, K. (2004), *Evolution towards smart home environments: empirical evaluation of three user interfaces*, Personal and Ubiquitous Computing, Volume 8, Springer-Verlag, London, pp. 234-240.

Kroll, Per, Kruchten, Philippe (2003). *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, Addison-Wesley Object Technology Series, Boston, MA.

Lee, S. J., Kim, S. W., Ahn, K. S. (2007), Remote Control of Smart Homes Using Korean Natural Language Processing, Lecture Notes In Artificial Intelligence, *Proceedings of the 1st KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, Springer-Verlag, pp. 892-900.

Leong, C. S., Goi, B. M. (2006), Smart Home Microcontroller: Telephone Interfacing, *International Conference, Glasgow, UK, May 8-11, 2006, Proceedings, Part IV*, Springer Berlin, pp. 424-431.

Miller, George A. (1995), WordNet: a lexical database for English, *Communications of the ACM*, Volume 38, ACM, New York, pp. 39-41.

Ministry of Housing and Local Government Malaysia (1999), *Housing in the New Millennium - Malaysian Perspective*, Retrieve 30 March 2008, from http://www.kpkt.gov.my/jpn/artikel3.htm.

Mollá, D., Hutchinson, B. (2003), Intrinsic versus extrinsic evaluations of parsing systems, *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing: are evaluation methods, metrics and resources reusable?*, Association for Computational Linguistics, Morristown, pp. 43-50.

Mynatt, E.D., Irfan Essa, Wendy Rogers (2000), *Increasing the opportunities for aging in place*, In Proceedings of ACM Conference on Universal Usability.

MYREN, *Update of TEIN activities: Malaysia Research & Education Network (MYREN)*, Retrieve 10 April 2008, from http://www.apan.net/documents/download.php?f=TEIN-Ph2-Foong.ppt&fc=UpdateofTEINActivities-MYRENSEAClusterNetwork.ppt

Nepper P., Treu, G., Kupper, A. (2007), Adding Speech to Location-based Services, *Wireless Personal Communications Journal*, volume 44, Springer Netherlands, pp. 245-261.

Normaliza, A. R., Siti, N. R., Suraya, A., Wan, R., Arbaie, S. (2008), Silence and the Elderly in Malaysia, *International Journal of the Humanities*, The International Journal of the Humanities, Volume 6, Issue 11, pp. 95-100.

Orpwood, R., Gibbs, C., Adlam, T., Faulkner, R., & Meegahawatte, D. (2005). The design of smart homes for people with dementia – user-interface aspects. *Universal Access in the Information Society*, 4(2), pp. 156-164.

Park, K.H., Bien, Z.Z. (2003), Intelligent sweet home for assisting the elderly and the handicapped, *In Proceedings of the 1st International Conference On Smart homes and health Telematics (ICOST'2003)*, Independent Living for Persons with Disabilities and Elderly People, IOS Press, 2003, pp. 151–158.

Park, K.H., Bien, Z., Lee, J.J., Kim, B.K., Lim, J.T., Kim, J.O., Lee, H., Stefanov, D., Kim, D.J., Jung, J.W., Do, J.H., Seo, K.H., Kim, C.H., Song, W.G., Lee, W.J. (2007), Robotic smart house to assist people with movement disabilities, *Journal of Autonomous Robots*, Volume 22, Number 2, Springer Netherlands, pp. 183-198.

Richard Sproat (2001), *Inferring the Environment in a Text-to-Scene Conversion System*, Human Computer Interaction Research AT&T Lab, NJ USA.

Ringbauer, B. (2005), Smart Home Control via PDA, *IFIP International Federation for Information Processing*, Volume 178/2005, Springer Boston, pp. 101-119.

Roy, N., Baltus, G., Fox, D., Gemperle, F., Goetz, J., Hirsch, T., Magaritis, D., Montemerlo, M., Pineau, J., Schulte, J., Thrun, S. (2000), Towards Personal Service Robots for the Elderly, *In Proceedings of the Workshop on Interactive Robotics and Entertainment (WIRE)*, Pittsburgh, PA, Carnegie Mellon University.

SAP, *Crystal Reports*, Retrieve 10 April 2008, from http://www.sap.com/solutions/sapbusinessobjects/sme/reporting-dashboarding/index.epx

SDA (2008), EyStar SmartHome Management System, Retrieve 26 May 2008, from http://www.eystar.com/serviceGateway.html

Seffah, A., Javahery. H. (2004), *Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces*, Chichester: John Wiley & Sons.

Sleator, D., Temperley. D. (1991), Parsing English with a Link Grammar, *Technical report CMU-CS-91-196*, Department of Computer Science, Carnegie Mellon University.

Sun Microsystems (1998), *Java Speech API Programmer's Guide*, Retrieve 26 April 2008, from
http://java.sun.com/products/java-media/speech/forDevelopers/jsapi-guide/

Snyder C. (2003), *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*, Morgan Kaufmann Publishers, United of State America.

Tokunaga Takenobu et al., *Constructing a lexicon of action*, Department of Computer Science, Tokyo Institute of Technology, 2002.

Wu, H. (2001), *Supporting sensor fusion for context aware computing*, Ph. D. thesis proposal, The Robotics Institute, Carnegie Millan University

Yan Li et al. (2000), *Speech-Driven Cartoon Animation with Emotions*, pg. 365 Microsoft Research China, Bejing.

Yang Jie, Zhu Xiaojin, Gross Ralph, Kominek John, Pan Yue, Waibel Alex (1999), Multimodal people ID for a multimedia meeting browser, *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, ACM, New York, pp. 159-168.

Ying-Hong Wang et al. (2006), Application Architecture of Semantic QA System, *Journal of Computers*, Vol.17, Feng Chia University, China.

Zajicek, M. (2004), A Special Design for Special People, *In Proceedings of the International Conference on Computers Helping People with Special Needs*, Paris, pp. 624

Zeng,X., Tan, M. (2007), The development of a language interface for 3D scene generation, Proceedings of the Second IASTED International Conference on Human Computer Interaction, ACTA Press, pp. 136-141.

Zukerman, I., Litman, D. (2001), Natural Language Processing and User Modeling: Synergies and Limitations, *User Modeling and User-Adapted Interaction*, Springer Netherlands, pp. 129-158.