Voice Recognition System for Massey University Smarthouse

A thesis presented in partial fulfilment of the requirements for the degree of

Master of Engineering
in
Information Engineering

at

Massey University, Auckland, New Zealand

Rafik Gadalla

2006

Acknowledgment

I would like to express my sincere gratitude to my project supervisor, Dr Tom Moir, for his guidance and support throughout the duration of this project.

I would also like to thank my colleagues Leon Kok, Grettle Lomiwes and Vaitheki Yoganathan and other members of the smarthouse projects for their effort and dedication towards the formation of the smarthouse.

I'm grateful to my family for all their support and encouragement. In particular I would like to thank my parents for their love, patience and encouragement, my fiancé Manal, my brothers and friends whom without their assistance this project would not have been made possible.

And above all, I would like to thank God for making it all happen.

Abstract

The concept of a smarthouse aims to integrate technology into houses to a level where most daily tasks are automated and to provide comfort, safety and entertainment to the house residents. The concept is mainly aimed at the elderly population to improve their quality of life.

In order to maintain a natural medium of communication, the house employs a speech recognition system capable of analysing spoken language, and extracting commands from it. This project focuses on the development and evaluation of a windows application developed with a high level programming language which incorporates speech recognition technology by utilising a commercial speech recognition engine. The speech recognition system acts as a hub within the Smarthouse to receive and delegate user commands to different switching and control systems.

Initial trails were built using Dragon Naturally Speaking as the recognition engine. However that proved inappropriate for use in the Smarthouse project as it is speaker dependent and requires each user to train it with his/her own voice.

The application now utilizes the Microsoft Speech Application Programming Interface (SAPI), a software layer which sits between applications and speech engines and the Microsoft Speech Recognition Engine, which is freely distributed with some Microsoft products. Although Dragon Naturally Speaking offers better recognition for dictation, MS engine can be optimized using Context Free Grammar (CFG) to give enhanced recognition in the intended application. The application is designed to be speaker independent and can handle continuous speech. It connects to a database

oriented expert system to carry out full conversations with the users. Audible prompts and confirmations are achieved through speech synthesis using any SAPI compliant text to speech engine.

Other developments focused on designing a telephony system using Microsoft Telephony Application Programming Interface (TAPI). This allows the house to be remotely controlled from anywhere in the world. House residents will be able to call their house from any part of the world and regardless of their location, the house will be able to respond to and fulfil their commands.

Table of Contents

List of Figures

List of Abbreviations

SAPI	Speech Application Programming Interface
CFG	Context Free Grammar
TAPI	Telephony Application Programming Interface
TCP	Transfer Control Protocol
IP	Internet Protocol
PCM	Pulse Code Modulation
LPC	Linear Predictive Coding
FFT	Fast Fourier Transform
MFCC	Mel Frequency Cepstral Coefficient
HMM	Hidden Markov Model
ASR	Automatic Speech Recognition
TTS	Text To Speech
API	Application Programming Interface
DDI	Device Driver Interface
XML	eXtensible Mark-up Language
COM	Component Object Model
SPI	Service Provider Interface
DTMF	Dual Tone Multiple Frequency
SALT	Speech Application Language Tags
IVR	Integrated Voice Response
RMS	Root Mean Square
SNR	Signal to Noise Ratio

Chapter 1: Introduction

The purpose of this project was to develop a voice recognition system, that can be used in Massey University Smarthouse to respond to the occupants needs and desires simply by taking their voice requests and transforming them into actions. The system acts as a hub that services and delegates all voice requests to other control systems within the house.

1.1 Massey Smart house

Massey University Smarthouse is a collaborative research and development project among the Institute of Information and Mathematical Sciences and the Institute of Technology and Engineering and other industry partners. The goal of the project is to create a house where technology and appliances in the house help make life easier, safer and more enjoyable for its occupants. It responds to the needs and desires of occupants by, for example, monitoring their health, adjusting lighting, temperature, or even ambient music to their personal preferences, and wherever possible assists, them in all their daily tasks. The Smarthouse main aims are:

- Monitor the health and safety of its occupants, by using the latest in information systems and biotechnology.
- Automate common house management tasks, thus allowing inhabitants to have a more enjoyable and comfortable life.
- Provide information and entertainment to the occupants upon their demand.

It should hide the technique and details of how it works and be completely intuitive to use (Human Centred Design).

The main beneficiaries of this project will be the elderly population who want to retain their independence, and their families and friends who can be secure in the knowledge that they are safe, well and comfortable. The health sector will benefit by being able to more effectively help and monitor people in their care. There will also be a number of other benefits for the construction industry, appliance industry, and for other people who wish to improve their quality of life.

The Massey University Smarthouse [1] will be a world-class showcase for the integration of house automation, health care and smart appliance technology. Figure 1.1 provides an overview of the different components of the Smarthouse that are discussed below

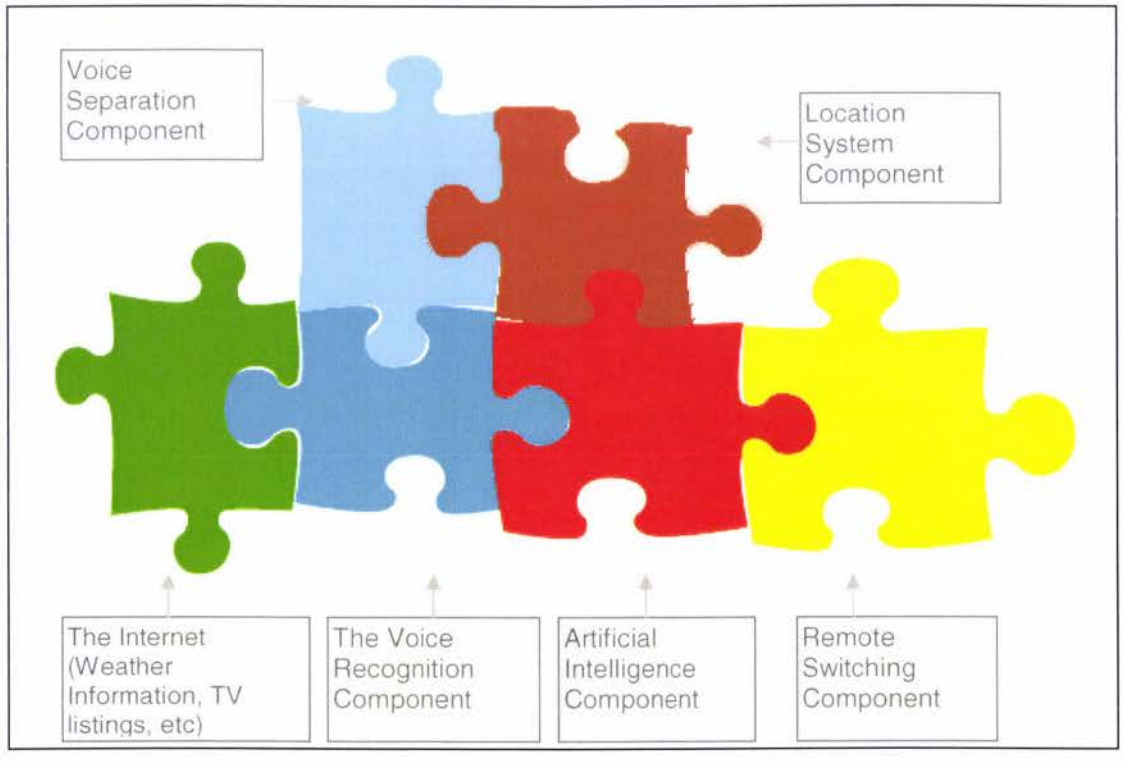

Figure 1.1 The different components of the Massey University Smarthouse and how they integrate together

1.1.1 Location and Positioning System

Tracking the position of occupants and devices within the house is essential to attain smart control and monitoring. The smarthouse therefore will be equipped with a Bluetooth ubiquitous network that consists of transceiver nodes that span across the roof of the entire house. The occupants of the house will be wearing a Bluetooth transmitting watch that contains their uniquely identifiable code that lets the house know who they are, and exactly where they are within the house.

Figure 1.2: Bluetooth watch worn by Massy Smarthouse occupants

1.1.2 Voice separation system

To enable the smarthouse to be controlled by voice, two approaches can be taken. The first is for all occupants to wear a voice capturing device, in the form of headset or watch or other. The second is to use wall or roof mounted microphones to allow for distant speech recognition. Because the first approach is restrictive to the occupants, Massey University's Smarthouse will be using beamformer microphone arrays. The development of the beamformer arrays utilise some well known beamforming algorithms to minimize noise, and provide clean, high quality speech for the speech

recognition system. The main algorithm used will be a modified version of the Griffiths-Jim beamformer.

1.1.3 House Management System

The house management system is a PC based software that contains all the rules that govern the operation of the house. It will act as the central control unit that will be communicating all the necessary information to and from other components within the house. The application will be equipped with an expert system implemented in the form of a database. The system will collect information from the location system, the speech recognition system and the different sensors within the house to manage the daily operations of the house in an intelligent manner.

1.1.4 Remote Switching System

Switching and control of appliances is made possible by a TCP/IP switching system built using an embedded system that, although capable of being used as a single and stand-alone device to aid in home-automation, also integrates into the smarthouse environment, allowing a number of smart appliances to be networked and controlled by the house management system. The device offers a simple web browser interface to show the status of connected devices at any given moment.

Chapter 2: Background

The following chapter provides some background on the theory and technologies used in this project. It also discusses the common programming interfaces and standards used in speech recognition and telephony applications. The chapter also explains previous and current related work by other universities and also the role of speech recognition in the business world.

2.1 How Speech Recognition Works

Human speech is a complex acoustic signal, made up of a large number of component sound waves. Speech can easily be captured into a digital form by means of sampling the audio signal at different intervals and quantizing it to levels that determine the amplitude of the signal at the sampling time.

To make a computer identify the different words within a speech is a complex task. Various mathematical and statistical procedures are applied to the signal before words can be identified. The process can be categorised into four high level steps:

1. Extracting discriminating speech features
2. Applying sub bound models and lexicon
3. Determining what phonemes were spoken.
4. Applying a "grammar" or set of rules to constrain the numbers of words the recogniser needs to identify
5. Mapping each set of phonemes to words and sentences.

2.1.1 Extracting Discriminating Speech Features

When an audio signal is passed from the sound card it is in a format that cannot be used by the recogniser to identify words, it has no patterns that correspond to what phonemes have been spoken. The audio signal at that format is just represented as the amplitude of the signal at different time intervals. The graph below shows time domain representation of my voice saying the phrase "Massey University".

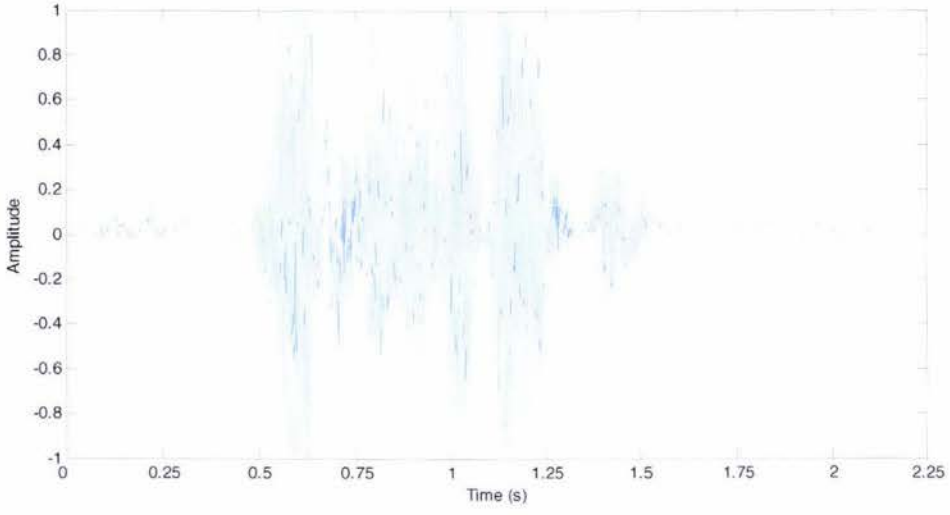

Figure 2.1: Amplitude vs. time graph for the phrase "Massey University"

To be able to identify any patterns or features, the PCM digital audio is transformed into a better representation. This is done using a number of algorithms mainly (FFT) Fast Fourier Transform followed by LPC analysis (Linear Predictive Coding) or MFCC (Mel Frequency Cepstral Coefficients). FFT works by selecting a subset of signal (typically 20ms long) at a time converting the audio data within that subset into the frequency domain. The result is a graph of frequency VS time describing the sound heard for that subset. Fig 2.2 provides a spectrograph of the speech signal above. Once the signal is in frequency domain, a number of discriminating features can be identified.

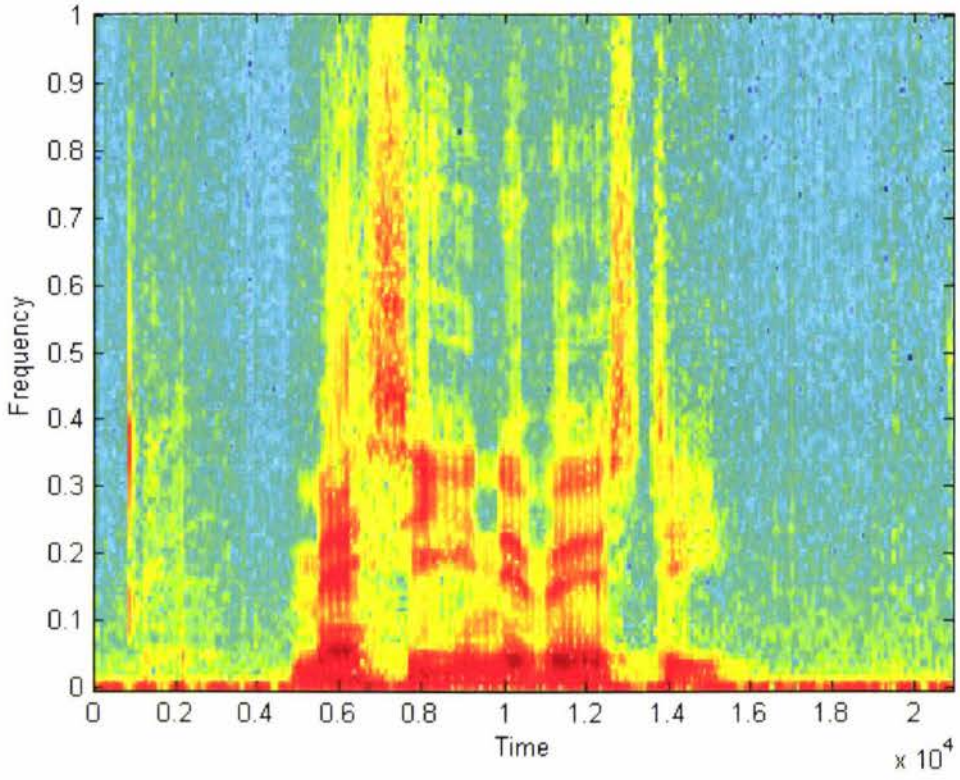

Figure 2.2: Spectrograph of the phrase "Massey University"

2.1.2 Extracting Phonemes

The resultant graphs of the FFT process and other algorithms are compared to database of several thousand graphs that identify different types of sounds in the English language. The sound is identified by matching it to its closest entry in the database, producing a number that describes the sound. This number is called the "feature number."

The next step is to match each feature number to a phoneme. However, the fact that difference people would say the phonemes differently and presence of background noise in the speech signal make this a difficult task. To get around this problem, each phoneme is constructed by using more than one feature number. The process by

which the recogniser determines which phonemes a set of feature numbers represent is reliant on statistical methods. Recognisers are equipped with statistical data on the probability of a feature number occurring in a phoneme and it uses this to determine the most likely phoneme to occur, based on a given set of feature numbers. The most commonly used algorithms for doing this is HMM(Hidden Markov Models) combined with Viterbi Search Algorithm.

There is another problem however. When the speaker says the phrase, "Massey University", he doesn't pause after saying each phoneme and therefore the feature numbers extracted are all joined together. Recognisers use HMM and the Viterbi algorithms (see section 2.4 for a further explanation) to compute not just the most probable individual words, but also sentences [2,3].

2.1.3 Applying Grammar and Language Models

Once the individual phonemes are obtained, the recogniser then goes through the process of combining those phonemes to form words by comparing each phoneme to a lexicon of pronunciations. There are a number of complications that have to be dealt with first. These mainly include different pronunciation of words by different users and figuring out when one word ends and another begins. The recogniser would apply a language model to optimise the accuracy of recognition. Language models [4] contain information about the acceptable sequence of words and are generally categorized into two forms:

Context free Grammar- this is a set of rules that limit the vocabulary and syntax structure of speech recognition to only those words and sentences that are applicable to the application's current state.

N-grams – Every language has a structure. In English, for example, it makes no sense to have two of the same verbs following each other in the same sentence. Recognisers can take advantage of that fact when recognising complete sentences to determine which words were spoken

This is best demonstrated by the use of an example. Let's assume the phrase spoken was "I like ice-cream" the recogniser might hypothesise that the spoken phrase is either "I scream" or "ice-cream". However, given that the word before was "like" it's more likely that the next word will be ice-cream rather than "I". *Figure 2.3* [5] provides an overall view of the structure of a typical continuos ASR engine.

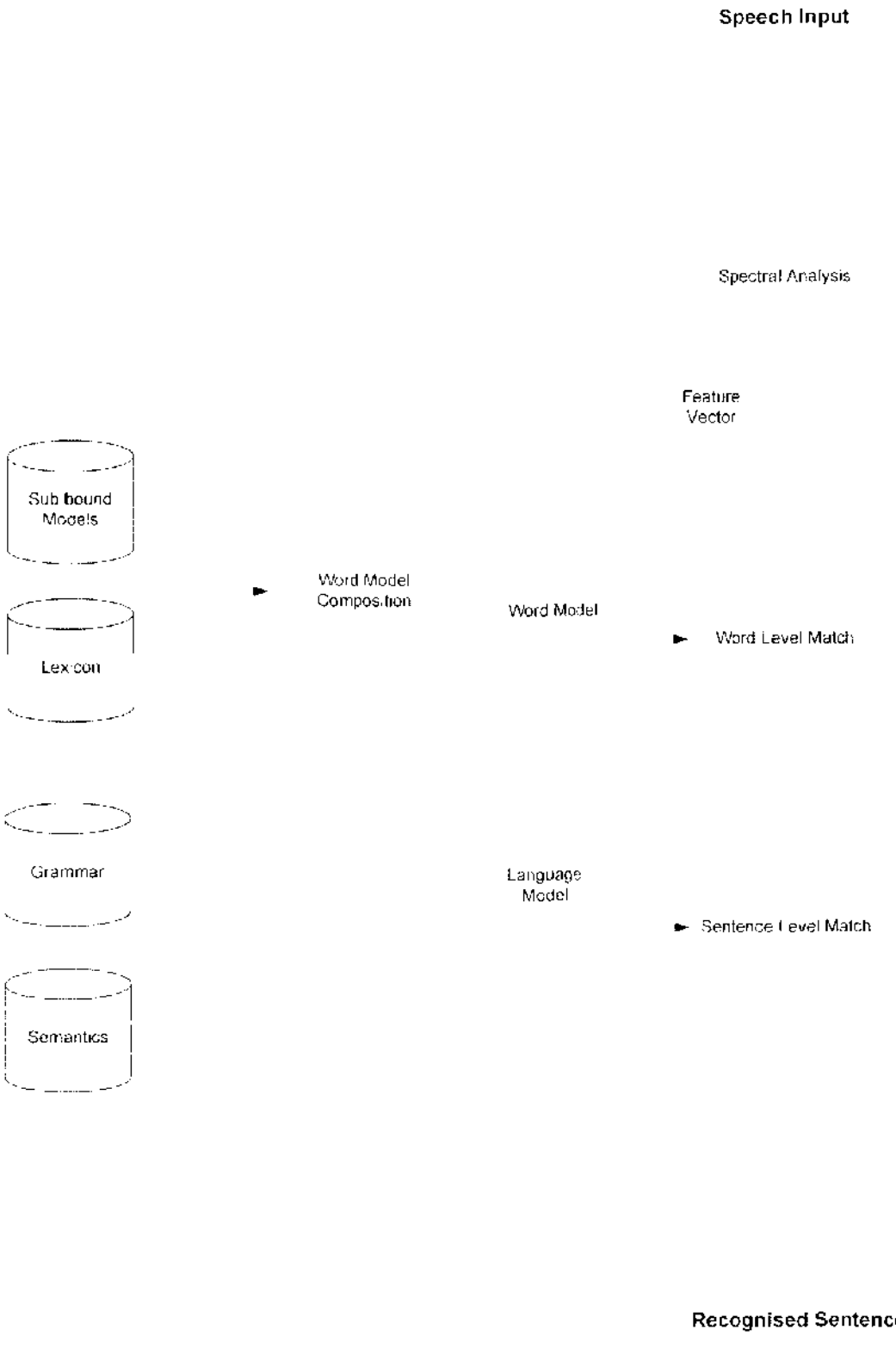

Figure 2.3: Structure of a conventional ASR engine

2.2 SAPI

The Microsoft Speech Application Programming Interface (SAPI) is a software layer used by application developers to write applications that communicate with speech recognition engines and Text-to-Speech (TTS) engines. SAPI itself is made up of two layers; an Application Programming Interface (API) and a Device Driver Interface (DDI). Applications communicate with SAPI using the API layer and speech engines communicate with SAPI using the DDI layer which handles all the low-level details needed to control and manage the real-time operations of different ASR engines.

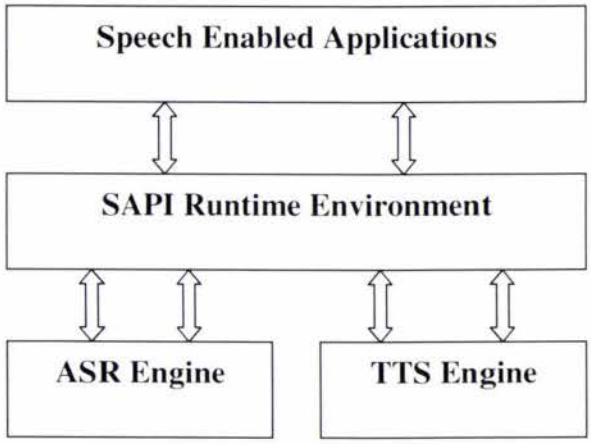

Figure 2.4: SAPI architecture

As shown in fig. 2.4, a speech-enabled application does not directly communicate with the ASR engine – all communication is done using SAPI. SAPI controls a number of aspects of a speech system, such as:

- *Controlling audio input, whether from a microphone, files, or a custom audio source; and converting audio data to a valid engine format.*

- *Loading grammar files, whether dynamically created or created from memory, URL or file; and resolving grammar imports and grammar editing.*

- *Compiling standard SAPI XML grammar format, and conversion of custom grammar formats, and parsing semantic tags in results.*

- *Sharing of recognition across multiple applications using the shared engine, as well as all marshaling between engine and applications.*

- *Returning results and other information back to the application and interacting with its message loop or other notification method.*

- *Storing audio and serializing results for later analysis.*

- *Ensuring that applications do not cause errors – preventing applications from calling the engine with invalid parameters, and dealing with applications hanging or crashing.*

The ASR engine performs the following tasks:

- *Uses SAPI grammar interfaces and loads dictation.*

- *Performs recognition.*

- *Polls SAPI for information about grammar and state changes.*

- *Generates recognitions and other events to provide information to the application.* [6]

2.3 TAPI

The Telephony Application Programming Interface (TAPI), jointly developed by Microsoft and Intel, is a set of tool and libraries , that allow computers and telephony devices to interact. It's a COM based programming interface that allows application to communicate over the PSTN or over IP telephony.

There are three major components that make up TAPI :

1. The Application Programming Interface
2. The TAPI Server
3. The Service Provider Interface

2.3.1 API

The Application Programming Interface (API) can be considered as a software layer that provides programmers with functions to access specific telephony features. In doing so, programmers have one standard and reliable method for accessing telephony hardware. This also leads to a consistent structure through out different programs, so users do not have to learn different ways of interacting with different application. The set of API's is based on the Component Object Model.

2.3.2 TAPI server

The TAPI server is the main engine that handles all requests. The API allows a program to make hardware requests, which are then passed on to the TAPI server to process. The TAPI server interacts with hardware through a layer called a service provider interface.

2.3.3 Service provider interface

Service Provider Interface (SPI) provides a low level abstraction layer that handles requests from telephony applications utilising TAPI, and translate them into the specific protocols that control the hardware device. SPIs are written by device manufacturers. The use of SPIs allows consistent higher level TAPI applications that can work with all different types of hardware. The modular structure of TAPI allows SPI's to be interchangeable without disturbing the higher interface layers; therefore, a manufacturer can write the SPI for an individual device without changing current TAPI-enabled programs.[7]

2.4 HMM

A hidden Markov model (HMM) is a statistical process used to determine the probability of occurrence of the true (hidden) state of a system given the observable states.

The system of interest is modelled as a Markov process and the goal is to try and determine the hidden parameters from the observable parameters based on the assumption that the next state is dependent on the current state (First Order Markov process) or the past K states (K^{th} Order Markov Process). In automatic speech recognition, the task is to find the most likely sequence of words **W*** given some acoustic input (in the form of extracted features). Here the extracted features are the observable states while the hidden state is the sequence of words.

$$W^* = \arg\max_{W \in G} P(W|X)$$

X is the set of extracted features (observations) = $\{x_1, x_2, x_3, \ldots x_n\}$

W* is the sequence of words that maximises the probability of **W** given **X**

G is the set of all possible Words.

The HMM can then be characterised by the set of states **S**={ S_1 , S_2 , S_3 ,....S_i }and the following parameters :

The initial state probability vector $\pi = \{\pi_i\}$

Matrix **A,** which represents the transition probabilities to go from state i to state $i+1$.

The model output symbol probability matrix $\mathbf{B} = \{Bjx\} = \{bj(x)\}$, where $bj(x)$ is the probability of outputting observation x, given that the model is in state j. [8]

Hidden Markov models are also common in others applications and used extensively in optical character recognition, natural language possessing, bioinformatics, and genomics.

2.5 XML

Extensible Mark-up Language (XML) is a plain text based language used to describe the structure and functions of the data contained within a file. XML is widely used in different applications and is growing to become the defacto standard for document exchange on the world wide web. XML uses tags much the same way as HTML, however, XML Tags describe what the data is and not how to display it [9]. One of the applications of XML is describing grammar used by ASR engines.

The following is an example of how XML is used to describe grammar:

```
<GRAMMAR LANGID="409">
    <DEFINE>
        <ID NAME="LIGHTSON" VAL="1"/>
        <ID NAME="LIGHTSOFF" VAL="2"/>
        <ID NAME="WEATHER" VAL="3"/>
        <ID NAME="NEWS" VAL="4"/>
    </DEFINE>
   <RULE NAME="PCNAME" TOPLEVEL="ACTIVE">
      <P>+jeeves</P><O>-please</O>
          <RULEREF NAME="COMMAND"/>
   </RULE>
   <RULE NAME="COMMAND">
      <LN PROPNAME="Command" PROPID="COMMAND">
         <PN VAL="1">Turn lights on </PN>
         <PN VAL="2">Turn lights off</PN>
         <PN VAL="3">Tell me the weather</PN>
         <PN VAL="4">Get the latest news</PN>
      </LN>
   </RULE>
</GRAMMAR>
```

2.6 Other Smart House Projects

In 2003, Duke University have began research into the smarthouse concept. Since then 45 different projects have emerged, mainly focused on areas of research such as; home automation techniques, safety and security and energy efficiency. Although some of the projects focus on noise cancellation for microphone technologies, not much research has gone into voice technologies itself [10].

Massachusetts Institute of Technology (MIT) Intelligent Room Project is also one of the major research work in that area. Research currently focuses on areas including the development of embedded and mobile devices that provide interfaces for smart appliances, and the design of network technologies that allow self identifying communication between mobile and stationary devices. The smart room project also looks at alternative ways of human machine interaction by utilizing perceptual technologies, for example the use of the lip movement or facial expressions to recognise speech [11].

The University of Florida is also involved in smart homes research. Their main focus is assisting elderly population with disabilities and special needs to live a safe and independent life. Some of the current research includes location tracking and positioning systems using ultrasonic and RF technologies, an automated cognitive assistant system to handle interaction with the users, and also a self sensing space system that monitors the condition of the house and its residence [12].

2.7 Speech Interface Standards

Speech interface standards describes the overall process of controlling speech functions and interacting with the user to achieve a given task. For example, the speech interface decides when the system listens, speaks, what to listen for and what to say.

Speech standards are specifications that have been formally recognised by a standards body. The most recognised standards bodies in the speech area are World Wide Web Consortium (W3C), the Internet Engineering Task Force (IETF), and the European Telephony Standards Institute (ETSI) [13].

There are several advantages to using standards when referring to speech enabled applications, both to the developers and the end users. From a development point of view, using standards reduces development time and lowers the technical risk it also allows for more interoperability. As for the end user it offers the freedom to choose from multiple sources if more than one vendor supports the standard rather than being locked in with one vendor.

There are currently two main competing standards for speech interface design, however, both standards are used for web and telephony application and hence could not be used for the smarthouse voice recognition system. Currently the most widespread standard for Win32 application development is Microsoft SAPI 5.1, which unfortunately does not offer a framework for working with speech interface.

2.7.1 VoiceXML

Voice XML was developed by the VoiceXML forum which was sponsored by a number of companies mainly AT&T, IBM, Lucent and Motorola, with the aim of developing a standard way of interacting with applications through voice.

Voice XML is an extension of the XML language that defines not only how speech enabled computer programs interact with users, but also deals with data models and procedural programming. It uses speech recognition and/or touchtone (DTMF keypad) for input, and pre-recorded audio and text-to-speech synthesis (TTS) for output.

VoiceXML was designed primarily for speech-based dialogues in telephony-based applications and uses XML Tags to instruct the voice browser to read prompts , listen for speech and manage dialogues [14]. The following is an example of VoiceXML could be used in an airline booking system.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<vxml version="2.0" lang="en">
<form>

<field name="Destination">
<prompt>What city would you like to travel too?</prompt>
<option>Auckland</option>
<option>Sydney</option>
<option>London</option>
<option>Melbourne</option>
<option>New York</option>
</field>
<field name="travellers" type="number">
<prompt>
   How many passengers are travelling to <value expr="Destination"/>?
</prompt>
</field>
<block>
<submit next="http://Webhost/handler" namelist="Destinations
travellers"/>
</block>

</form>
</vxml>
```

2.7.2 SALT

Speech Application Mark-up Language (SALT) is an open and royalty free standard for describing speech interface. SALT consists of a light weight set of XML tags that are linked to Document Object Model (DOM) properties, events and methods. SALT is used in conjunction with other mark-up languages (HTML, XHTML, WML) to add a spoken dialogue interface and telephony access to information and applications, from PCs, telephones, Tablet PCs, and wireless mobile devices.

SALT was published by the SALT forum in 2002 as a result of collaborations between various companies including Microsoft, Intel, Cisco Systems and ScanSoft working together [15].

SALT differs from VoiceXML in the following ways [16]:

- VoiceXML has a built in execution model while, SALT enables application developers to write customized dialog flow.
- VoiceXML has a high-level API while, SALT has a lower-level API.
- VoiceXML deals with speech interface, data, and control flow while SALT focuses on the speech interface only.

The following is same example that was used to demonstrate VoiceXML written using Salt. *Cities.GRXML* is the grammar file that contains a List of cities.

```
<html xmlns:salt="http://www.saltforum.org/02/SALT">
<body>

<form id="booking" action="booking.html">
<input id="input" type="text" />
</form>
<salt:prompt id=" DestinationP" >
 What city Would you like to travel too?
</salt:prompt>
<salt:listen id="DestinationL" onreco="Process_Dist()"> >
<salt:grammar src="cities.grxml" />
</salt:listen>
<salt:prompt id="TravellersP" >
  How many passengers are travelling?
</salt:prompt>
<salt:listen id="TravellersL" onreco="Process_Trv()">
</salt:listen>
</body>
</html>
```

2.8 Speech Recognition in Commercial Environments

As businesses strive to minimize costs, while increasing customer satisfaction, alternative methods of customer interaction are required to stay ahead of competitors. Of the many alternative methods available, voice recognition stands apart as it offers a seemingly natural way of interaction. Recent advances in the technology, together with the ever increasing accuracy and robustness of ASR engines has made the business case for using speech technology more solid.

The potential for enhanced customer interactions makes speech technology an attractive choice for call centres. Most call centres rely on DTMF based Interactive Voice Response (IVR) technologies which is inherently slow as it relies on complex menu systems, which the user would have to navigate through to get to the desired option. Speech recognition systems can effectively expedite that navigation process.

For example, in a flight booking call centre, if a DTMF based IVR system was to be used, and the customer was asked to select which city he would like to travel to, the system would have to read out a list of available cities and the corresponding number to enter. This process would be greatly simplified if a speech recognition system is used where customers could just name the city they wish to travel to.

In a study commissioned by Nunace Communications Inc and conducted by Harris Interactive [17], one of US leading market research companies. 66% of users who interacted with a speech automated call centre were highly satisfied with their experience. As this study was done in 2003, it is anticipated that this figure would

have improved with the recent advances in the technology over the last two years. The success of a speech enabled IVR system, however, depends more on the design of the speech interface rather than that of the speech system accuracy as found in [18].

Although speech technologies allow businesses to offer customers more convenient methods of interaction, speech technology cannot totally replace operators. Most organisations who succeed in implementing call centres using speech technologies only do so to reduce the load placed on operators, so more time could be spend dealing with unordinary customer issues.

There are several commercial Speech Recognition Platforms available for large scale enterprise use, of which the main four are:

- Nuance Voice Platform 3.0
- Microsoft Speech Server 2004
- Scansoft SpeechPAK Application Kits
- IBM WebSphere Voice 2.4

(At the time of writing this documents Nunace and ScanSoft were in the process of merging)

Chapter 3: Problem Formulation and System Requirements

3.1 Problem Formulation

The recent proposal for Massey University to set up a smarthouse has meant that certain key technologies associated with speech recognition and communications will have to be investigated in some depth. The smarthouse requires speech recognition in order to switch appliances, lighting, TV, etc. Although speech recognition systems are quite commonplace nowadays, their application in a smarthouse environments has not been tested or proven before. There are a large number of commercial speech recognition engines available, however, most of them are intended for use in dictation where the application only coverts voice and display it as text on a screen. The most common uses for such type of recognisers are office tasks like writing letters, filling forms and sending emails.

Massey Smarthouse required an application that does not just recognise speech and convert it to text, but rather something that can understand what is being said to it and respond with the appropriate words or actions. The success of the Smarthouse project depends on a number of factors. From a speech recognition point of view, the application must give the impression that it is intelligent and approachable. To enable this, the application must be able to respond promptly to users requests. It also has to understand different users with different voice tones and accents. The ability to recognise voice in environments with realistic ambient noise is one of the main obstacles that this Smarthouse project will have to overcome to provide a medium that

is natural and convenient. Another important aim for the project is to avoid occupants having to wear headsets with microphones to be able to communicate with the house. Instead, the house will be equipped with arrays of beamformer microphones that can capture sound from different directions. Much research is currently being undertaken by other project members to reduce background distortions to a minimum, however, it's still anticipated that there will be some noise issues.

Selection of the right speech recognition engine to use is not a trivial task. There were no published papers that directly compare the features and reliability of the latest versions of those recognisers when used in command and control applications. The decision had to be based on the following factors:

1. Ability to limit the number of words or phrases to a finite set, which is commonly referred to as Context Free Grammar (CFG). This can significantly improve the accuracy and performance of the application.
2. Cost of deployment. Some of the commercial recognisers and text to speech engines cost in excess of $5000 NZD, while others require separate servers to run as they demand much processing power.
3. Support for telephony applications.

A clever speech interface design can significantly improve the usefulness of the speech recognition feature. The interface must make the user feel in control, but also ask specific and targeted questions to keep the conversation within boundaries. It also needs to respond with appropriate confirmations that it has acknowledged what the user has asked for to avoid ambiguity. The design should also take into account that

users can word commands in different ways, for example while someone might say "Turn lights on" another person will say "Switch Lights on"

The application must be configurable and maintainable to suit different environments, and the user should be able to alter commands or actions if he/she desires. Therefore a graphical user interface should be present where users can see how the application is preforming and alter some of the settings to give optimum performance.

The implementation of the system takes into consideration three emerging software engineering paradigms discussed in [19] that enhance the use and simplifies the deployment of smarthouse software. These paradigms are visual programming, component-based software construction, and connection-based programming.

3.2 Specifications

The following points summarise the requirements of the application to be developed:

- Continuous speech recognition system to recognize commands of up to 15 words long
- Win32 application or windows service with GUI configuration utility
- Speaker independent
- User configurable commands also an easy to use wizard to create new commands
- The software program must be easily installed by any intermediate-level computer user.
- Provide all the necessary functionality to monitor the program state at any given point in time
- All environmental variables must be configurable at runtime

- Respond to all user commands in an acceptable time frame. no more than 2 second lag between end of command utterance and first response made by the system
- System must inform the user weather the command uttered was recognised or not
- Integrate with a local or Remote (over the internet) Expert System
- Only respond when a certain keyword is said
- Ability to carry on a dialogue with users with the assistance of an Expert System
- Ability to confirm certain actions with user before committing and/or request further information
- Initially single threaded application that can be converted to a multithreaded application at a later stage
- Utilize voice synthesis technology to respond back and deliver information to user
- Keep a log of all recognised and misrecognised commands and actions taken along with timestamps for each event
- Integrate telephony functionality to allow the applications to listen to commands over the PSTN or VOIP
- Extensively test the noise robustness of the final product

Chapter 4: System Implementation

4.1 Speech Interface Design concepts

The Smarthouse required that most of its day to day functions be controlled by voice. Therefore a speech recognition engine with the ability to accept custom commands had to be utilised. The voice commands used for controlling the house are chosen to be relatively short in length, such as three to fifteen words at most. The selection of very common words or expressions as commands should be avoided where possible to prevent possibilities of premature/false recognitions in a noisy house environment. To help speed up the recognition process the recogniser only listens to a small and fixed set of vocabulary as opposed to all words in the English language. That contains all the necessary commands. This is achievable through the use of CFG.

The speech interface is implemented in a way so that for a user to control the house he must first say a keyword to let the system know that next sentence is a command. For the purpose of this project the keyword "Jeeves" (the name of the virtual butler of the house) was chosen. For example if a user would like to turn the lights on he would say "Jeeves, turn lights on" and not just "turn lights on". Appropriate choice of this keyword is critical as the choice of a common English word could lead to incorrect and premature response from the system.

The system also needs to inform the user that it has recognised a command and ask for more information or confirmation if needed. The system does this by using synthesised voice which was made possible by using a TTS engine. In the event where the user does not respond back to the system, the system would time out if no

valid response has been received within 5 seconds, and command execution would be cancelled.

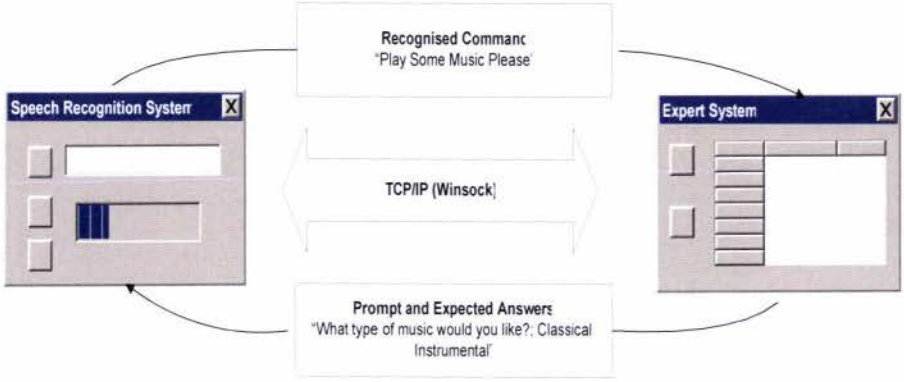

Figure 4.1: Communication between the speech system and expert system

By utilising CFG which is supported by all SAPI compliant speech recognition engines, we can create a set of grammars that defines all the possible commands used. CFG is specified in a separate file using XML syntax. The files contains a set of rules that can be made active or inactive at run time. A CFG file is referred to as Static Grammar and it is only used for initial commands. Once a conversation has been started with the user the Expert System builds dynamic grammar at runtime and passes it through to the Speech System. See figure 4.1 for an over view of interaction between the Expert System and the speech system.

To help improve the recognition process the concept of "Word Spotting" [20] is utilised. Word spotting allows a command and control application to appear more intelligent by listening only to a particular phrase and ignoring any other utterances before or after. For example if a user says "Jeeves, turn the lights on" or "Jeeves, would you please turn the lights on" the application would only spot the words "turn the lights on" . This obviously allows the system to appear more natural.

4.2 Speech Recognition Implementation

4.2.1 System Initialization

When the application starts a number of processes need to take place. Mainly the application will read all the commands from the database and then build an XML file that defines the grammar which is going to be used. The application then loads this grammar into memory, ready for recognition. Other processes that occur during the initialisation state include TCP/IP initialisation, TTS Engine Initialisation and loading the grammar files. As these processes could take a long time if there are a large number of commands in the database, the application displays a splash screen (*figure 4.2*) to let the user know the application hasn't crashed.

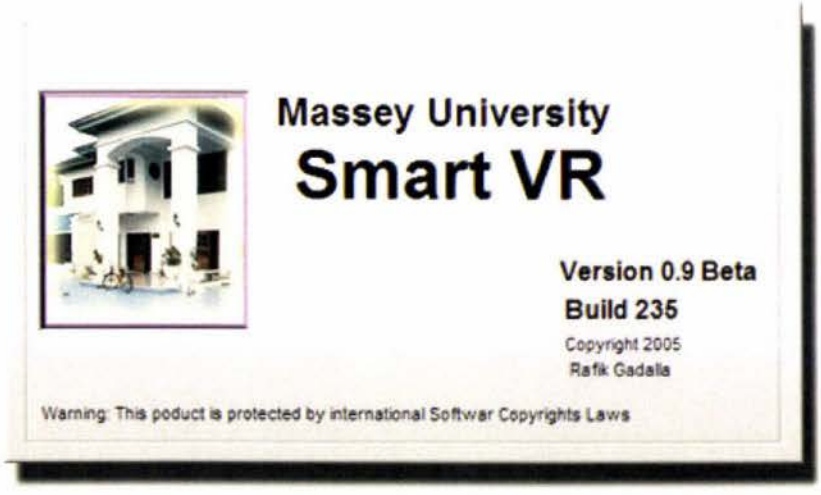

Figure 4.2: Application's Splash Screen

4.2.2 Command Execution

Once the initialisation phase is completed, the application uses API calls to the speech recogniser to determine whether a phrase of interest has been detected or not. The process works as follows:

By using the ISpRecoContext interface the application establishes a connection to the SR engine. It then uses the ISpRecognizer interface to control certain aspects of the ASR engine. This includes which audio input device or speaker profile to use. The application also uses this interface to determine what events it is interested in. An example of these events are SPEI_RECOGNITION, SPEI_HYPOTHESIS, SPEI_FALSE_RECOGNITION, SPEI_SOUND_START and many others. By using a suitable grammar file and the ISpRecoGrammar interface the application manages the words and phrases that the SR engine will recognise. Once recognition has occurred, the ISpRecoResult interface is used by the application to retrieve information about the SR engine's event [6].

If the event retrieved is indeed a recognised phrase as set by the static grammar file the application would then pass that phrase on to the expert system by using the TCP/IP protocol. The application then waits for a reply message from the Expert System. What happens next depends on the contents of that message, the application could either ask for more information from the user and creates a dynamic grammar in memory with all the expected answers or deliver a final response/action. The whole process will continue to happen until a terminate message is received from the expert system or the user stops responding. The system will then return to its initial state and waits for the next command to be issued.

4.2.3 Performing other Functions

The application will be deployed in different environments and hence all settings will need to be customizable to allow the system to be fine tuned for each environment.

The system also takes into considerations that different users might like to use different commands than the default ones. The system offers a graphical user interface that allows users to create new words, train the system to understand a different pronunciation of an existing word, etc. All these functions are discussed in more detail in section 4.3.

4.2.4 Commands Database

To allow concurrent and efficient access to commands, a relational database was created to provide a centralised source for commands and their corresponding functions. The database was implemented using MS SQL server 2000 and can provide the necessary information about commands and how they are handled by the system. Although this database was designed by myself, the expert system that interfaces with this database and determines how a command is handled is written by one of the other Smarthouse project members (see introduction).

Figure 4.3 provides a conceptual view of the database structure that can be employed by the expert system to handle commands and carry a limited conversation with the user to extract the necessary information. When a user issues a command that has been recognised by the speech system, the expert system will look up that command in the commands database. Based on the *ATH_LVL* field the system will determine whether the user has the right authority level to issue that command. The system will then create a response and a set of expected answers based on the contents of the *Expected_Answers* tables. When a user comes back with an answer, this answer is checked against the *Expected_Answers* table and the next action is determined based on the *Act_Id* field. By looking at the *Act_Typ* filed of the *Actions* table the system determines what type action needs to be executed. The different types of possible

actions are explained in the *Actions_Ref* table. Initially there will be three types of possible actions:

1- "Do" Action: This is where the system will make a call to the switching and control unit, to physically alter a property of the house, for example switch the lights on.

2- "Say" Action: This is where the system will speak to the user for example read out the news, or weather forecast.

3- "Lnk" Action: This is where the system will link back to another response. This happens when the system needs more information from the user. The system will continue to loop until a "DO" or "SAY" action is perceptible. An example of this situation is when the house user says "read me the news" in which case the system would need to get more information as to what type of news is required, e.g., sports, world, local, etc.

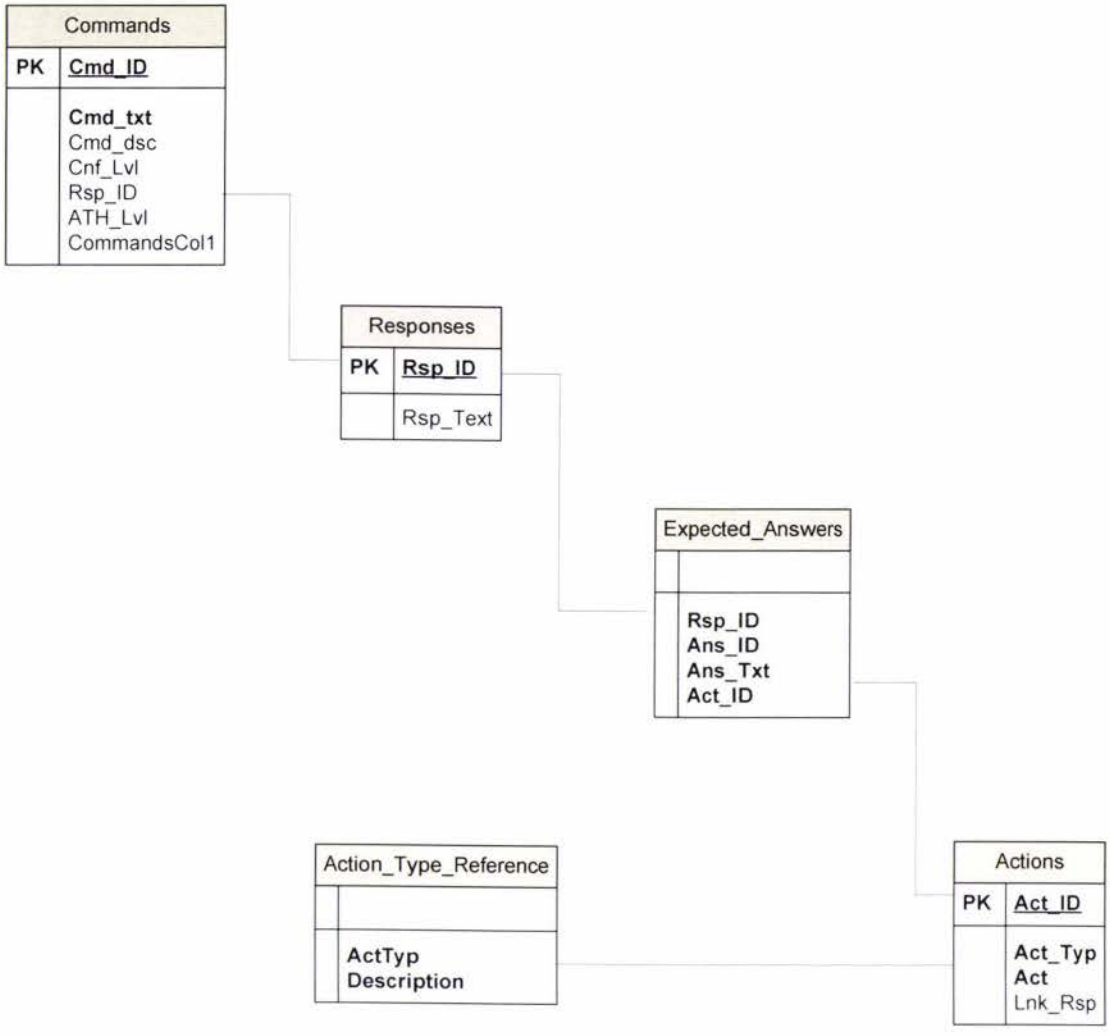

Figure 4.3: Smarthouse command database schema

4.2.5 Speech Recognition flow chart

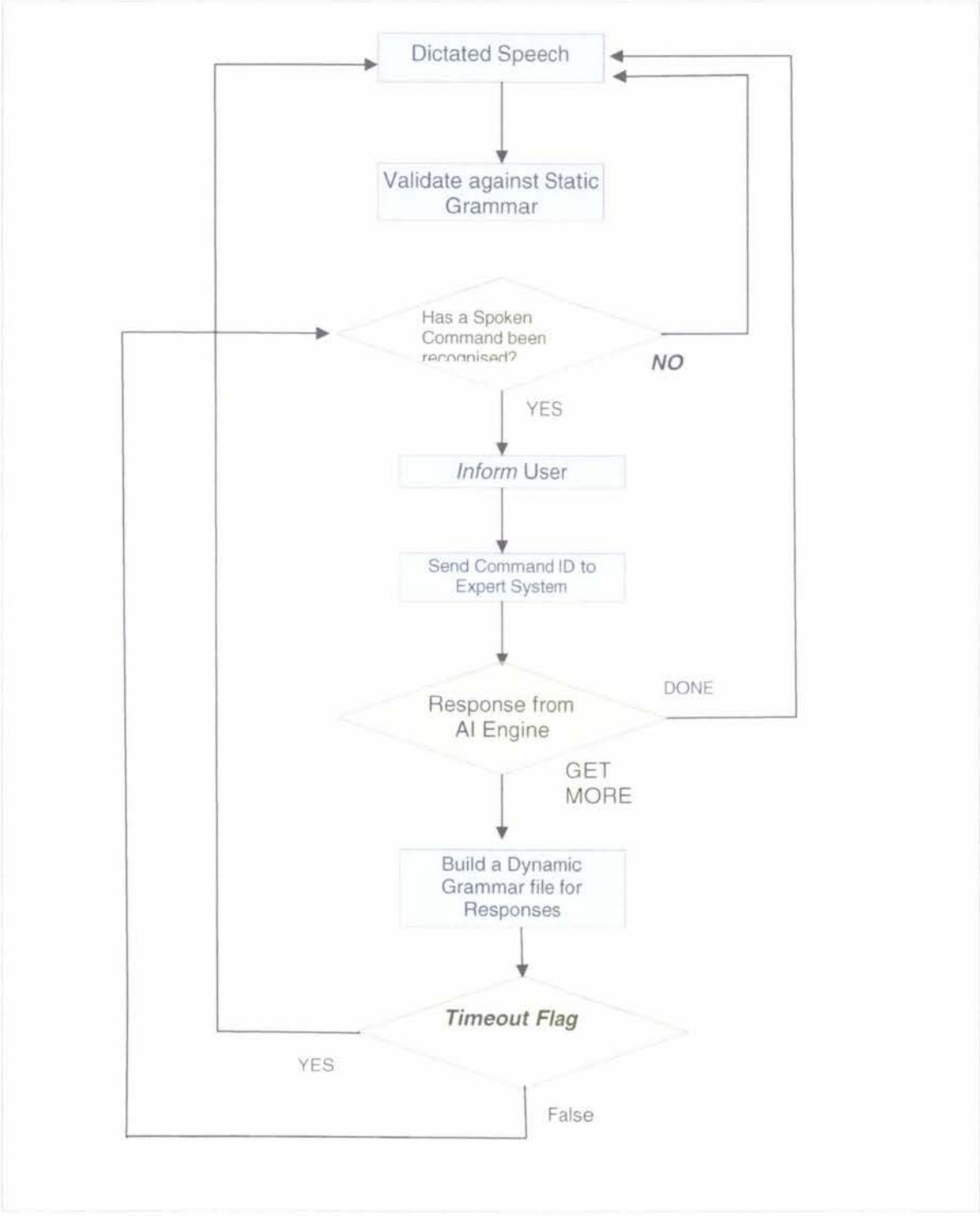

Figure 4.4: Simplified flow chart of the speech recognition application

4.3 Telephony Implementation

To allow the house to be remotely accessible, other interaction interfaces were exploited. The combination of telephony and speech recognition presented the most accessible way to allow for remote interaction with the house. Once the speech recognition program was functional, effort was put in to allow sound input and output to be directed to telephony stream.

A separate application was built to handle telephony interactions. The application utilises both SAPI and TAPI to achieve the required task and has to take into account certain considerations including:

- Calls made to the house in error.
- Handling other calls while engaged in a conversation.
- Allow only a subset of the commands to be spoken.
- Security and authentications.
- Handling calls from a noisy environment.
- Allow DTMF inputs in addition to speech recognition.

As the main goal of this project was the in-house speech recognition section, the telephony application was built to fulfil only some of the requirements above for the purpose of demonstration. The flowchart of the application is given in *Figure 4.5*.

The application was integrated into the existing Speech Recognition application and by using Status flags the Main application was made to switch between in-house and

telephony audio sources. By default the main application initialises only in-house recognition once a notification for an incoming call is received the application sets the telephony flag, switches to the appropriate ASR engine for telephony recognition and initialises the required components necessary for other telephony recognition. There have been some bugs with integrating both applications however, mainly due to the limited support of Visual Basic to threading.

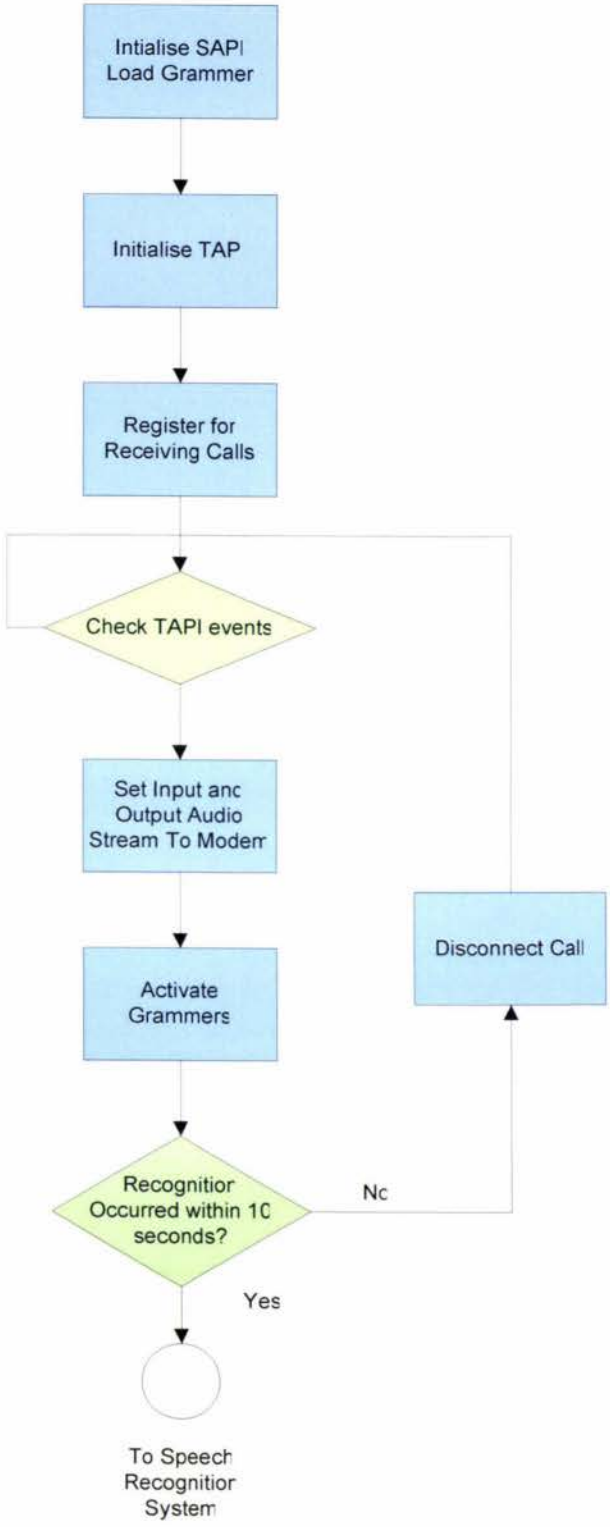

Figure 4.5: Flowchart of telephony handling application

4.4 Graphical User Interface

4.4.1 GUI Design Considerations

Users want developers to build applications that meet their needs and are easy to use. Most people regard the user interface as the software itself. Intuitive user interfaces are important for reasons such as ease of use and simplicity of training other people to use it. Support costs are reduced at the same time as well, due to less people needing help with the application.

Functionality is just as important in an application, but the way it's delivered by the interface is also important. A central concept in designing successful user interfaces is *consistency*. A program should react to user inputs in a consistent manner. For example, objects such as buttons should be in the same places throughout the application. The same colour scheme should also be used [21].

Adhering to previously set standards is also a good idea. Microsoft has set up software design standards in 1995 [22], which define the use and layout of application components. In this project I have attempted to adhere to these rules. Simple things such as the application's top menu layout to the use of the buttons and labels were designed to reflect the current design rules as shown in *Figure 4.6* exhibits. Standard naming conventions are used along with the standard shortcut keys assigned.

Figure 4.6: Standard naming convention used in the application's to menu

The wording of messages and labels within the application should be appropriate as well as informative. This will affirm to the user that he/she is in control.

Objects in the application should be grouped effectively. Logical grouping of related buttons and fields should be grouped to communicate their relationship to the user.

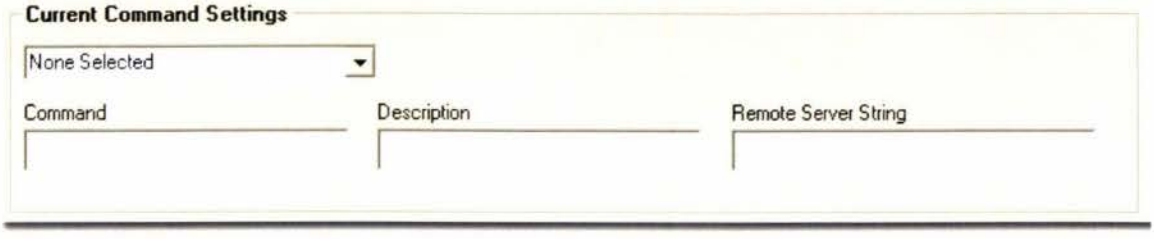

Figure 4.7: Logical grouping of components

Figure 4.7 displays the logical grouping of components in the application. All the command settings and information are grouped with a surrounding frame to create a visual border from the rest of the components. This helps to communicate to the user which items on the form are related to each other.

4.4.2 Application Interface

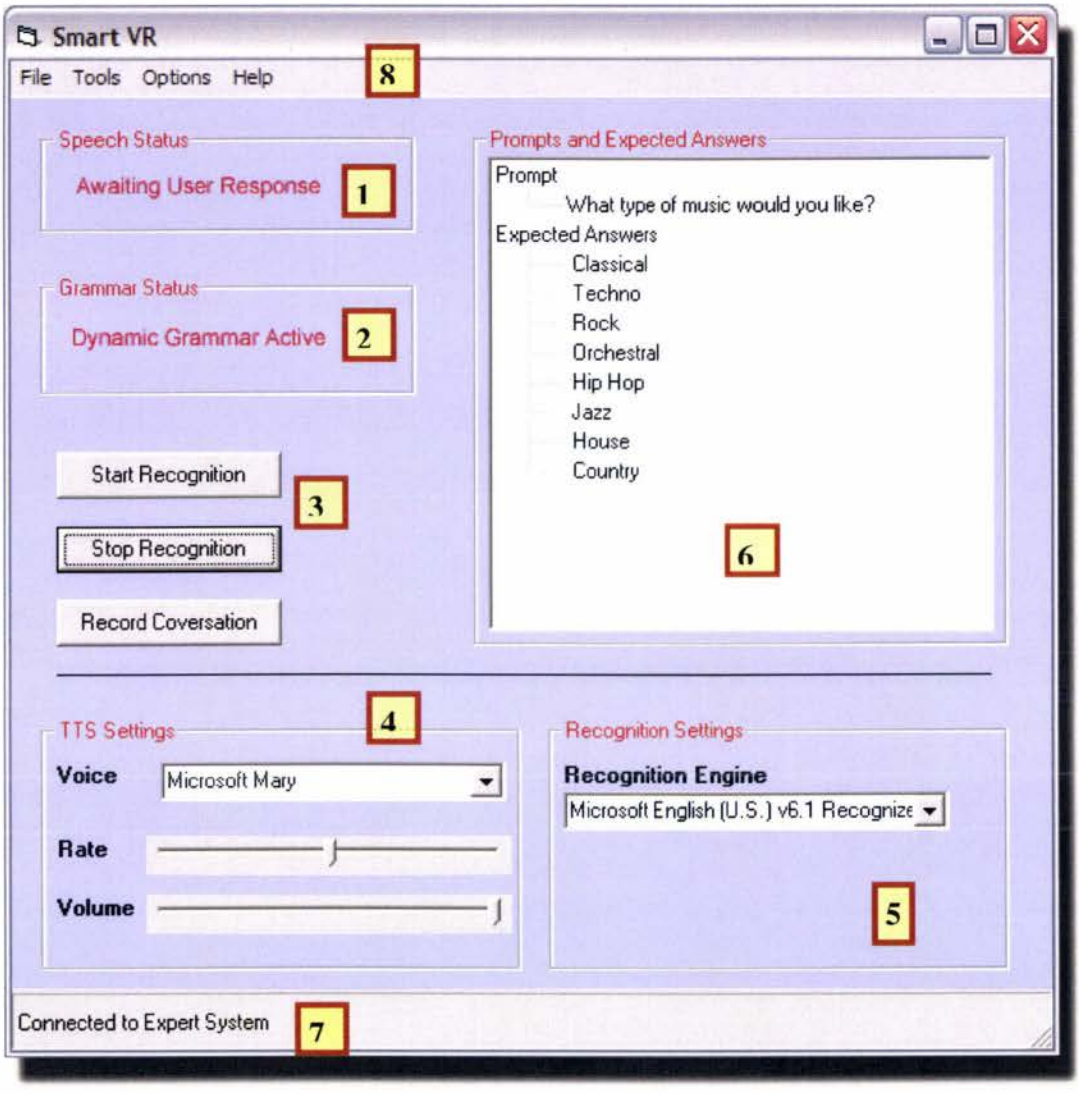

Figure 4.8: The application's main form. The different buttons and dialogs are numbered one to eight.

1. **Speech Status** – This displays feedback about the application's status with regards to speech. It can have several different values such as "Awaiting Command", "Awaiting User Response" or "Speaking"

2. **Grammar Status** – This simply displays the current active grammar file if static grammar is in use or "Dynamic Grammar" if dynamic grammar is in use.

3. **Function Buttons** – This area provides 2 command buttons to enable and disable speech. This can be helpful when the user wants to temporarily disables speech. A "Record Conversation" button is also provided, when this button is pressed the program calls a third party application to record whatever conversation is taking place. This functionality is useful for analyzing the performance of the application offsite.

4. **Text to Speech Controls** – This area of the program allow users to adjust TTS settings to their preferences. A list of all SAPI compliant voices is listed in a combo box and can be selected at runtime. The speed and volume of the voice can also be varied.

5. **Speech Recognition Controls** – This area consists of a combo box that allows users to select which SAPI compliant ASR engine is used to process the incoming audio signal.

6. **Prompts and expected answers** – This area is only active after a user issues a command. Any prompts or expected answers received form the expert systems are displayed here. Expected answers can be considered a list of options that the user can choose from.

7. **The Status Bar** – This status bar mainly displays the application status with regards to the connectivity to the Expert System. Its displays messages such as "listing on Port <Port Number>","Connecting" ,"Connected to Expert System" or any error messages the could occur.

8. The application's main menu includes different options. The most important being the **Tools** menu that contains functions inherited from Microsoft Windows Speech Controls.

A. **Mic Training Wizard** – This option allows the user setup and adjust the microphone input level for most accurate speech recognition. (*Figure 4.9*)

Figure 4.9: Microphone Training Wizard

B. **Train New Word** – This option allows the user to train a new word and add it to the main dictionary to be available for recognition later on. (*Figure 4.10*)

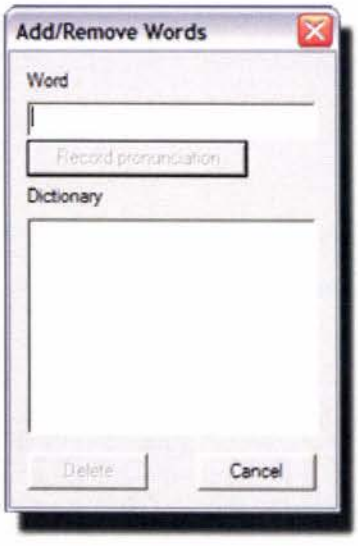

Figure 4.10: Add New Words Wizard

C. **Train User** – in the event of only a single user using the system, recognition accuracy can be significantly increased by going through the user training wizard. This wizard which allows the user to customize the acoustic models to his/her voice. (*Figure 4.11*)

Figure 4.11: User Training Wizard

The inclusion of the above three wizards in the main application is important to make it as stand-alone as possible.

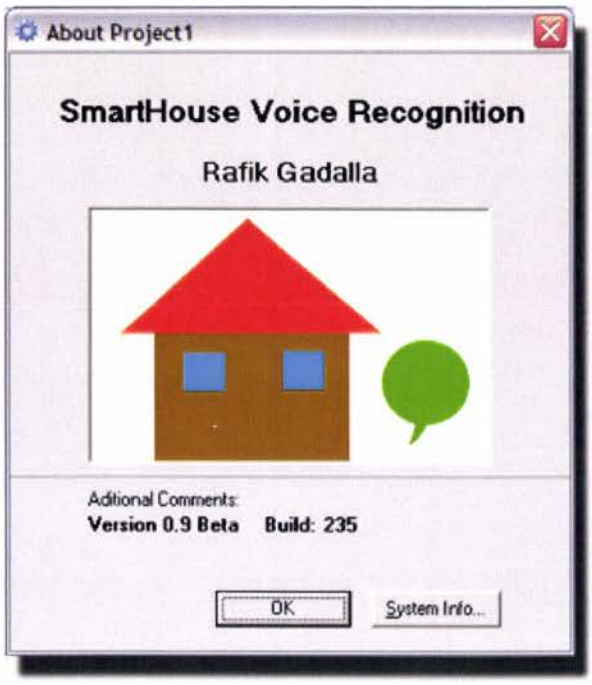

Figure 4.12: Help-About Window

Finally, there is an "About" dialog (*Figure 4.12*) providing program version information. The "Systems Info..." button provides the user with a detailed summary of his or her computer. This is useful when providing support for the application – the user has easy access to and can supply relevant information about his/her computer.

Chapter 5: System Testing

5.1 Preliminary Tests

There are several commercial speech recognisers available on the market today. Before the implementation of the project was to commence, a decision had to be made on which ASR engine would be the best to use. The decision must take into consideration a number of factors determined by how and where the application would be used. In a smarthouse environment, one of the most important considerations is to have an ASR application that is speaker independent or one that can switch between users quickly enough without the user experiencing a long delay. Other important factors to be considered are the ability to have a custom finite vocabulary set and context free grammar. This significantly increases recognition accuracy and speed.

In addition to the above features, the recogniser must be able to handle relatively noisy environment, for example, the house user should not have to turn off the TV to say a command. Furthermore, the ASR engine needs to be easily deployable on a standard home PC and not requiring expensive server hardware to run. The cost of the ASR itself must be accounted for as this could vary greatly among products.

Feature comparison analyses were carried out on different commercial ASR products to determine which product would be the most ideal. Results of this analysis and a feature comparison are presented in Chapter six. The findings of other related projects with regards to which ASR engine to use were also taken into account [23] [24].

5.2 Final Evaluation Methodology

Once the application had been developed a number of tests had to be carried out to evaluate the performance of the application. There are a number of standard criteria to asses the performance of a speech recognition system, including Word Error Rate (WER), Match Error Rate (MER), Relative Information Lost (RIL) and Word Information Lost(RIL) [25], however, these methods are more suited for a continuos dictation system rather than simple command and control. The evaluation methodology discussed in this chapter take into account the recommendation in [26] which states that a speech recognition system should be evaluated with respect to the application it is used in.

At the time of writing this thesis the expert system that speech application relies on to process and executes user commands was not completed, therefore a small program was written to simulate this expert system and allowed us to have and indication of the overall performance of the system.

Testing of the system was done by placing a subject speaker inside a quiet room with sound recording equipment. The speaker was then asked to read a number of command phrases which is then recorded and altered digitally to introduce different types and levels of noise and then fed into a special application that used the same ASR as the main application. That application counts the number of phrases recognised correctly and outputs the results on the screen for further analysis. This is discussed further in section 5.2.6.

5.2.1 Room Setup and Recording Equipment

To obtain an optimum test environment an isolated acoustic chamber was desirable, however this was not available, so all the tests were carried out in a medium sized (4x5 m) underground room. The room was well isolated from external noise and the only noise source present was that from the air-conditioning vents, however this is not a significant issue as it is constant throughout the experiment. The sound pressure of the room was measured using a DSE sound level meter to be 45 dB ± 2dB.

Recording was done using a Logitech desktop microphone connected to a notebook computer with a SoundBlaster pro soundcard. There was a number of recording software programs available, however Adobe Audition was chosen for this procedure as it was more powerful and was relatively easier to obtain than other software packages. Audio files were recorded in 22KHz, 16bit PCM format as this is the acceptable format for the SAPI engine.

5.2.2 Selection and training of Subject Speakers

In a typical house environment, there are likely to be a number of users with different ages, gender and accents. Because the system will not be trained for each individual user, it is important to test how the system responds to a variety of different voices. For this experiment a group of four speakers participated in the test:

Speaker 1. Female with American Accent

Speaker 2. Female with NZ accent

Speaker 3. Male with a foreign language as his first language

Speaker 4. Male with Scottish Accent

Although Speaker 3 has another language as his first language, he could also speak English fluently.

Before the recording began, each speaker was asked to do trial readout of the commands. This was done for two reasons. First to get the speaker familiar with reading the commands and second was to tune the microphone to the optimum volume required by the ASR. The microphone tuning wizard of windows XP was used to do this process for each speaker.

Once the microphone adjusting was completed, each speaker was asked to read one command at a time with a four second interval between the start of each command. To improve consistency and make the process effortless for the speakers a program was created in C# to display each command. When the start button is pressed the program would display one command at a time and after four seconds it displays the next command. Each command was displayed in a different colour to make it easier to identify the transition from one command to another (see *Figure* 5.1).

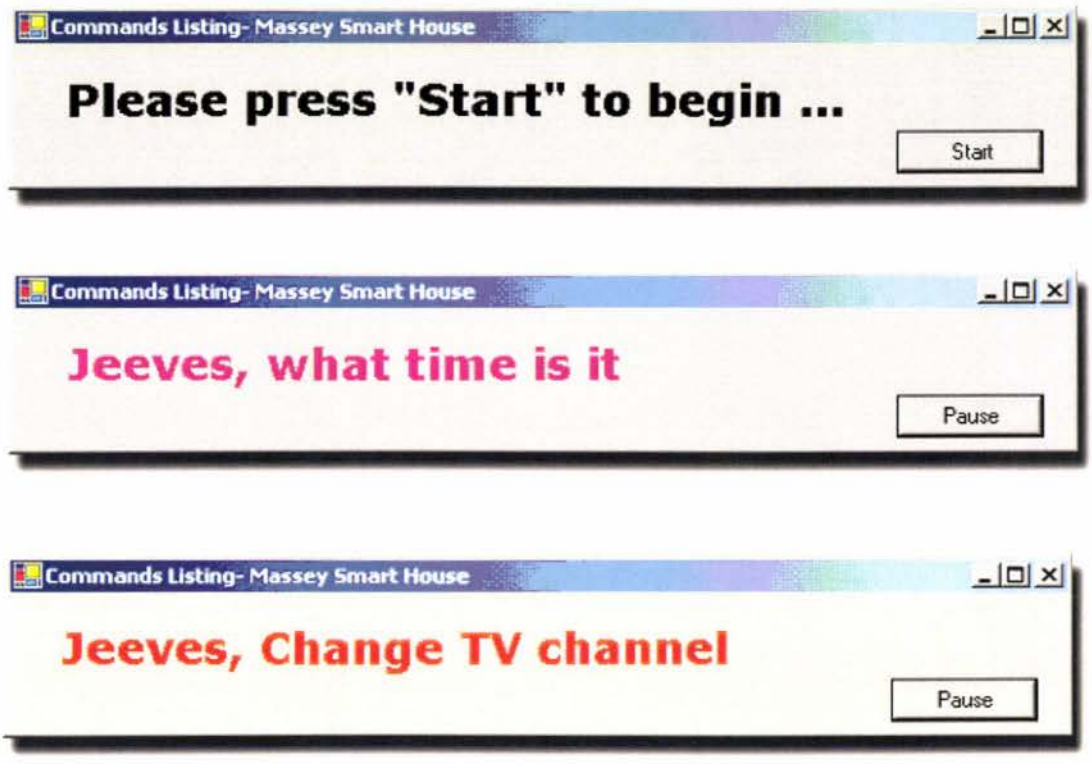

Figure 5.1: Program used to assist speakers to read commands

5.2.3 Design of command phrases

The smarthouse will need to be able to recognise a large number of commands possibly more than 200. Ideally testing should be done with a number of commands close to that, however as this process would be incredibly time consuming for the volunteer speakers a command set of only twenty phrases were chosen, and instead the grammar file was allowed to have a larger number of commands (40 commands). This should be sufficient to give a fair indication of the accuracy of the ASR.

Each command was prefixed with the same keyword this proved to increase the accuracy of command recognition and in a house environment will avoid any false recognition if that command is mentioned in a normal conversation. The same keyword mentioned in the previous chapter was used (Jeeves).

5.2.4 Introduction of Noise

The application will need to preform accurately over a different range of signal to noise ratios, although the house will employ some noise cancelling microphones, it is expected that there will still be some noise present. To test the impact of noise on the speech recognition accuracy, different types of noise that represent the main types of noise present in a normal suburban house, were added to the original recordings. The different noise types were:

- Noise 1- Residential neighbourhood ambience - Exterior - Fairly Active Background, Off Stage Construction, Occasional Off Stage Garbage Truck, Off Stage Car Start and Driving Away, Car In, Medium Close Up By and Away, Off Stage Resident Walla

- Noise 2- crowd,indoor,10-15 people, talking , glasses clink - A vintage recording selection (obtained from International SFX Library).

- Noise 3- Group of children talking outside – (obtained from Coll Anderson Sound EFX)

- Noise 4- Ambient electronic – (obtained from Simon Wolfe Project CD).

Each noise type was obtained from its respective source. The noise recording was then resampled to 44.1 KHz and repeated multiple times. The noise recording was then normalised so that the maximum amplitude was -3dB. This was necessary to

ensure that noise and the speech sound signals both had the same power. Each noise recording was then stored in MS Windows PCM WAV file format.

A Matlab script was used to automate the mixing of the noise and speech files with different SNR. Special care had to be taken to stop noise from overlapping silent gaps between commands as this would confuse the ASR engine at lower SNR. The full listing of the script is presented in the appendix section. The following describes algorithm employed by the script:

1- Read speech and noise signals into a Matlab vector.

2- Measure the RMS of the speech signal.

3- Step through the speech file and measure the RMS of the signal for an interval of five milliseconds and compare this value against the RMS value of the whole speech.

4- If the RMS of the window is greater than a certain factor **X** multiplied by the RMS of the whole speech then this part is active speech and noise should be added; otherwise if the RMS value of the window is less than that of the whole speech then this is a silent pause and no noise should be added.

5- Repeat step 3 and 4 for different signal to noise ratios.

The choice of factor **X** mentioned in step 4, was best determined by trial and error. A value too high (e.g. 0.8) would mean most of the speech would be treated as silence and a value too low (e.g 0.05) would mean any noise in the silence period would be regarded as active speech. To minimize any errors resulting from choosing an inadequate factor, a complementing technique was used by which the pause period between one command and another was manually adjusted to complete silence. This was done for each original recording using an open source software utility (Audacity).

The script creates eight different files representing SNR for each unique combination of noise type and speaker. The SNR in decibels is given by the formula

$$SNR = 10\log_{10}\left(\frac{Signal}{Noise}\right)$$

Given the SNR value the ratio of signal to power was calculated and applied.

5.2.5 Telephony Testing

In order to test the performance of the ASR engine with speech transmitted over the PTSN network, telephone quality speech was simulated by applying a band path filter (300-3500 Hz) to each of the above mentioned sound files and was analysed in the same manner. The band path filtering process was done using Matlab. A 4th order Butterworth filter was chosen as it offers a monotonic and flat magnitude response in the passband. The transfer function of a Butterworth filter in the digital domain is given by [27] :

$$H(z) = \frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{-1} + \cdots + b(n+1)z^{-n}}{1 + a(2)z^{-1} + \cdots + a(n+1)z^{-n}}$$

The Matlab script used to implement this process is shown below:

```
[speech,FS,BS]   = wavread(SpeechFile);
wc2=3500/FS/2;
wc1=300/(FS/2);
wc=[wc1 wc2];
[b,a] = BUTTER(4,wc);
filtered_speech = filter (b,a,speech);
soundsc(filtered_speech, FS)
```

5.2.6 Analysis of Sound files

To analyse the speech and noise recordings obtained, and quantify the results, a special application was created that uses the same ASR as the main application. The applications accepts a WAV file as input and tries to recognise the individual commands spoken in the file and outputs the results onto the screen. The application uses the SAPI SOUND_END event to determine the end of each command. If no recognition occurs a carriage return is output to the screen. The results are then compared to the actual commands spoken and a score out of 20 is tabulated for graphical analysis.

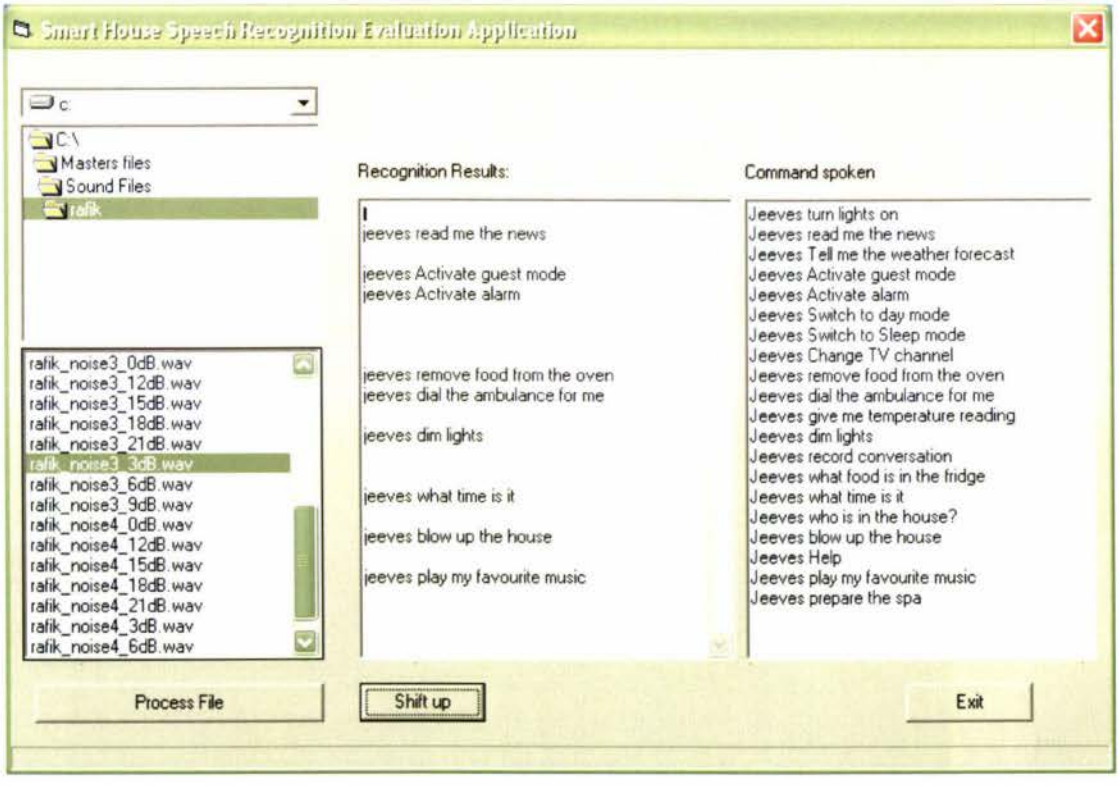

Figure 5.2: Program used for testing and ASR engine accuracy

Chapter 6: Results and Discussion

6.1 ASR Engines Feature Comparison

As indicated in section 5.1, most commercial ASR engines were evaluated before the implementation of the system took place. The sections below briefly outline the strengths and weaknesses of each engine.

6.1.1 Dragon Naturally Speaking

Dragon Naturally Speaking is a product from ScanSoft Software Inc. It is regarded as one of the best speech recognition programs available, however it is speaker dependant and requires every speaker to train it with his/her own voice. It's mainly used for dictation, but can be used in Command and Control mode. SDK for program is available although not well documented, which makes development more difficult. The program requires large amount of CPU power to run.

6.1.2 Microsoft SAPI Kit

SAPI is the industry based standard for windows platform speech programming. It offers an excellent framework for developers who want to integrate speech into their application. Included with SAPI 5.1 is Microsoft Speech Recogniser Version 5.0 which can be used in speaker independent mode and can provide good recognition results when used in command and control environment. Microsoft Recogniser version 6.1 is also available and comes bundled with Microsoft Office 2003. Excellent recognition results, good documentation, low cost of deployment and compliance with coding standards made this the ASR of choice for this project.

6.1.3 Vocon 3200

Vocon is also a product of ScanSoft Inc. and is primarily aimed at the automotive industry and embedded developments. The program also runs on most PC operating systems. The program is highly portable and provides speaker independent command and control. Subjective tests on a sample application built on the Vocon Engines showed excellent performance in noisy environments. SDK is available, however, it is very expensive.

6.1.4 IBM via Voice

Originally developed by IBM, ViaVoice is a great rival to Dragon Naturally Speaking, however, the product is now marketed and supported by the same company, ScanSoft Inc. The program offers similar features to Dragon NS, but can operate in speaker independent command and control mode. SDK is provided by a third party company named Wizard Software.

6.2 System Evaluation Results

In Order to asses the performance of speech recognition application in a smart house, a series of experiments were conducted to evaluate the effect of different noise types and speakers on the recognition accuracy of the system. It is noted that the purpose of this experiment is to assess the overall recognition accuracy of the application with relation to its use within the smarthouse and not just that of the particular ASR used.

The results of accuracy for speakers with different accents under different SNR are presented in *Figures 6.1, 6.2, 6.3* and *6.4*. The results indicate that the system achieves maximum recognition accuracy at a SNR around 12dB. The best recognition accuracy of 100% was achieved for speaker 1 and the worst was 90% for Speaker 3. This is due to the fact that Speaker 1 has an American accent and the ASR engine was trained using US English voices where as speaker 3 have a slightly different pronunciation as English is not his first language.

The impact of different noise types present in a home environment on recognition accuracy was also studied and graphical results are presented in *Figures 6.5, 6.6, 6.7* and *6.8*. The graphs below show that the ASR engine can cope with some noise types slightly better than others. This is evident in the situations of Noise 1 (traffic/wind noise) and Noise 2 (children talking) where the system was able to achieve higher accuracies at lower SNR. This can be explained by the fact that wind and traffic noise occurs at lower frequencies that does not overlap much with human speech and hence the system was able to filter it out correctly. As for the Noise 2, children tend to pause a lot while talking and hence the speech signal is not completely overlapped by the noise, which resulted in higher recognition accuracy.

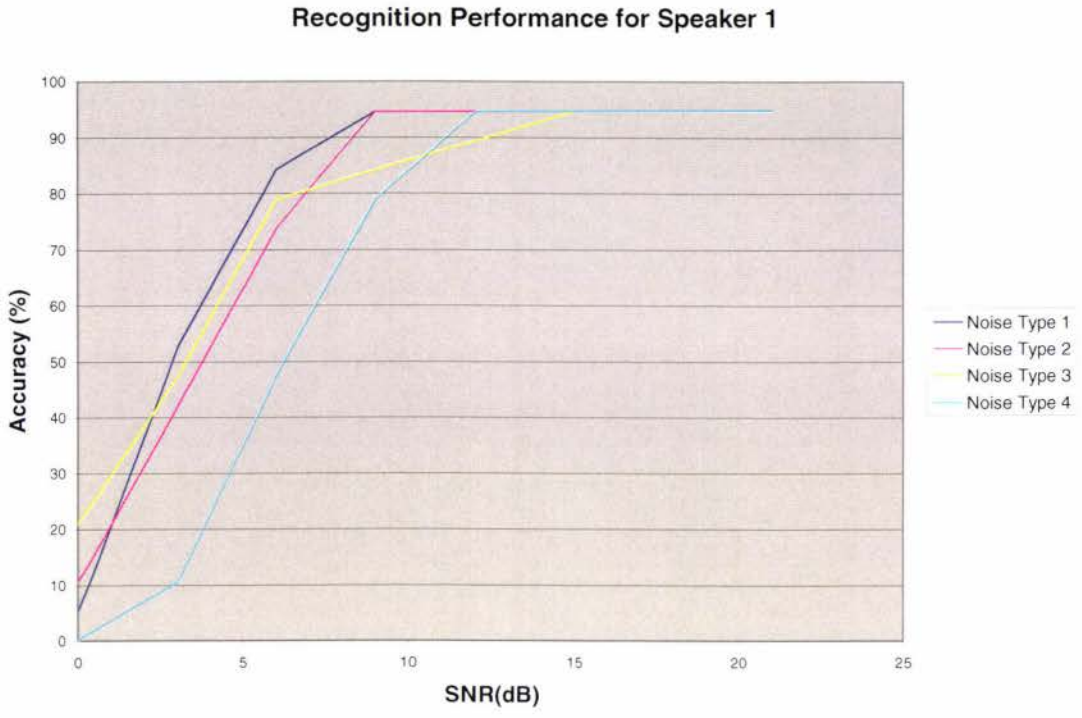

Figure 6.1: Recognition accuracy graph for Speaker 1 (female with American accent)

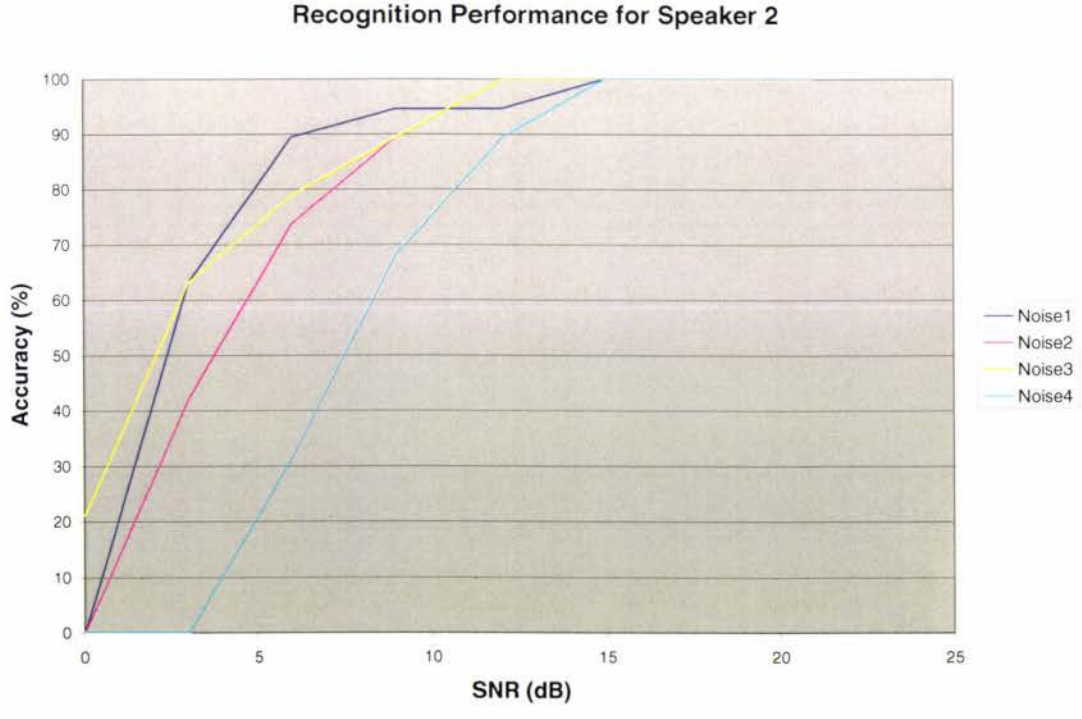

Figure 6.2: Recognition accuracy graph for Speaker 2 (female with New Zealand accent)

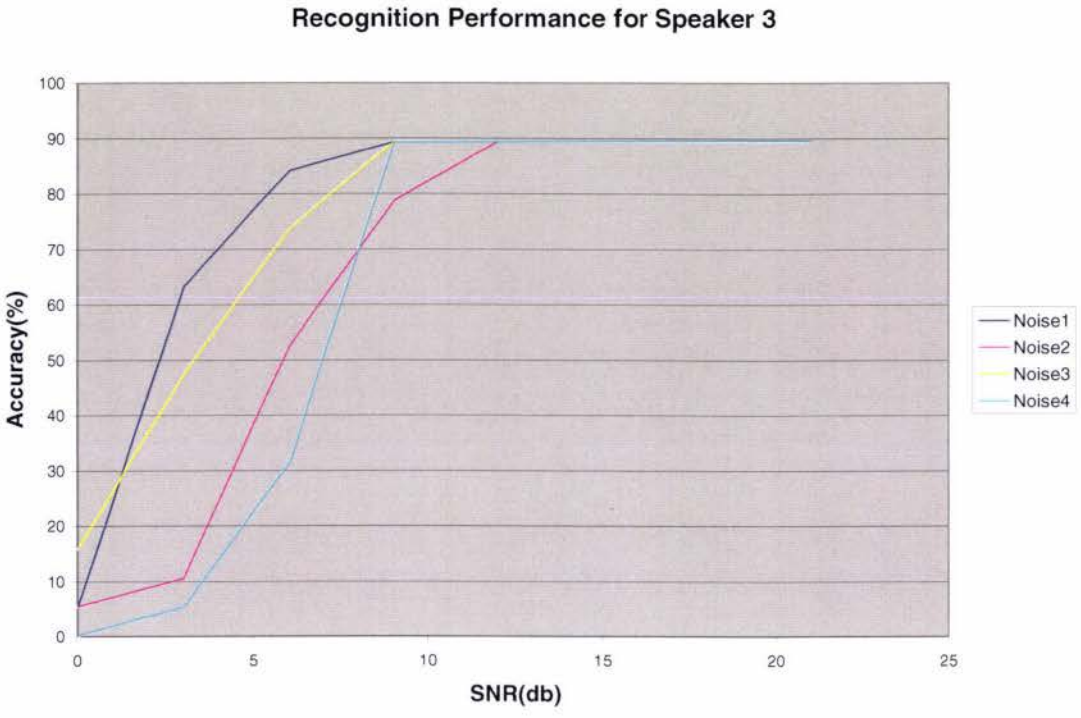

Figure 6.3: Recognition accuracy graph for Speaker 3 (male with foreign accent)

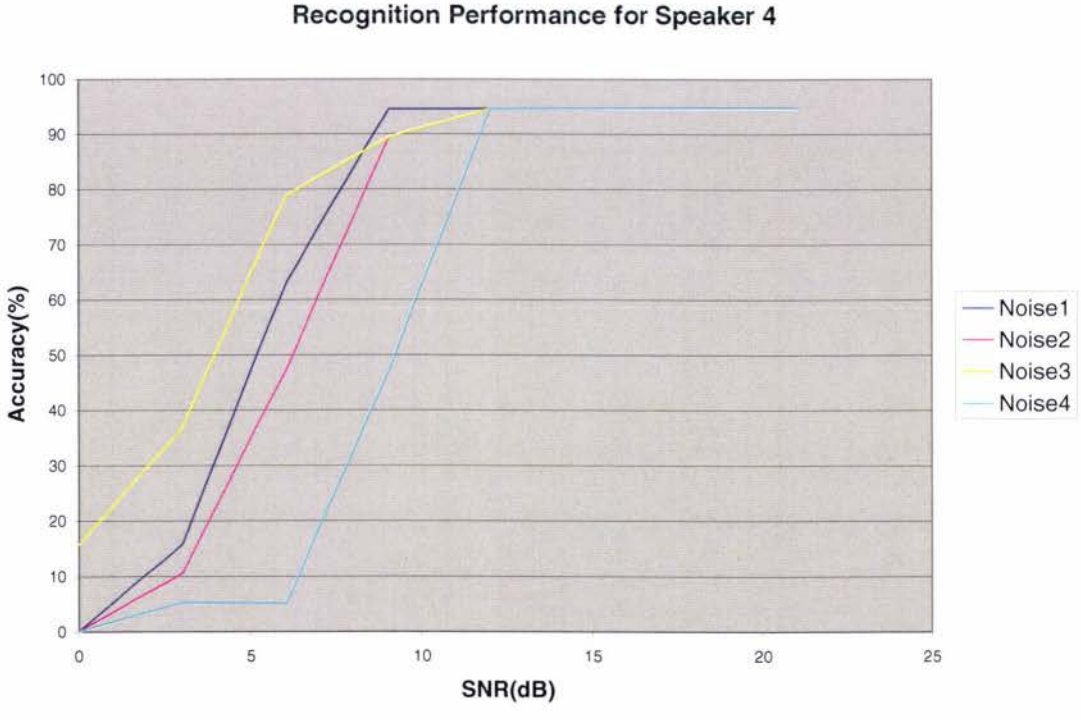

Figure 6.4: Recognition accuracy graph for Speaker 4 (male with Scottish accent)

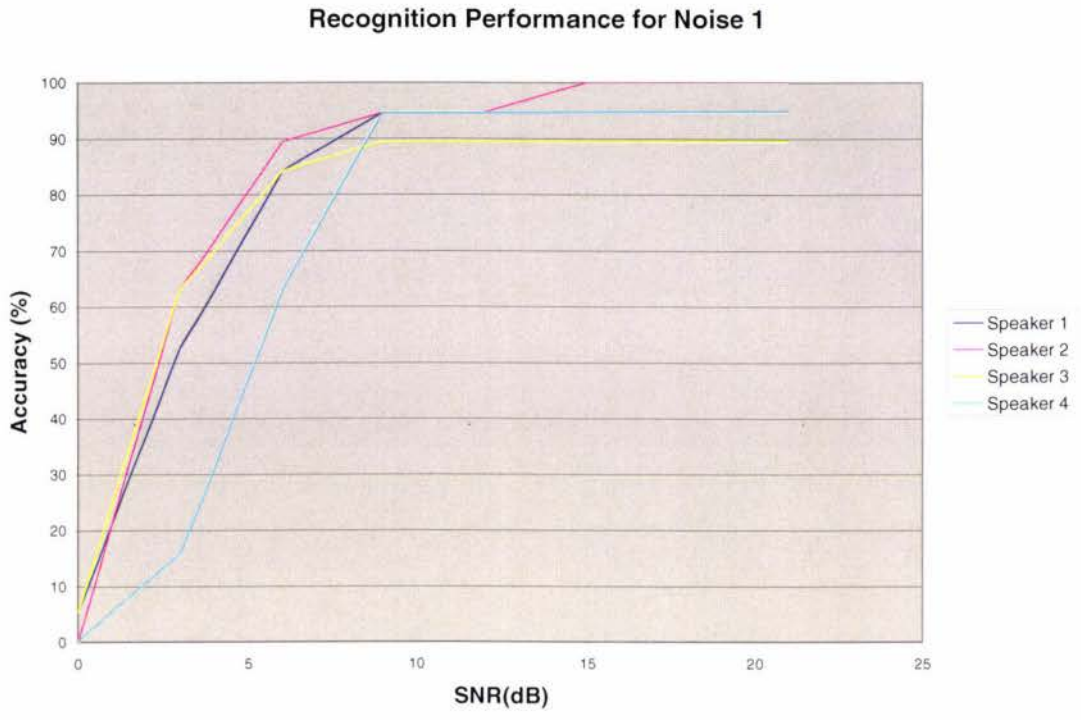

Figure 6.5: Recognition accuracy graph for Noise 1 (Traffic Noise)

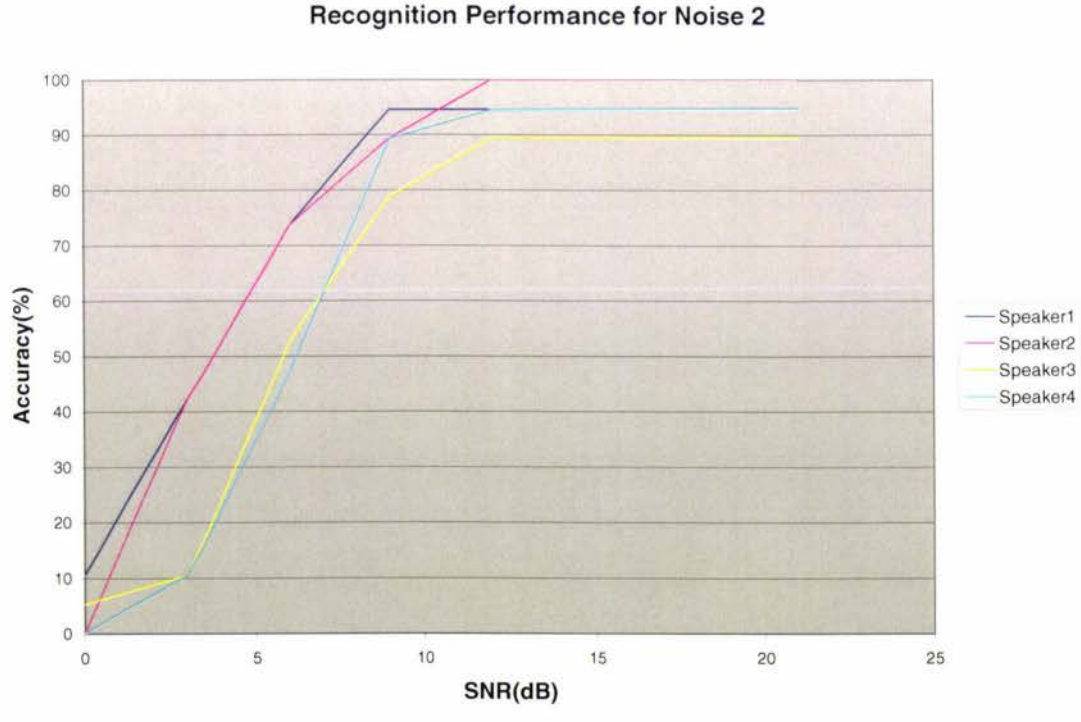

Figure 6.6: Recognition accuracy graph for Noise 2 (Crowd Noise)

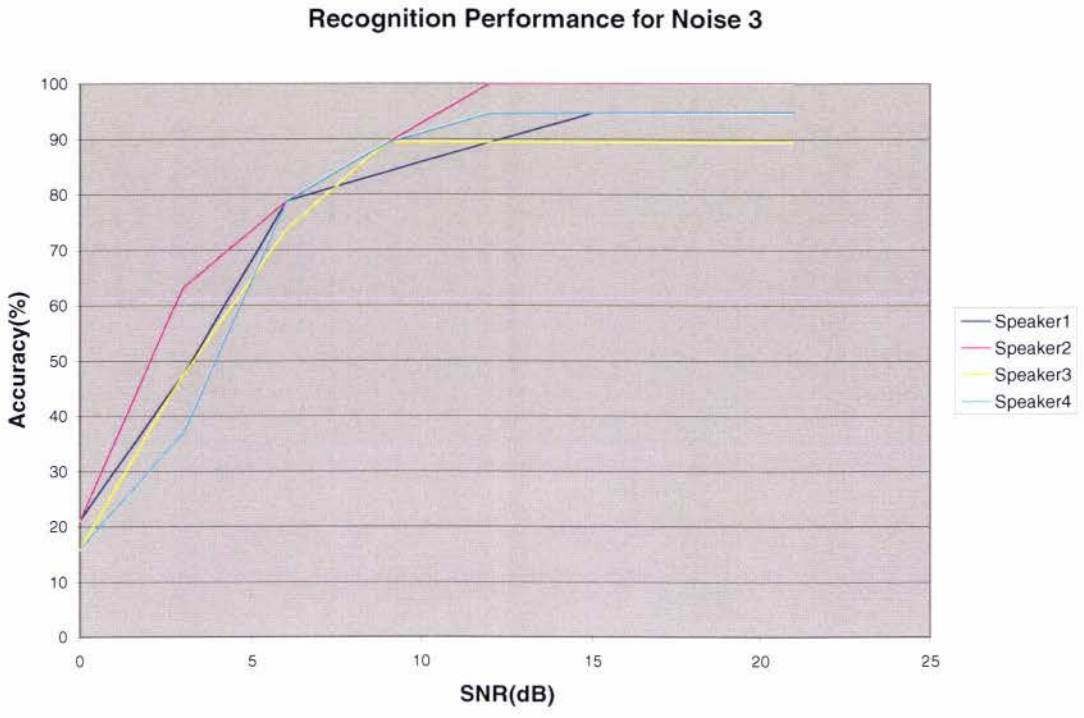

Figure 6.7: Recognition accuracy graph for Noise 3 (Children Talking Noise)

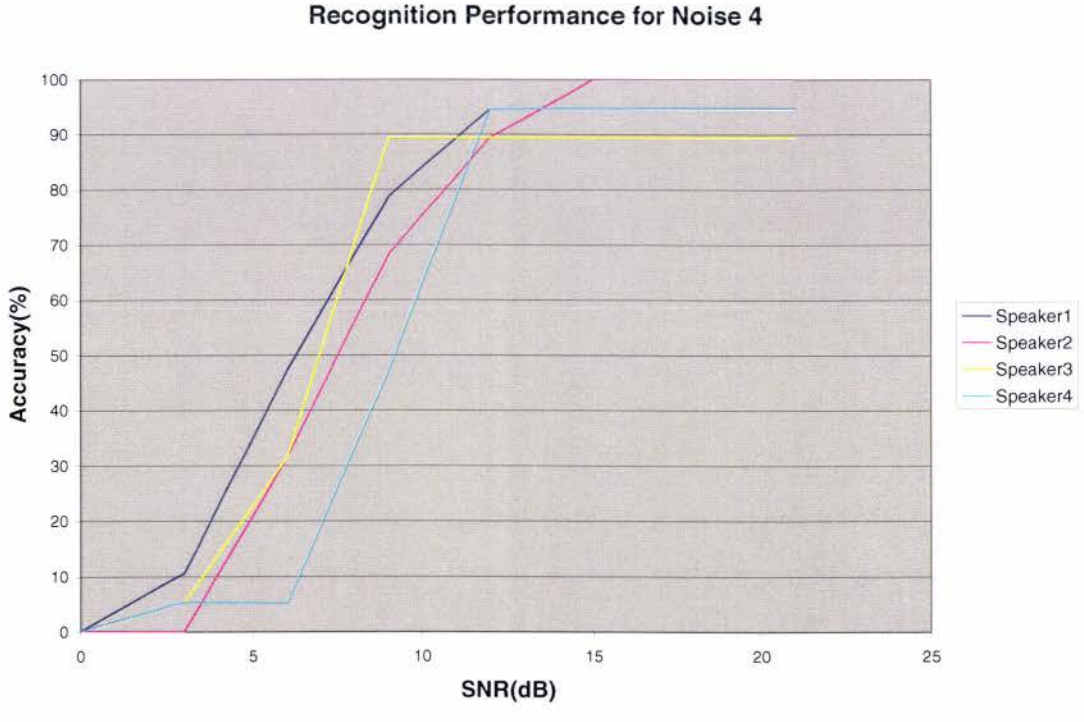

Figure 6.8: Recognition accuracy graph for Noise 4 (Ambient Music)

Telephony quality speech have yielded very poor results using Microsoft Recogniser v 6.1 as the ASR engine was not particularly trained for this type of recognition. To overcome the problem, Microsoft Telephony Recongiser v7.0 was used. The main application should be able to dynamically switch between recognisers based on the speech input type. The graphs presented in *Figure 6.9* reflect the recognition accuracy results for speaker 1. Unfortunately a maximum accuracy of only 73% percent could be obtained, it was noticed however that the engine had preformed better in real life testing than it did in simulation. This perhaps could be related to the fact the engine was not optimised to accept sound form a pre-recorded source.

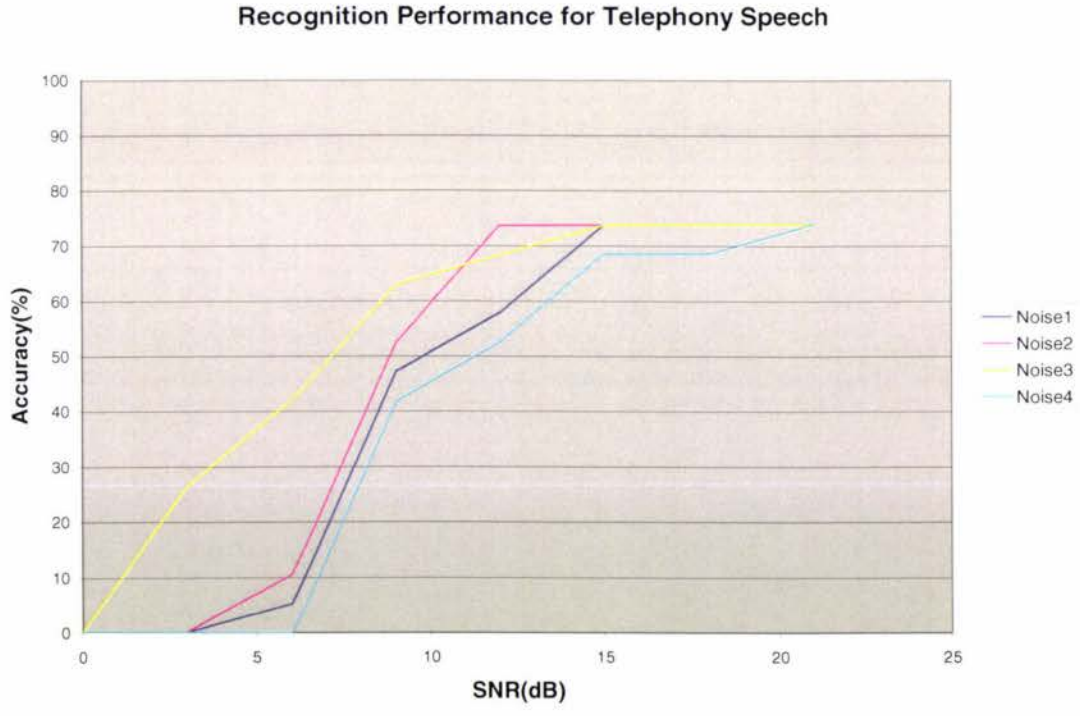

Figure 6.9: Recognition accuracy graph for Speaker 1 using bandpass filtered Speech to simulate telephony quality speech

6.3 Improving Recognition Accuracy

There are several signal processing techniques which we can apply to the speech signal to improve the SNR to give more accurate recognition results. Provided constant noise from stationery source techniques, such as spectral subtraction and adaptive noise cancellation can be utilised.

In Spectral Subtraction techniques the speech signal is analysed to determine when noise alone is present and when speech is present. At periods when there is no speech present, the noise characteristics are determined. The system then tries to subtract out noise from speech using those noise characteristics. This technique is commonly used in mobile phones.

Adaptive Noise Cancellation techniques [28] rely on the use of multiple microphones, most commonly two. One microphone is used to pick up the combined speech and noise signal and the other to pick up the noise. A filter with varying coefficients is then applied to the noisy signal. The coefficients of the filter are determined by the level of noise picked up from the second microphone.

Another commonly used noise reduction technique that will be employed in the Massey SmartHouse is Beamforming microphones. The use of beamforming microphone arrays in speech processing is not new, however, recent advances in DSP integrated circuits has made it more affordable and available. Beamforming techniques utilizes an array of microphones and by considering sound propagation principles, it differentiates between noise and speech. The idea works as follows:

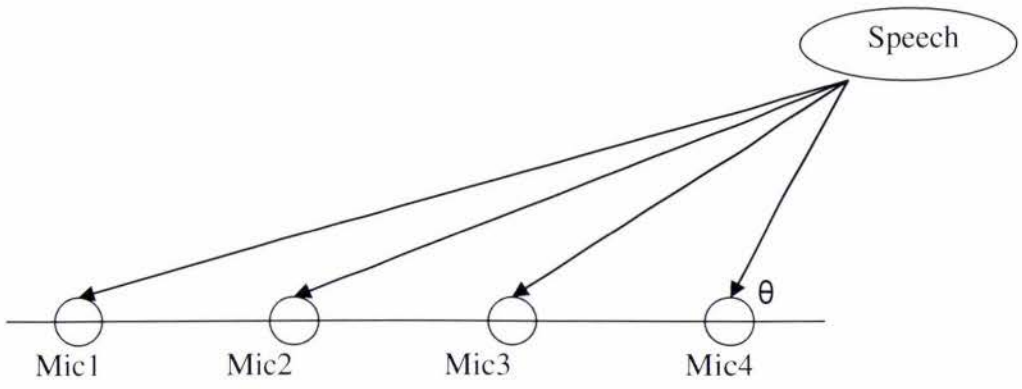

Fig 6.10: Beamformer microphone array

Consider a number of microphones placed in a straight line with a fixed common distance between each one. An approaching sound wave at angle θ to the array index will arrive at one of the microphones before the others. The time delay between the wave arriving at the first microphone and second microphone is then calculated and hence angle θ is worked out which indicates the direction of speech. An increased SNR can be obtained by summing the individual microphone signals (with an appropriate delay added to compensate for the time delay). The improved SNR is a result of signals from that specific direction adding constructively and signals form other directions adding destructively. This implementation is referred to as Delay & Sum Beamformer.

The above implementation combined with an adaptive noise cancellation system can provide improved SNR. The sum of the signals provide the primary speech reference while the difference of signals provide the noise reference. This implementation of beamoformers is known as a Griffiths-Jim beamformer. Typical SNR improvements

for a 4-microphone Delay & Sum Beamformer is approximately 4dB while a Griffiths-Jim beamformer can expect to give a 5 to 8dB improvement.

Another implementation of beamformers which also works with non stationary noise sources is to compute which direction the main signal is coming from and amplify the output of that particular microphone, while suppressing the output of all others. This implementation of beam formers can provide 15-20 dB improvement in SR [29].

Chapter 7: Conclusion and Future Work

7.1 Conclusion

The purpose of this project was to develop a voice recognition system, that can be used in Massey University Smarthouse to respond to the occupants needs and desires simply by taking their voice requests and transforming them into actions. The system acts as a hub that services and delegates all voice requests to other control systems within the house.

The thesis discussed Massey Smarthouse, its purpose and the different projects involved. The thesis then discussed the background technologies that were utilised by the system. These included the concept of automated computer speech recognition, its challenges and how it works. Other technologies were also discussed including speech synthesis, common industry standards associated with speech implementation, and the impact of speech technologies in the business world.

A description of the system implantation and how it was designed was discussed. The final product was a software solution that preformed the requirements stated above. The system was able to successfully recognise commands from different speakers with a high level of accuracy without any training when close range microphones were used in a relatively low noise environment. The system provides a framework that handles the lower level configuration and implantation details to allow a flexible speech interface model to be built on top. The system was built to be as user friendly as possible, all buttons and fields are self-explanatory. The users are able to control

most parameters of the program at run time. The Program is easy to install and integrate into the smarthouse main computer.

An attempt has been made to extend the system's communication method to also include telephony. A program has been designed to allow for input of voice commands over the PSTN, although this worked as a stand alone application, combining it with the standard application was not successful due to some of the limitations of the API used which was mainly designed for Win32 applications.

A more detailed study of the system's recognition accuracy when operating in a noisy environment was presented. The system was tested with different types of noise and signal to noise ratios and it was found that above 90% accuracy can be achieved with an average SNR of 9db or higher. Alternative methods for improving recognition accuracy via means of hardware and software signal processing were also discussed.

Speech technology has been developing rapidly. However, it is quite evident that it is yet to reach maturity. The development is moving forward steadily, and with increasing computational performance of modern personal computers the technology may advance to a level where it will be one of the standard methods of human machine interactions.

7.2 Future Work

There are a number of improvements that could be made to the system. One of the main drawbacks of the system is that it can only serve one user at a time, this was acceptable as the Smarthouse is still in development and testing stages. Once the Smarthouse is ported into the real world it needs to respond to multiple users simultaneously. This can only be achieved by allowing the application to multithread, which is unfortunately not possible with the programming language used in this application. Future versions of the system need to be migrated to a .NET language and the Microsoft Speech Applications Software Development Kit (the .NET version of SAPI). There are two advantages to this approach:

1. The ability to multithread the application
2. Ease of integration of telephony services.

To allow the system to serve multiple users in different rooms simultaneously, the expert system must also be changed to handle concurrent requests. Another required step is to integrate the location and positioning system with the speech system so that each location in the house is associated with a running thread.

Minor improvements that can be made to the current application include creating a graphical user interface for speech dialogue editing; this would allow advanced users or administrators to customize the system easily. Another improvement would be to increase the reliability, robustness and error recovery of the application by improving upon the code and the speech interface.

References

1. Massey University Smart House Project webpage. [cited 2005 19 June]; Available from: http://smarthouse.massey.ac.nz/

2. Minker, W., *Speech and human-machine dialog*. English language ed. 2004, Boston: Kluwer Academic Publishers.

3. Makhoul, J. and R. Schwartz, *State of the art in continuous speech recognition.* Voice communication between humans and machines, 1994: p. 165-198.

4. McTear, M., *Spoken dialogue technology, towards the conversational user interface*. 2004, New York: Springer.

5. Furui, S., *Digital speech processing, synthesis, and recognition*. 2001, New York: Marcel Dekker.

6. *Microsoft Speech SDK 5.1 Help Documentation*. [cited 2005 23 June]; Available from: http://www.microsoft.com/speech/techinfo/apioverview/.

7. *Microsoft Platform SDK documentation*. [cited 2004 15 August]; Available from: http://msdn.microsoft.com/library/en-us/sdkintro/sdkintro/devdoc_platform_software_development_kit_start_page.asp.

8. Rabiner, L.R., *A tutorial on hidden Markov models and selected applications in speech recognition.* Proceedings of the IEEE, 1989. **77**(2): p. 257-286.

9. *W3C Extensible Markup Language Overview*. [cited 2005 17 May]; Available from: http://www.w3.org/XML/.

10. *Duke University Smart House Project*. [cited 2005 21 Feb]; Available from: http://delta.pratt.duke.edu/.

11. *Massachuessettes Institute of Technology(MIT) Oxygen Project*. [cited 2005 21 Feb]; Available from: http://oxygen.lcs.mit.edu/index.html.

12. *Univeristy of Florida Smart House Project.* [cited 2005 21 Feb]; Available from: http://www.rerc.ufl.edu/.

13. Larson, J.A. *State of Speech Standards.* 2003 [cited 2005 13 April]; Available from: http://www.larson-tech.com/Writings/Std.htm.

14. *VoiceXML Standards and Recommendation.* [cited 2005 17 May]; Available from: http://www.w3.org/Voice/.

15. *Speech Application Language Tags Standards and Recommendation.* [cited 2005 17 May]; Available from: http://www.saltforum.org/.

16. Potter, S. and J.A. Larson, *VoiceXML and SALT. How are they different , and why* in *Speech Technology Magazine* 2002.

17. *Harris Interactive, Interactive Speech Satisfaction Study.* [cited 2005 4 April]; Available from: http://www.nuance.com/assets/pdf/speech_study.pdf.

18. Delogu, C., et al. *A comparison between DTMF and ASR IVR services through objective and subjective evaluation.* in *Interactive Voice Technology for Telecommunications Applications, 1998. IVTTA '98. Proceedings. 1998 IEEE 4th Workshop.* 1998.

19. Jahnke, J.H., M. d'Entremont, and J. Stier, *Facilitating the programming of the smart home.* Wireless Communications, IEEE [see also IEEE Personal Communications], 2002. **9**(6): p. 70-76.

20. Weinschenk, S. and D.T. Barker, *Designing effective speech interfaces.* 2000, New York: Wiley.

21. Ambler, S.W. *User Interface Design:Tips and Techniques.* 2000 [cited 2005 14 April]; Available from: www.knowlesys.com/technology/design/user_interface_design_tips_and_techniques.pdf.

22. Microsoft. *Official Guidelines for User Interface Developers and Designers.* 1995 [cited 2005 14 April]; Available from: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwue/html/part2.asp.

23. Marin, R., et al. *Automatic speech recognition to teleoperate a robot via Web.* in *Intelligent Robots and System, 2002. IEEE/RSJ International Conference on.* 2002.

24. Sebastian, D. and P.C. Pedersen. *Development of a field-deployable voice-activated ultrasound scanner system.* in *Bioengineering Conference, 2004. Proceedings of the IEEE 30th Annual Northeast.* 2004.

25. Morris, A.C., V. Maier, and P. Green, *From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition.* ICSLP, 2004.

26. Damnati, G. *Evaluating speech recognition in the context of a spoken dialogue system: critical error rate.* in *Automatic Speech Recognition and Understanding, 2001. ASRU '01. IEEE Workshop on.* 2001.

27. *Matlab 7 Software Documentation.* 2004, Mathworks Inc.

28. Dhanjal, S. *Noise cancellation techniques in speech signals (DLPNC).* in *Electrical and Computer Engineering, 1993. Canadian Conference on.* 1993.

29. Wenger, M.P. *Noise Rejection - The Essence of Good Speech Recognition.* [cited 2005 12 Feb]; Available from: http://www.knowlesacoustics.com/images/pdf/white/NoiseRejection.pdf.

Bibliography

"Microsoft Developer Network - MSDN Library Network Help files." Last Retrieved 23 June 2005, from http://msdn.microsoft.com.

Balentine, B. (2001). How to build a speech recognition application : a style guide for telephony dialogues. San Ramon, Calif., EIG Press.

Becchetti, C. (1999). Speech recognition : theory and C++ implementation. Chichester ; New York, Wiley.

Chen, C.-P. (2004). Noise robustness in automatic speech recognition, University of Washington: 124.

Comerford, L., D. Frank, et al. (2001). The IBM Personal Speech Assistant. Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on.

Contolini, M., K. Stolmenov, et al. (2004). Voice technologies for telephony services. Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE.

Deng, Y. and S. Hu (2004). "Development and application of a sound telephone system based on TTS/SR engine and TAPI." Journal of Wuhan Automotive Polytechnic University **26**(2): 9-11.

Evans, J. R., W. A. Tjoland, et al. (2000). "Achieving a hands-free computer interface using voice recognition and speech synthesis [for Windows-based ATE]." Aerospace and Electronic Systems Magazine, IEEE **15**(1): 14-16.

Kotelly, B. (2003). The art and business of speech recognition : creating the noble voice. Boston, MA, Addison-Wesley.

Makhoul, J. and R. Schwartz (1994). "State of the art in continuous speech recognition." Voice communication between humans and machines: 165-198.

Shi, Y. (2004). An investigation of grammar design in natural-language speech-recognition, University of Windsor (Canada): 202.

Simon, C. (2002). "COM objects, C#, and the Microsoft Speech API." Windows Developers Journal **13**(9): 18-26.

Srinwasan, S. and E. Brown (2002). "Is speech recognition becoming mainstream?" Computer **35**(4): 38-41.

Starner, T. E. (2002). "The role of speech input in wearable computing." Pervasive Computing, IEEE **1**(3): 89-93.

Stefanov, D. H., Z. Bien, et al. (2004). "The smart house for older persons and persons with physical disabilities: structure, technology arrangements, and perspectives." Neural Systems and Rehabilitation Engineering, IEEE Transactions on [see also IEEE Trans. on Rehabilitation Engineering] **12**(2): 228-250.

Yamada, T., S. Nakamura, et al. (2002). "Distant-talking speech recognition based on a 3-D Viterbi search using a microphone array." Speech and Audio Processing, IEEE Transactions on **10**(2): 48-56.

Appendices

Appendix A: Matlab Script Used During Testing Process

```
speaker = 'maryan';
noise_type = 'noise4';
directory= 'C:\temp\';
noise_file= strcat(directory, noise_type);
noise_file= strcat(noise_file, '.wav');
speech_file=strcat(directory, speaker);
speech_file=strcat(speech_file,'.wav');

SNR_factor= [1 0.5 0.25 0.125 0.0625 0.03125 0.015625 0.0078125];
SNR_DB=     [0 3 6 9 12 15 18 21]
signal_threshhold = 0.16;

[speech,FS,BS]   = wavread(speech_file); % Read in Orginal Speech file

noise            = wavread(noise_file); % Read in noise file

%time of whole file  =
file_length = length(speech)/FS;

%set window length  = 5 ms
window_length=round(50*(FS/1000));

%rms power of whole file =
rms = sqrt(mean(speech.*speech));

wbh=waitbar(0,'Processing...');
SNR_factor_index = 1;
while (SNR_factor_index <= length(SNR_factor)),
    i =1;
    while i < (length(speech)- window_length),
        window_wave = wavread(speech_file, [i (i+window_length)]);
        if (sqrt(mean(window_wave.*window_wave))) >
(rms*signal_threshhold),
            wave_with_noise = window_wave +
((SNR_factor(SNR_factor_index)) * wavread(noise_file, [i
(i+window_length)]));
        else
           wave_with_noise = window_wave;
        end
        %append this to the main wave file
        if i == 1,
           final_wave =  wave_with_noise;
        else
           final_wave = [final_wave ; wave_with_noise];
        end

        i = i + window_length;

    end
```

```
        combined_file = strcat(directory,speaker);
        combined_file = strcat(combined_file, '\');
        combined_file = strcat(combined_file, speaker);
        combined_file = strcat(combined_file, '_');
        combined_file = strcat(combined_file, noise_type);
        combined_file = strcat(combined_file, '_');
         combined_file = strcat(combined_file,
         num2str(SNR_DB(SNR_factor_index)));
        combined_file = strcat(combined_file, 'dB');
        combined_file = strcat(combined_file, '.wav');

        %normalize the output file to avoid clipping
        final_wave = 0.99*final_wave/max(abs(final_wave));

        wavwrite(final_wave,41000,16,combined_file);
        waitbar((SNR_factor_index/4) *100);
        SNR_factor_index = SNR_factor_index + 1;

    end

    close(wbh);
```

Appendix B: Grammar File Used During Testing Process

```
- <GRAMMAR LANGID="409">
- <DEFINE>

  </DEFINE>
- <RULE NAME="PCNAME" TOPLEVEL="ACTIVE">
  <P>+jeeves</P>
  <O>-please</O>
  <RULEREF NAME="Commands" />
  </RULE>
- <RULE NAME=" Commands">
- <LN PROPNAME="Command" PROPID="RANK">
  <PN VAL="1">turn lights on</PN>
  <PN VAL="2">read me the news</PN>
  <PN VAL="3">tell me the weather forecast</PN>
  <PN VAL="4">activate guest mode</PN>
  <PN VAL="5">activate alarm</PN>
  <PN VAL="6">switch to day mode</PN>
  <PN VAL="7">switch to Sleep mode</PN>
  <PN VAL="8">change TV channel</PN>
  <PN VAL="9">remove food from the oven</PN>
  <PN VAL="10">dial the ambulance for me</PN>
  <PN VAL="11">give me temperature reading</PN>
  <PN VAL="12">dim lights</PN>
  <PN VAL="13">record conversation</PN>
  <PN VAL="14">what food is in the fridge?</PN>
  <PN VAL="15">what time is it</PN>
  <PN VAL="16">who is in the house</PN>
  <PN VAL="17">blow up the house</PN>
  <PN VAL="18">Help</PN>
  <PN VAL="19">play my favourite music</PN>
  <PN VAL="21">call my mother</PN>
  <PN VAL="22">call my son</PN>
  <PN VAL="23">clean the floors</PN>
  <PN VAL="24">what is my blood pressure</PN>
  <PN VAL="25">retreive my health information</PN>
  <PN VAL="26">transfer call to bedroom one</PN>
  <PN VAL="27">dial the police for me</PN>
  <PN VAL="28">read me a novel</PN>
  <PN VAL="29">open garage door</PN>
  <PN VAL="30">warm up the car</PN>
  <PN VAL="31">order pizzas</PN>
  <PN VAL="32">book movie tickets</PN>
  <PN VAL="33">water the plants</PN>
  <PN VAL="34">remove rubbish</PN>
  <PN VAL="35">lock all doors</PN>
  <PN VAL="36">open windows</PN>
  <PN VAL="37">adjust humidity level</PN>
  <PN VAL="38">play my messages</PN>
```

```
<PN VAL="39">check my mailbox</PN>
<PN VAL="40">prepare coffee</PN>
  </LN>
  </RULE>
  </GRAMMAR>
```