

SPEECH ENABLING SYSTEM

(VOICE DICTATION)



P-566

A project work done at
Rishinet Pvt. Ltd., Bangalore.

Submitted in partial fulfilment of the requirements
for the award of the Degree of
MASTER OF COMPUTER APPLICATIONS
of Bharathiar University

Submitted by

I. SEKAR

Reg. No : 9838M0518

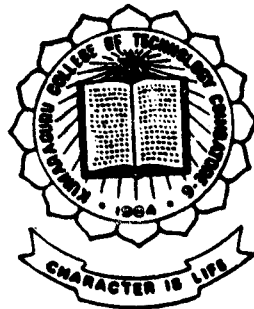
Under the guidance of

External Guide

Mr. R. Sureshramachandran B.E.,
Project Manager,
Rishinet Pvt. Ltd,
Domlur Layout,
Bangalore-560071

Internal Guide

Dr. S. Thangasamy B.E.(Hons) Ph.D
Head of the Department,
Dept of Computer Science & Engg,
Kumaraguru College of Technology,
Coimbatore-641006



2000-2001

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Kumaraguru College of Technology

COIMBATORE, Tamil Nadu-641006

APRIL 2001

CERTIFICATE

This is to certify that the project work entitled

SPEECH ENABLING SYSTEM

Submitted to the

Department of Computer Science and Engineering

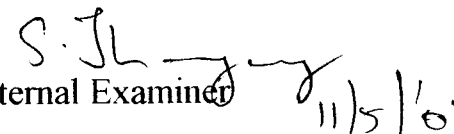
Kumaraguru College of Technology

in partial fulfillment of the requirements for the award of the degree of Master of Computer applications is a record of original work done by Mr.I.Sekar, Reg.No. 9838M0518 during his period of study in the Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore under my supervision and this project work has not formed the basis of award of any Degree/Diploma Associateship / Fellowship or similar title to any candidate of any University.


Professor and Head 8/5/01


Staff-in-charge 8/5/01

Submitted to University Examination held on 11-05-2001


Internal Examiner 11/5/01


External Examiner 11/5/01



C E R T I F I C A T E

This is to certify that the project work entitled " Speech Enabling System " submitted by Sekar I. Roll No.9838M0518, Student of the final year MCA (Master of Computer Applications) of Kumaraguru college of Technology (Affiliated to Bharathiar University) Coimbatore is a bonafide work carried out by him under our guidance, from December 2000 to March 2001 at Rishinet Pvt., Ltd, Bangalore-560 071. This is required for the fulfillment of the degree of Master of Computer Applications of Bharathiar University for year 2000-2001. The student will not be allowed to take the source code outside the organization.

During this period we found that, he to be a good learner and hard working. We wish him the best in all his future endeavors.

Mr. H R K Ravi
Managing Director.

DECLARATION

I hereby declare that the project entitled '**Speech Enabling System**', submitted to **Bharathiar University** as the project work of **Master of Computer Applications Degree**, is a record of original work done by me under the supervision and guidance of **Mr.R.Suresh Ramachandran, B.E.**, Rishinet Pvt Ltd., Bangalore and **Prof. Dr.S.Thangasamy B.E.(Hons), Ph.D.**, Professor and Head of the Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore and this project work has not found the basis for the award of any Degree/Diploma/ Associateship/Fellowship or similar title to any candidate of any university.

Place: Coimbatore

Date: 8/5/2001

I. Sekar
I.Sekar

Counter Signed By

Prof. Dr.S.Thangasamy B.E.(Hons), Ph.D.,
Head of the Department of Computer Science and Engineering,
Kumaraguru College of Technology,
Coimbatore.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who are responsible for the successful completion of this project.

I wish to express my sincere and heartfelt gratitude to our esteemed Principal **Dr. K.K.Padmanabhan, B.sc., (Engg), M.Tech., Ph.d.**, Kumaraguru College of Technology for giving us the needed encouragement in starting this project and carrying it out successfully.

My hearty thanks and a deep sense of gratitude to my guide **Prof.Dr.S.Thangasamy B.E.(Hons), Ph.D.**, Professor and Head of the Department of Computer Science and Engineering for his benevolent attitude to push up this project to its position of success

It gives me great pleasure to express my profound gratitude to **Mr.R.Suresh Ramachandran, B.E.**, Project Manager, Rishinet Pvt Ltd., Bangalore, for his invaluable and inspiring guidance throughout the progress of this project, without which this could not have been completed.

TABLE OF CONTENTS

S.No	CONTENTS	PAGE NO
	Synopsis	
1	Introduction	1
1.1	Project overview	2
1.2	Organization profile	3
2	System Study and Analysis	9
2.1	Existing system - limitations	9
2.2	Proposed system	11
2.3	Requirements on new system	12
2.4	User characteristics	17
3	Programming environment	19
3.1	Hardware configuration.	19
3.2	Description of software tools used	22
4	System Design & Development	29
4.1	I/P Design	29
4.2	O/P Design	31
4.3	Process Design	32
5	System Implementation & Testing	36
5.1	System Implementation	36
5.2	System Testing	67
5.3	Refinements based on feedback	69
6	Conclusion	73
7	Scope for future development	74
	Bibliography	77
	Appendices	79

SYNOPSIS

The objective of this project is to develop a speech enabling system for the computer to understand what the user speaks. It completely removes the need of using the keyboard & mouse. The word spoken by the speaker received through a microphone using a SOUND BLASTER 32 KIT. The analog input voice given is converted into its equivalent digital data stored as WAV file. The process of taking a user's speech and storing it in a file called speech dictionary. This word is compared with pronunciation dictionary. If the word matches it will be converted into text. The system is implemented using WINDOWS NT as operating system and VISUAL C++ Ver 6.0 as the language for development.

INTRODUCTION

The communication between user and computer is generally through input devices viz. keyboard, mouse etc. and output devices viz., CRT display, printers etc., But for a human, the natural and spontaneous way of communication is through his/her voice. So it will be quite easier for the user to communicate with the computer system through his/her voice.

Generally in a windows application the short cut keys and mouse are used to operate it. Each time we have to use the keyboard or move the mouse to that location and click the required command. Even while typing the user has to be familiar with typing and there are a lot of mistakes like transposition error etc. and also continuous typing affects the human ergonomics. Although such types of products are commercially available, this is an attempt to simulate one of them. Speech Enabling System basically involves the concept of Speech Recognition. The user access and control the system through his voice.

1.1 PROJECT OVERVIEW

Speech recognition offers certain users the best way to interact with a computer and promises to be the dominant form of human-computer interaction in the near future. Recent advances in software speech recognition engines and hardware performance are accelerating the development and acceptance of the technology. Most people are familiar with speech recognition applications based on dictation grammars, also known as continuous speech recognition. These applications require a large commitment from the user, who has to spend time training the computer and learning to speak in a consistent manner to assure a high degree of accuracy. This is too much of a commitment for the average user, who just wants to sit down and start using a product. Users of this technology tend to be those who must use it or are highly motivated to get it working for some other reason, like people with various physical disabilities. However, there are other forms of speech recognition based on different grammars. These grammars represent short-run solutions that can be used by more general audiences.

1.2 ORGANIZATIONAL PROFILE

Rishinet Pvt. Ltd.

Rishinet Pvt Ltd. is an entity whose prime focus is on providing quality e-Solutions swiftly. In a business environment where time-to-market is everything, we inject our rich knowledge base and well owned expertise to secure our diverse client base with enduring solutions. The company commenced operations in 1996 with a team of inspired professionals who brought with them, a wealth of IT related experience and a single unified objective : to make the company a leading solution provider, with a qualitative difference in the web space. Today, Rishinet is what it is solely because of the firm commitment of its people and their unending focus on quality.

At Rishinet, the very same intrinsic qualities are reinforced. The focus is on innovation and solutions using flexible but definitive processes. The approach is to average learning's from our experience into every new opportunity and bring to it, radically new perspectives and solutions. Our corporate culture is built on fundamentally simple principles with a focus on building our team of quality people with professional values. All roe ensure that Rishinet consistently delivers more than what is expected.

Infrastructure

A sprawling complex that houses a carpet area of 6,000 sq. feet with the capacity to seat 100 software professionals is not what we view as our only infrastructure. We take into account inspiration levels, knowledge sharing technologies and personal dedication as key factors to leveraging the best out of our infrastructure.

Our development facility stationed at Bangalore offers facilities such as videoconferencing, state-of-the-art multi-platform servers and connectivity that ensures swift working paces. A 24 hour global network service further reinforces our commitment to meeting our client requirements instantaneously and optimally.

Our facility has been awarded ISO 9001 certification by KEMA, Netherlands and has been re-certified recently. We presently use our considerable quality and project management experience in setting up processes and systems towards securing enhanced quality standard certifications.

To effectively manage our overseas clientele and provide localized support, we have set up technical support and marketing offices in USA and UK. We are also working through partner organizations in other countries, to further reinforce our presence in global markets.



Quality

At Rishinet we believe in adhering to stringent delivery schedules and quality systems that swiftly secure our clients with world class solutions.

The basic structure of the Rishinet Quality System Include

Software practices and methodologies that meet the latest international software engineering standards (ISO/SEI). Compatible guidelines in conformance with ISO 9001 Standards.

A Quality Focus with a difference

A key element of our Quality System is its continuous process enhancement of each functions and methodology followed within different lines of our business. We follow Deming's PDCA cycle (Plan, Do, Check, Act) to ensure that projects are executed in strict compliance with both, the short-term as well as the long-term objectives of our clients.

Expertise Our core competency lies in swiftly understanding client requirements and their business objectives. The focus is always on delivering quality solutions and our highly competent team draws inspiration by looking upon each project as an opportunity-not a task. Rishinet software solutions and consulting

services include custom design programming while building in modifications to existing software and implementation services

A quick view of our expertise are

Rishinet provides international standards of quality across all our services. We believe in partnering with our customers in order to assist them in fulfilling their business objectives.

We offer technical expertise across specialized business lines ensuring timely delivery and at competent costs.

Rishinet brings together a balanced mix of people, technology and the clarity to focus on client requirements.

We adopt a flexible approach in modeling our business to meet customer needs in an optimized manner

Business Lines

A diverse yet highly evolved expertise base is what makes Rishinet the dynamic entity that it is. Our offerings have been demarcated into four broad areas that include Software Services, Co-Product Development, Professional Services and Smart Sourcing.

Software Services

These services largely comprise of Consulting Services Architecture, Design, Development & Enhancement Services, Web Enabling Services, Porting & Re-Engineering Services, Sustainance Services and Testing Services.

Co-Product Development

Rishinet is in a position to offer a range of product development services that call upon our versatile mix of skills. Our product development stream enables us to offer the following services.

Detailing of Specifications

- Conducting a technical feasibility study.
- Co-architecting, engaging in design and development.
- Planning and implementing the feature enhancement of products.
- Building in localization of products

Professional Services

In an industry where technical competence takes precedence over all else, we bring to every solution and optimal mix of

manpower expertise and technology. Rishinet has put in place a stringent process of selecting competent software professionals who are assigned to projects according to their skill sets and core competencies.

Smart Sourcing

Rishinet assists diverse international client base in offshore outsourcing of technically skilled manpower across a varied skill set base and domain industry knowledge.

SYSTEM STUDY AND ANALYSIS

2.1 EXISTING SYSTEM

The communication between user and computer is generally through input devices viz. keyboard, mouse etc. and output devices viz., CRT display, printers etc., Generally in a windows application the keyboard keys and mouse are used to operate it. The entire process involves, the existing system does not recognize our voice. The existing system we can access the system and control the system through only keyboard or mouse.

The input given to the system generally via the keyboard. The output taken by CRT display, printers. The existing system is not flexible to make further study or analysis of the data and is not user friendly. Hence it takes longer time and larger effort.

2.2 LIMITATION OF EXISTING SYSTEM

Drawbacks in the existing system are

- The existing system does not recognize our voice. So that It does not access our system without using Keyboard and mouse.
- Each time we have to use the keyboard or move the mouse to that location and click the required command. Even while typing the user has to be familiar with typing and there are a lot of

mistakes like transposition error etc and also continues typing affects the human ergonomics.

- The existing system does not have the advanced features like Speech to Text, Voice Command, i.e To access and control the system Through voice.

- Even while typing the user has to be familiar with typing and there are a lot of mistakes like transposition error etc and also continued typing affects the human ergonomics.

- It take more time.

- Given less amount of input to the system per second

PROPOSED SYSTEM

The steps involved in these procedures

- The data available on the Microsoft.com. The Microsoft Speech SDK 5.0 downloaded from the Microsoft.com.
- The word spoken by the user is received by the microphone is converted into digital wave file by the Speech SDK.
- The process of taking an user's speech and storing it in a file called speech Dictionary. The Speech SDK 5.0 is convert the digital values is compared with speech dictionary.
- Using the inputs from the user this process will check the data dictionary to find if it is able to translate the message. If it cannot or if it is unable to determine what the spoken word is a best guess (20-80% confidence as opposed to 80-100% for 'correct' words) will be made which will be displayed outlined on the screen. If the process absolutely can not guess (0-20% confidence) then an empty box will be displayed.

This Speech Recognition engine will be used in Banking Product of Rishinet Company.

REQUIREMENTS FOR NEW SYSTEM

The following requirements needed for new system

- SAPI Hardware
- General Hardware Requirements
- Software Requirements-Operating Systems and Speech Engines
- Special Hardware Requirements-Sound Cards, Microphones, and Speakers

SAPI Hardware

Speech systems can be resource intensive. It is especially important that SR engines have enough RAM and disk space to respond quickly to user requests.

For these reasons, it is important to establish clear hardware and software requirements when designing and implementing your speech-aware and speech-enabled applications. Not all pcs will have the memory, disk space, and hardware needed to properly implement SR and TTS services.

General hardware, including processor speed and RAM memory.

Software, including operating system and SR/TTS engines.

Special hardware, including sound cards, microphones, speakers, and headphones.

General Hardware Requirements

Speech systems can tax processor and RAM resources. SR services require varying levels of resources depending on the type of SR engine installed and the level of services implemented. TTS engine requirements are rather stable, but also depend on the TTS engine installed.

SR Processor and Memory Requirements

In general, SR systems that implement command and control services will only need an additional 1MB of RAM (not counting the application's RAM requirement). Dictation services should get at least another 8MB of RAM-preferably more. The type of speech sampling, analysis, and size of recognition vocabulary all affect the minimal resource requirements.

These memory requirements are in addition to the requirements of the operating system and any loaded applications. The minimal Windows 95 memory model should be 12MB. Recommended RAM is 16MB and 24MB is preferred.

Software Requirements

Software Requirements-Operating Systems and Speech Engines. The Microsoft Speech API can only be implemented on Windows 32-bit operating systems. This means you'll need Windows 95 or Windows NT 3.5 or greater on the workstation.

The most important software requirements for implementing speech services are the SR and TTS engines. An SR/TTS engine is the back-end processing module in the SAPI model.

The new wave of multimedia pcs usually has SR/TTS engines as part of their initial software package. For existing pcs, most sound cards now ship with SR/TTS engines.

Special Hardware Requirements

Special Hardware Requirements :

1. Sound Cards,
2. Microphones, and
3. Speakers

Complete speech-capable workstations need three additional pieces of hardware

A sound card for audio reproduction

Speakers for audio playback

A microphone for audio input

Just about any sound card can support SR/TTS engines. Any of the major vendors' cards are acceptable, including Sound Blaster and its compatibles, Media Vision, ESS technology, and others. Any card that is compatible with Microsoft's Windows Sound System is also acceptable.

A few speech-recognition engines still need a DSP (digital signal processor) card. While it may be preferable to work with newer cards that do not require DSP handling, there are advantages to using DSP technology. DSP cards handle some of the computational work of interpreting speech input. This can actually reduce the resource requirements for providing SR services. In systems where speech is a vital source of process input, DSP cards can noticeably boost performance.

SR engines require the use of a microphone for audio input. This is usually handled by a directional microphone mounted on the pc base. Other options include the use of a lavalier microphone draped around the neck, or a headset microphone that includes headphones. Depending on the audio card installed, you may also be able to use a telephone handset for input.

Most multimedia systems ship with a suitable microphone built into the pc or as an external device that plugs into the sound card.

If the system will be used in a noisy environment, close-talk microphones should be used. This will reduce extraneous noise and improve the recognition capabilities of the SR engine.

Speakers or headphones are needed to play back TTS output. In private office spaces, free-standing speakers provide the best sound reproduction and fewest dangers of ear damage through high-levels of playback. However, in larger offices, or in areas where the playback can disturb others, headphones are preferred.

2.4 USER CHARACTERISTICS

Initially users of this system will have to acquire the knowledge of how this system works. Further the user will have to customize the system by pronouncing the language through a micro phone.

FEASIBILITY STUDY

Feasibility Assessment

For any project it is necessary to have a feasibility study i.e., the likelihood that the system will be useful for the organization.

Tests of feasibility all equally important are studies.

- Operational feasibility
- Technical feasibility
- Economic feasibility

Operational feasibility

A project is feasible if the project meets the organizations operating requirement the questions are whether there are Major barriers to implementation software. As far this project is concerned. It is operationally feasible one.

Technical Feasibility

Technical feasibility is concerned with such questions .As whether the necessary technology exists to do what is planned. Does the Proposed equipment have the technical capacity to hold the data required etc. as far as the project is concerned it is technically feasible. Various types of hardware and software are available. Consequently technology was not a constraint to system development.

At present the capacity of the secondary is 4.3GB hence Memory is not a constraint. The proposed system is developed using structural techniques.

Accuracy, ease of access, security and reliability is ensured.

Financial & Economic Feasibility

Any system developed must be good invested for the organization. In Economic feasibility we have to consider whether the resources and human power applied for the purpose makes enough returns. For this management System some human power is needed for data entry and since the system Provide very fast and accurate access of data it is economic feasible.

PROGRAMMING ENVIRONMENT

3.1 HARDWARE CONFIGURATION

To develop a Speech Enabling System used following Hardware Specification

PROCESSOR	Pentium II
PROCESSOR SPEED	233 MHz
RAM	32 MB
SOUND CARD	Yamaha

Input Devices

Micro Phones

Speakers

Head phones

SR Engine 16 MB RAM

SAPI Hardware

Speech systems can be resource intensive. It is especially important that SR engines have enough RAM and disk space to respond quickly to user requests.

General hardware

- sound cards
- microphones
- speakers, and
- headphones

A sound card for audio reproduction.

Speakers for audio playback.

A microphone for audio input.

General Hardware Requirements

Speech systems can tax processor and RAM resources.

SR Processor and Memory Requirements

In general, SR systems that implement command and control services will only need an additional 1MB of RAM (not counting the application's RAM requirement). Dictation services should get at least another 8MB of RAM-preferably more. The type of speech sampling, analysis, and size of recognition vocabulary all affect the minimal resource requirements.

SR engines require the use of a microphone for audio input. This is usually handled by a directional microphone mounted on the pc base. Other options include the use of a lavalier microphone

draped around the neck, or a headset microphone that includes headphones. Depending on the audio card installed, you may also be able to use a telephone handset for input.

Most multimedia systems ship with a suitable microphone built into the pc or as an external device that plugs into the sound card.

The quality of the audio input is one of the most important factors in successful implementation of speech services on a pc. If the system will be used in a noisy environment, close-talk microphones should be used. This will reduce extraneous noise and improve the recognition capabilities of the SR engine.

Speakers or headphones are needed to play back TTS output. In private office spaces, free-standing speakers provide the best sound reproduction and fewest dangers of ear damage through high-levels of playback. However, in larger offices, or in areas where the playback can disturb others, headphones are preferred.

3.2 DESCRIPTION OF SOFTWARE TOOLS USED

MS-SPEECH SDK	:	Microsoft Speech Software Development Kit
PSDK	:	Platform Software Development Kit
VC	:	Visual C++
TTS	:	Text to Speech Engine
SRE	:	Speech Recognition Engine
DSR	:	Direct Speech Recognition
SAPI	:	Speech Application Programming Interface

SAPI

SAPI 5.0 gives developers a rich set of speech services for building high-performance applications that run on desktop, mobile, and server platforms. The speech services in SAPI 5.0 are enabled through the Component Object Model (COM), the Windows standard specification for software interoperability. The SAPI 5.0 speech services are compatible with and can be leveraged by many third-party speech recognition engines and synthesizers, programming languages and tools.

SAPI 5.0 is a software layer that allows speech-enabled applications to communicate with both speech recognition and TTS engines. SAPI 5.0 includes an API and a Device Driver Interface (DDI). Applications communicate with SAPI 5.0 via the API layer; speech engines communicate via the DDI layer.

With SAPI 5.0, a speech-enabled application and a speech recognition engine do not communicate directly with each other: all communication is done via SAPI. In addition, SAPI takes responsibility for a number of functions in a speech system, such as

- Controlling audio input, whether from a microphone, files, or custom audio source.
- Converting audio data to a valid speech engine format.
- Loading grammar files, whether created dynamically or from memory, a URL, or a file. With SAPI 5.0, developers can build a grammar that includes a variety of natural-language phrases that might be spoken by users.
- Resolving grammar imports and editing.
- Compiling standard SAPI XML grammar format, converting custom grammar formats, and parsing semantic tags in results.
- Sharing speech recognition across multiple applications using the shared engine; all coordination between the shared engine and applications is performed by SAPI.

- Returning results and other information to the application and interacting with its message loop or other notification method. This return of results allows an engine to have a simple threading model because SAPI 5.0 performs much of the thread handling.

- Storing audio and serializing results for later analysis.

Ensuring that applications do not cause errors, preventing applications from calling the speech engine with invalid parameters, and dealing with applications that hang or crash.

The speech recognition engine is responsible for :

- Using SAPI grammar interfaces and loading dictation.

- Performing speech recognition.

- Polling SAPI to learn about grammar and state changes.

- Generating recognition's and other events that provide information to the application.

The Speech SDK version 5.0 contains a new version of SAPI, new Speech Recognition and Text-to-Speech engines, tools. We are developing new speech-enabled applications for Microsoft Windows using SAPI.

ABOUT VISUAL C++

The new Microsoft Visual C++ provides you with a comprehensive, up-to-the-minute production-level development

environment for developing all Windows 95/98 and windows NT Applications.

The Microsoft Visual C++ IDE is an Integrated Development Environment that allows you to easily create, open, view, edit, save, compile and debug all of your C and C++ application. As an integral part of the Microsoft Development Studio. The advantage of the language development suit is the ease of learning and use provided by such a specific languages syntax. We can easily develop and combine multi language source files in to one program.

The Visual C++ component of the Microsoft Developers Studio, just like any new state-of-the-art development environment, can on first encounter be a very intimidating product .While the initial window seems straight forward ,as soon as you begin peeking and poking around submenus and their related dialog windows you can easily become overwhelmed by the options and apparent complexity of this new world.

We developing multitasking ,object-oriented, GUI, Internet aware application really is not easy task.

MFC

MFC Library is a collection of C++ classes. It is provided as a dynamic link library, so our application has access to the classes in MFC. A DLL consists of executable functions that are loaded into the memory and independent from any application. Libraries such as MFC are called as Application Frameworks, because they give the user a framework for an application.

Application written in MFC is called API function, if needed. MFC as we know is an object-oriented interface to Windows.

MFC contains 200 classes, some of which you'll use directly and others of which will serve primarily as base classes for classes of your own. Some of the classes are exceedingly simple, such as the CPoint class that encapsulates a point object (a location characterized by x and y coordinates). Others are complex, such as the CWnd class that encapsulates the functionality of a window. In an MFC program, you rarely need to call the Windows API directly.

Instead, you create objects from MFC classes and call member functions belonging to those objects. Many of the hundreds of member functions defining the class library are thin wrappers around the Windows API and even have the same names as the corresponding API functions. MFC is not only a library of classes. MFC is also an application framework. More than merely

a Collection of classes, MFC helps define the structure of an application and handle many routine chores on the application's behalf. Starting with CWinApp, the class that represents application.Itself, MFC encapsulates virtually every aspect of program's operation. The framework supplies the WinMain() function and WinMain() in turn calls the application objects member functions to make the program go. One of the CwinApp member functions called By WinMain() - Run() - encapsulates the message loop that literally runs the program. The framework also provides abstractions that go above and beyond what the windows API has to offer. For example, MFC's Document /View architecture builds a powerful Infrastructure on top of the API that separates a program's data from Graphical representations ,or views, of that data. Such abstractions are totally foreign to the API and don't exist outside the framework of MFC.

The MFC development team designed a comprehensive implementation of the windows Application Program Interface (API). This C++ library encapsultaes the most important data structures and API function calls within a group of resuable classes.

Class libraries such as the MFC offer many advantages

- Elimination of functions and variable name collisions
- Encapsulation of code and data within the class

- Inheritance
- Often, reduced code size resulting from well-designed class libraries
- Resulting classes appearing to be the natural extensions of the language.
- An extensive exception-handling design that makes application code less Subject to failure.
- Complete support for all windows functions, controls, messages, GDI, Graphics primitives, menus and dialog boxes.
- Support for the Component Object Model(COM).
- Elimination of many switch/case statements that are a source of error.
- Small code with fast implementation.

SYSTEM DESIGN AND DEVELOPMENT

OVERVIEW OF THE SYSTEM

Speech Enabling System is an research oriented project . It is basically involves the concept of client-server technology wherein the client requests for some service through our voice, which is provided by the server. The main goal was to access the system and control the system through our voice without using keyboard and mouse.

To achieving this objective, the system contains information on careers both conventional and non-conventional such as Banking, Chartered Accountant, Chartered financial Analyst, Company Secretary, Fashion, Management, Merchandise, Foreign Languages etc.,

4.1 INPUT DESIGN

The application should provide the user given input through our voice. The style of input is established during software requirements analysis and Design input styles will vary the degree of human interaction. The inputs to the system are fully voice. Sequentially, we have to give the input. The first thing the application has to do is to choose the Input Media where the SR will occur. It does so by selecting a Microphone as an Input media.

The application then selects a voice for Speech Recognition Engine that suits its needs, with the language corresponding to Speech to text.

The SR engine is then connected to the desired Input Media. The application can begin to convert speak to text by sending voice to the SR engine. As you can see all this seems to be very simple but one of the main point is to connect the engine to audio hardware. At this time Microsoft provides audio destination objects for wave file input and for multimedia devices.

4.2 OUTPUT DESIGN

The output is the essential element of any system. As the output can be presented in many different ways, the appropriate method should be used while presenting the user with the output that is convenient for them. So the output is the prime element to be designed in the system. As the application itself is a report writing utility, reports have to be designed. Also dialogue which is the main form of interaction between the system and the user, should be friendly and must provide the after information if any in a convenient way.

4.3 PROCESS DESIGN

General SR Design Issues

There are a number of general issues to keep in mind when designing SR interfaces to your applications.

It is also a good idea to make speech services an optional feature whenever possible. Some installations may not have the hardware or RAM required to implement speech services. Even if the workstation has adequate resources, the user may experience performance degradation with the speech services active. It is a good idea to have a menu option or some other method that allows users to turn off speech services entirely.

When you add speech services to your programs, it is important to make sure you give users realistic expectations regarding the capabilities of the installation. This is best done through user documentation. You needn't go into great length, but you should give users general information about the state of SR technology.

Along with indications that speech services are active, it is a good idea to provide users with a single speech command that displays a list of recognized speech inputs, and some general online help regarding the use and capabilities of the SR services of your program. Since the total number of commands might be quite large,

you may want to provide a type of voice-activated help system that allows users to query the current command set and then ask additional questions to learn more about the various speech commands they can use.

It is also a good idea to add confirmations to especially dangerous or ambiguous speech commands. For example, if you have a voice command for "Delete," you should ask the user to confirm this option before continuing. This is especially important if you have other commands that may sound similar-if you have both "Delete" and "Repeat" in the command list you will want to make sure the system is quite sure which command was requested.

In general, it is a good idea to display the status of all speech processing. If the system does not understand a command, it is important to tell users rather than making them sit idle while your program waits for understandable input. If the system cannot identify a command, display a message telling the user to repeat the command, or bring up a dialog box that lists likely possibilities from which the user can select the requested command.

In some situations, background noise can hamper the performance of the SR engine. It is advisable to allow users to turn off speech services and only turn them back on when they are needed. This can be handled through a single button press or menu

selection. In this way, stray noise will not be misinterpreted as speech input.

There are a few things to avoid when adding voice commands to an application. SR systems are not very successful when processing long series of numbers or single letters. "M" and "N" sound quite alike, and long lists of digits can confuse most SR systems. Also, although SR systems are capable of handling requests such as "move mouse left," "move mouse right," and so on, this is not a good use of voice technology. Using voice commands to handle a pointer device is a bit like using the keyboard to play musical notes. It is possible, but not desirable.

Voice Command Menu Design

The key to designing good command menus is to make sure they are complete, consistent, and that they contain unique commands within the set. Good command menus also contain more than just the list of items displayed on the physical menu. It is a good idea to think of voice commands as you would keyboard shortcuts.

Useful voice command menus will provide access to all the common operations that might be performed by the user. For example, the standard menu might offer a top-level menu option of

Help. Under the Help menu might be an About item to display the basic information about the loaded application. It makes sense to add a voice command that provides direct access to the About box with a Help About command.

These shortcut commands may span several menu levels or even stand independent of any existing menu. For example, in an application that is used to monitor the status of manufacturing operations within a plant, you might add a command such as Display Statistics that would gather data from several locations and present a graph onscreen.

When designing menus, be sure to include commands for all dialog boxes. It is not a good idea to provide voice commands for only some dialog boxes and not for others.

SYSTEM IMPLEMENTATION & TESTING

5.1 SYSTEM IMPLEMENTATION

Implementing Speech Technology

Windows allows us to implement speech technology for your automotive application. Speech technology enables you to use voice command , voice dictation and voice response as a means of interacting with an application in the automotive computing environment. Speech technology provides an alternative means of interacting with an automotive computing device.

Speech recognition

Speech recognition is the ability of a computer to understand a spoken word for the purpose of receiving command and data input from the speaker.

Technology Issues

As advanced as SR/TTS technology is, it still has its limits. This section covers the general technology issues for SR and TTS engines along with a quick summary of some of the limits of the process and how this can affect perceived performance and system design.

SR Techniques

The following techniques are used for speech recognition

Speech recognition technology can be measured by three factors

1. Word selection
2. Speaker dependence
3. Word analysis

Word selection deals with the process of actually perceiving "word items" as input. Any speech engine must have some method for listening to the input stream and deciding when a word item has been uttered. There are three different methods for selecting words from the input stream. They are

- Discrete speech
- Word spotting
- Continuous speech

Discrete speech is the simplest form of word selection. Under discrete speech, the engine requires a slight pause between each word. This pause marks the beginning and end of each word item. Discrete speech requires the least amount of computational resources. However, discrete speech is not very natural for users. With a discrete speech system, users must speak in a halting voice. This may be adequate for short interactions with the speech system, but rather annoying for extended periods.

A much more preferred method of handling speech input is word spotting. Under word spotting, the speech engine listens for a list of key words along the input stream. This method allows users to use continuous speech. Since the system is "listening" for key words, users do not need to use unnatural pauses while they speak. The advantage of word spotting is that it gives users the perception that the system is actually listening to every word while limiting the amount of resources required by the engine itself. The disadvantage of word spotting is that the system can easily misinterpret input. For example, if the engine recognizes the word run, it will interpret the phrases "Run Excel" and "Run Access" as the same phrase. For this reason, it is important to design vocabularies for word-spotting systems that limit the possibility of confusion.

The most advanced form of word selection is the continuous speech method. Under continuous speech, the SR engine attempts to recognize each word that is uttered in real time. This is the most resource-intensive of the word selection methods. For this reason, continuous speech is best reserved for dictation systems that require complete and accurate perception of every word.

The process of word selection can be affected by the speaker. Speaker dependence refers to the engine's ability to deal with different speakers. Systems can be speaker dependent,

speaker independent, or speaker adaptive. The disadvantage of speaker-dependent systems is that they require extensive training by a single user before they become very accurate. This training can last as much as one hour before the system has an accuracy rate of over 90 percent. Another drawback to speaker-dependent systems is that each new user must re-train the system to reduce confusion and improve performance. However, speaker-dependent systems provide the greatest degree of accuracy while using the least amount of computing resources.

Speaker-adaptive systems are designed to perform adequately without training, but they improve with use. The advantage of speaker-adaptive systems is that users experience success without tedious training. Disadvantages include additional computing resource requirements and possible reduced performance on systems that must serve different people.

Speaker-independent systems provide the greatest degree of accuracy without performance. Speaker-independent systems are a must for installations where multiple speakers need to use the same station. The drawback of speaker-independent systems is that they require the greatest degree of computing resources.

Once a word item has been selected, it must be analyzed. Word analysis techniques involve matching the word item to a list

of known words in the engine's vocabulary. There are two methods for handling word analysis: whole-word matching or sub-word matching. Under whole-word matching, the SR engine matches the word item against a vocabulary of complete word templates. The advantage of this method is that the engine is able to make an accurate match very quickly, without the need for a great deal of computing power. The disadvantage of whole-word matching is that it requires extremely large vocabularies-into the tens of thousands of entries. Also, these words must be stored as spoken templates. Each word can require as much as 512 bytes of storage.

An alternate word-matching method involves the use of sub-words called phonemes. Each language has a fixed set of phonemes that are used to build all words. By informing the SR engine of the phonemes and their representations it is much easier to recognize a wider range of words. Under sub-word matching, the engine does not require an extensive vocabulary. An additional advantage of sub-word systems is that the pronunciation of a word can be determined from printed text. Phoneme storage requires only 5 to 20 bytes per phoneme. The disadvantage of sub-word matching is that it requires more processing resources to analyze input.

Windows for Automotive uses discrete speech recognition, which recognizes words delineated-or separated-by pauses.

Text-to-Speech (TTS) - TTS technologies convert textual information into synthetic speech output. TTS technologies are used in voice-processing applications requiring the creation of broad, unrelated and unpredictable vocabularies to formulate phrases that are “spoken” by the computing device. An automotive computing device can read text using the TTS algorithms to determine phrasing, or you can adjust the phrasing using control codes.

Implementing speech technology in an automotive application becomes a true advantage in the car environment. Speech technology allows the driver to interact with an application running on an automotive computing device without requiring the driver to shifting focus from the task of driving. Speech technology is a primary enabler of the digital car environment.

Microsoft Speech API

Speech recognition and TTS capabilities are integrated into an in-vehicle computing device with the Microsoft Speech API (SAPI), which provides a standardized method for applications to

integrate with speech technologies via communication with both speech recognition and TTS engines.

Windows implements SAPI by following API are used

- Voice-Text API
- Text-To-Speech API
- Voice Command API
- Speech Recognition API

Voice-Text API - This component provides high-level TTS capabilities-the audio equivalent of the printf function.

Text-To-Speech API - The Text-To-Speech API provides detailed control of a TTS engine. An automotive application can adjust the playback of spoken text by inserting text-to-speech control tags, which alter the talking speed, style, pitch and other qualities of the voice.

Audio Destination Object - The Audio Destination Object allows the TTS engine to send speech output to as multimedia wave device.

Voice Command API - The Voice Command API provides high-level speech recognition-often called command and control-for applications. Voice commands mimic Windows menus, giving the user a menu of commands to speak-such as “Tell me the time” or

“Open the file”. An application using voice commands can share a speech recognition engine and audio source with other applications on the user’s system.

Speech Recognition API

The Speech Recognition API provides detailed control of a speech recognition engine for application that requires more sophisticated command and control or applications that take dictation. Depending on the capabilities of the engine, an application may be able to get detailed information about the recognition result and use it to give feedback to the user or alter the result.

Audio Source Object

The Audio Source Object allows the speech recognition engine to obtain speech input from a multimedia wave device. The engine does not have to consider what kind of WAV input device is available or what kind of audio drivers or APIs are supported.

The Speech API is implemented as a series of Component Object Model (COM) interfaces. This chapter identifies the top-level objects, their child objects, and their methods.

The SAPI model is divided into two distinct levels

- (1) High-level SAPI
- (2) Low-level SAPI

High-level SAPI-This level provides basic speech services in the form of command-and-control speech recognition and simple text-to-speech output.

Low-level SAPI-This level provides detailed access to all speech services, including direct interfaces to control dialogs and manipulation of both speech recognition (SR) and text-to-speech (TTS) behavior attributes.

Each of the two levels of SAPI services has its own set of objects and methods.

High-Level SAPI

The high-level SAPI services provide access to basic forms of speech recognition and text-to-speech services. This is ideal for providing voice-activated menus, command buttons, and so on. It is also sufficient for basic rendering of text into speech.

The high-level SAPI interface has two top-level objects-one for voice command services (speech recognition), and one for voice text services (text-to-speech).

Voice Command

The Voice Command object is used to provide speech recognition services. It is useful for providing simple command-and-control speech services such as implementing menu options, activating command buttons, and issuing other simple operating system commands.

Voice Command Object

The Voice Command object supports three interfaces

The Voice Command interface

The Attributes interface

The Dialogs interface

The Voice Command interface is used to enumerate, create, and delete voice menu objects. This interface is also used to register an application to use the SR engine. An application must successfully complete the registration before the SR engine can be used. An additional method defined for the Voice Command interface is the Mimic method. This is used to play back a voice command to the engine; it can be used to "speak" voice commands directly to the SR engine. This is similar to playing keystroke or mouse-action macros back to the operating system.

The Attributes interface is used to set and retrieve a number of basic parameters that control the behavior of the voice command system. You can enable or disable voice commands, adjust input gain, establish the SR mode, and control the input device (microphone or telephone).

The Dialogs interface gives you access to a series of dialog boxes that can be used as a standard set of input screens for setting and displaying SR engine information. The SAPI model identifies five different dialog boxes that should be available through the Dialogs interface. The exact layout and content of these dialog boxes is not dictated by Microsoft, but is determined by the developer of the speech recognition engine. However, Microsoft has established general guidelines for the contents of the SR engine dialog boxes.

Dialog Box Name Description

About Box Used to display the dialog box that identifies the SR engine and show its copyright information.

Command Verification Can be used as a verification pop-up window during a speech recognition session. When the engine identifies a word or phrase, this box can appear requesting the user

to confirm that the engine has correctly understood the spoken command.

General Dialog Can be used to provide general access to the SR engine settings such as identifying the speaker, controlling recognition parameters, and the amount of disk space allotted to the SR engine.

Lexicon Dialog Can be used to offer the speaker the opportunity to alter the pronunciation lexicon, including altering the phonetic spelling of troublesome words, or adding or deleting personal vocabulary files.

The Voice Menu Object and the Menu Object Collection

The Voice Menu object is the only child object of the Voice Command object. It is used to allow applications to define, add, and delete voice commands in a menu. You can also use the Voice Menu object to activate and deactivate menus and, optionally, to provide a training dialog box for the menu.

The voice menu collection object contains a set of all menu objects defined in the voice command database. Microsoft SAPI defines functions to select and copy menu collections for use by the voice command speech engine.

The Voice Command Notification Callback

In the process of registering the application to use a voice command object, a notification callback (or sink) is established. This callback receives messages regarding the SR engine activity. Typical messages sent out by the SR engine can include notifications that the engine has detected commands being spoken, that some attribute of the engine has been changed, or that spoken commands have been heard but not recognized.

Voice Text

The SAPI model defines a basic text-to-speech service called voice text. This service has only one object—the Voice Text object. The Voice Text object supports three interfaces:

- The Voice Text interface
- The Attributes interface
- The Dialogs interface

The Voice Text interface is the primary interface of the TTS portion of the high-level SAPI model. The Voice Text interface provides a set method to start, pause, resume, fast forward, rewind, and stop the TTS engine while it is speaking text. This mirrors the VCR-type controls commonly employed for pc video and audio playback.

The Voice Text interface is also used to register the application that will request TTS services. An application must successfully complete the registration before the TTS engine can be used. This registration function can optionally pass a pointer to a callback function to be used to capture voice text messages. This establishes a notification callback with several methods, which are triggered by messages sent from the underlying TTS engine.

The attribute interface provides access to settings that control the basic behavior of the TTS engine. For example, you can use the Attributes interface to set the audio device to be used, set the playback speed (in words per minute), and turn the speech services on and off. If the TTS engine supports it, you can also use the Attributes interface to select the TTS speaking mode. The TTS speaking mode usually refers to a predefined set of voices, each having its own character or style (for example, male, female, child, adult, and so on).

The Dialogs interface can be used to allow users the ability to set and retrieve information regarding the TTS engine. The exact contents and layout of the dialog boxes are not determined by Microsoft but by the TTS engine developer. Microsoft does, however, suggest the possible contents of each dialog box.

When an SR engine is created, a link to a valid audio input device is also created. While it is possible to create a custom audio input device, it is not required. The default audio input device is an attached microphone, but can also be set to point to a telephone device.

The rest of this section details the low-level SAPI SR objects and their interfaces.

The SR Enumerator and Engine Enumerator Objects

The role of the SR Enumerator and Engine Enumerator objects is to locate and select an appropriate SR engine for the requesting application. The Enumerator object lists all available speech recognition modes and their associated installed engines. This information is supplied by the child object of the Enumerator object: the Engine Enumerator object. The result of this search is a pointer to the SR engine interface that best meets the service request.

The Enumerator and Engine Enumerator objects support only two interfaces :

- The ISREnum interface is used to get a list of all available engines.
- The ISRFind interface is used to select the desired engine.

Low-Level SAPI

The low-level SAPI services provide access to a much greater level of control of Windows speech recognition and text-to-speech services. This level is best for implementing advanced SR and TTS services, including the creation of dictation systems.

SPEECH RECOGNITION

The Speech Recognition object has several child objects and collections. There are two top-level objects in the SR system: the SR Engine Enumerator object and the SR Sharing object. These two objects are created using their unique CLSID (class ID) values. The purpose of both objects is to give an application information about the available speech recognition engines and allow the application to register with the appropriate engine. Once the engine is selected, one or more grammar objects can be created, and as each phrase is heard, an SR Results object is created for each phrase. This object is a temporary object that contains details about the phrase that was captured by the speech recognition engine. Figure 15.2 shows how the different objects relate to each other, and how they are created.

The SR Sharing Object

The SR Sharing object is a possible replacement for the SR Enumerator and Engine Enumerator objects. The SR Sharing object uses only one interface, the ISRSharing interface, to locate and select an engine object that will be shared with other applications on the pc. In essence, this allows for the registration of a requesting application with an out-of-process memory SR server object. While often slower than creating an instance of a private SR object, using the Sharing object can reduce strain on memory resources.

The SR Sharing interface is an optional feature of speech engines and may not be available depending on the design of the engine itself.

The SR Engine Object

The SR Engine Object is the heart of the speech recognition system. This object represents the actual speech engine and it supports several interfaces for the monitoring of speech activity. The SR Engine is created using the Select method of the ISREnum interface of the SR Enumerator object described earlier. Table 15.3 lists the interfaces supported by the SR Engine object along with a short description of their uses.

The Grammar Object

The Grammar object is a child object of the SR Engine object. It is used to load parsing grammars for use by the speech engine in analyzing audio input. The Grammar object contains all the rules, words, lists, and other parameters that control how the SR engine interprets human speech. Each phrase detected by the SR engine is processed using the loaded grammars.

The Grammar object supports three interfaces :

- **ISRGramCFG**-This interface is used to handle grammar functions specific to context-free grammars, including the management of lists and rules.
- **ISRGramDictation**-This interface is used to handle grammar functions specific to dictation grammars, including words, word groups, and sample text.
- **ISRGramCommon**-This interface is used to handle tasks common to both dictation and context-free grammars. This includes loading and unloading grammars, activating or deactivating a loaded grammar, training the engine, and possibly storing SR results objects.

The Grammar object also supports a notification callback to handle messages regarding grammar events. Optionally, the grammar object can create an SR Results object. This object is discussed fully in the next section.

The SR Results Object

The SR Results object contains detailed information about the most recent speech recognition event. This could include a recorded representation of the speech, the interpreted phrase constructed by the engine, the name of the speaker, performance statistics, and so on.

- **ISRResBasic** Used to provide basic information about the results object, including an audio representation of the phrase, the selected interpretation of the audio, the grammar used to analyze the input, and the start and stop time of the recognition event.
- **ISRResAudio** Used to retrieve an audio representation of the recognized phrase. This audio file can be played back to the speaker or saved as a WAV format file for later review.
- **ISRResGraph** Used to produce a graphic representation of the recognition event. This graph could show the phonemes used to construct the phrase, show the engine's "score" for accurately detecting the phrase, and so on.

- `ISRResCorrection` Used to provide an opportunity to confirm that the interpretation was accurate, possibly allowing for a correction in the analysis.
- `ISRResEval` Used to re-evaluate the results of the previous recognition. This could be used by the engine to request the speaker to repeat training phrases and use the new information to re-evaluate previous interpretations.
- `ISRResSpeaker` Used to identify the speaker performing the dictation. Could be used to improve engine performance by comparing stored information from previous sessions with the same speaker.
- `ISRResModifyGUI` Used to provide a pop-up window asking the user to confirm the engine's interpretation. Could also provide a list of alternate results to choose from.
- `ISRResMerge` Used to merge data from two different recognition events into a single unit for evaluation purposes. This can be done to improve the system's knowledge about a speaker or phrase.
- `ISRResMemory` Used to allocate and release memory used by results objects. This is strictly a housekeeping function.

GRAMMER

A grammar defines the words or phrases that an application can recognize. Speech recognition is based on grammars. An application can perform speech recognition by using three different types of grammars: context-free, dictation, and limited-domain. Each type of grammar uses a different strategy for narrowing the set of sentences it will recognize. Context-free grammar uses rules that predict the next words that might possibly follow the word just spoken, reducing the number of candidates to evaluate in order to make recognition easier. Dictation grammar defines a context for the speaker by identifying the subject of the dictation, the expected language style, and the dictation that's already been performed. Limited-domain grammar does not provide strict syntax structures, but does provide a set of words to recognize. Limited-domain grammar is a hybrid between a context-free grammar and a full dictation grammar.

Each grammar has its advantages and disadvantages. Context-free grammars offer a high degree of accuracy with little or no training required and mainstream PC requirements. Their drawback is that they cannot be used for data entry, except from a list of predefined phrases. They do offer a way to begin offering speech capabilities in products without making large demands on

users before they understand the benefits of speech recognition. They represent an ideal entry point to begin rolling this technology out to a general audience. You can achieve up to 97% recognition accuracy by implementing commands and very small grammars

Dictation grammars require a much larger investment in time and money for most people to be able to use in any practical way. Speech recognition is an excellent fit for clinicians, who are accustomed to dictating patient information and then having transcriptionists type that data into a computer. The feedback from early adopter audiences is helping to accelerate the development of usable speech recognition interfaces.

None of the current speech recognition vendors are achieving greater than 95% accuracy with general English dictation grammars. That translates to one mistake for every 20 words, which is probably not acceptable to most people. The problem is further magnified when a user verbally corrects something and their correction is not recognized. Most users will not tolerate this and will give up on the technology. If a more limited dictation grammar is used, levels of accuracy over 95% can be achieved with a motivated user willing to put in months of effort.

Limited-domain grammars represent a way to increase speech recognition accuracy and flexibility in certain situations

without placing large demands on users. An application might use a limited-domain grammar for the following purposes

Command and control that uses natural language processing to interpret the meaning of the commands

Forms data entry in which the scope of the vocabulary is known ahead of time
Text entry in which the scope of the vocabulary is known ahead of time
This type of grammar could be an interim step between context-free and dictation grammar-based applications.

VOICE DICTATION

IMPLEMENTATION OF VOICE DICTATION IN WINDOWS APPLICATION

The windows applications the input given to the system through via following

This application can be activated in 3 modes.

- Using Keyboard
- Using mouse - Clicking on the command
- Using speech enabling system.

In the voice activation mode the command input from the user is recorded, converted into digital values. It is compared with

pronunciation dictionary, then it is converted into text in the application.

The "Voice Dictation" module allows users to dictate into applications. It uses the metaphor of an invisible edit box to describe how the application receives and displays text from a user's speech.

The "Voice Dictation" object employs the invisible edit box to receive the text whenever the user speaks a word. The application employing the Voice Dictation object is then notified that the text in the edit box has changed, and how it has changed, so that it can display any changes to the user.

Voice Dictation provides considerably more functionality than mere transcription. It provides automatic text formatting (capitalization, spacing), translation of punctuation words into symbols, built-in glossary entries, limited commands, and a GUI so users can correct the dictation engine. It has the potential to support more functionality in the future, especially when continuous dictation becomes widespread. Thus, an application will most likely maintain a GUI of its own with a rich-text control that can be kept synchronized with the invisible edit box, displaying formatted and punctuated text from dictation.

About Voice Dictation

Some elements of Voice Dictation that you should know of are .

Topics

Speech lends itself to classification by topic. An application should have a unique "topic" for each classification of document that might be dictated. Example topics might be "E-mail", "Formal Writing", or "C-code" (these classifications use drastically different styles of language). Voice Dictation will store information on topics away so that any information the recognizer learns will be stored, along with any glossary entries that the user generates specifically for the topic.

Adding a new topic might take a lot of hard disk or memory, so unless an application has a very good reason for creating a new topic, it should try to use an existing topic. Topics cannot be merged.

Much of the information stored within a topic data structures is engine dependent so switching dictation engine vendors may cause a loss of some data.

Dictation Session

Whenever an application wishes to use dictation, it must instantiate a "Dictation Session." Each session must be based upon

a "Topic" and have a window handle associated with it. The dictation session will only be actively listening when that window has the keyboard focus. It is possible for an application to have multiple sessions, or multiple applications each to have their own sessions. As a user switches between the applications he/she will be dictating into a different session.

Correction Window

The application can also have the dictation object provide some GUI, although the application doesn't necessarily need the GUI.

The "Correction Window" displays some user-interface that allows the user to correct the last word spoken and/or the currently selected word. It will look something like this: (The Correction Window GUI may look different for continuous dictation systems.)

Glossary Entries

A glossary entry is one or more words that, if dictated, will be replaced by a text string. It is a textual translation of a symbol. Thus, a user could have a glossary entry for "copyright" that converts it to "©", or "a t and t" to "AT&T". Since users can add

glossary entries, they are designed so they're easy for users to understand and modify.

Applications can provide their own glossary entries for a specific site, and even more complex interactions by providing their own commands.

Voice Dictation provides the following built-in glossary entries. Different glossary entries might be built-in, depending upon the language.

Glossary	Content
"Indent-Paragraph" Miscellaneous	Inserts a tab character Miscellaneous conversions like "et-cetera" to "etc."
"New-Line"	Inserts a new line.
"New-Paragraph"	Inserts two lines.
"One-Space" to "Ten-Spaces"	Inserts several spaces.
"Paragraph"	Inserts two lines
Punctuation	Inserts various punctuation marks.

Commands

Applications should use commands for non-textual actions or textual-modifications that are too complex for glossary entries. Voice Dictation supports some command & control capability while the user dictates-it listen for commands specified by the application. Users can modify these commands.

Voice Dictation also provides built-in commands. We have a limited number in the first version but will expand the built-in commands as resources and technology (continuous dictation) allow. Applications cannot add built-in commands, although they can rename or disable them if they don't use them. Voice Dictation provides the following built-in commands. More commands may be available, depending upon the language.

Inverse Text Normalization

"Inverse text normalization" is a feature in the Voice Dictation module that automatically formats text spacing, capitalization, and converts some phrases to a more symbolic form. For example, inverse text normalization is what converts the spoken phrase, "the time is eight oh five p m period" to "The time is 8:05 PM." Applications can control the inverse text normalization

by passing a Voice Dictation context free grammar into the Voice Dictation module. A description of the Voice Dictation Context Free Grammar is given later.

Voice Dictation provides the following "inverse text normalization" features to modify the text. Some of them can be turned on/off.

Feature	Effect
Automatic Spacing	Places spaces between words and sentences.
Automatic Capitalization of Words.	Learns the capitalization corrections from users and use that capitalization until the user changes it again.
Automatic Capitalization of Sentences.	Automatically capitalizes words at the beginning of a sentence.
Number to Digits	Converts "six hundred forty two" to "642". It also handles decimals.
Years	Converts "nineteen ninety four" to "1994".

Interaction with Voice Commands

Voice Dictation will use the speech recognition sharing object. It will share a speech recognition engine with voice-commands if the engine can support both dictation and continuous CFGs. If the engine supports this, users will be able to have dictation and command and control listening at the same time. If two separate engines are used then Voice Dictation and voice commands will be mutually exclusive, so turning on one will turn off the other.

Reducing Memory Footprint

The Voice Dictation object maintains a mirror image of the text that is in the document. If a user loads in a large document, this might get unwieldy, especially if audio recordings are maintained for each of the words. Audio data is inherently large. Here are some ways to alleviate some of the memory problems: 1? The application can call `IVDctAttributes` . Alternatively, an application might only keep the last page or paragraph that the user has dictated in the Voice Dictation buffer. This places a cap on the size of the data and may work well for the user. However, if the user scrolls back a

few pages and wishes to hear what he/she said then they might be disappointed because the information won't be available.

Localization Issues

For the Voice Dictation object to correctly recognize a specific language it must be supplied with an engine supporting the specific language. Most features of the Voice Dictation object will work without any localization of the object, but some features require localization.

Features that need localization

Default Global Glossary - If no default table for the global glossary exist for the specific language then nothing will be loaded in. An application or user could add their own in. Punctuation is handled by the glossary.

Built-In Commands : If a translation table for the built-in commands does not exist then the English-named commands will be loaded in. An application or user could rename them into the vernacular.

Inverse Text Normalization : If the localized default CFG is not available then the application will need to set one.

5.2 SYSTEM TESTING

Testing is an activity that ensures that a correct system is built. It is restricted to being performed after the development phase is completed. But it is to be carried out in parallel with all the stages of system development. Software testing has been carried out by using the existing data and found to be matching well.

The implementation of the system in the organization comes as the last Step of the system development. This phase of system consist of

1. Testing the document programs with the sample input
2. Correction of any errors identified.
3. Making necessary changes to the system with the actual data.
4. Performing a parallel run of the system and find out any errors of calculation present and correct to them
5. Training the user personnel.

Testing Objectives :

There are several rules that can serve as testing objectives. They are,

- Testing is a process of executing a program with the intension of finding an error.

- A good test case is one that has a high probability of finding an undiscovered error.

- A successful test is one that uncovers an undiscovered error

If testing is conducted successfully according to the objection stated above, it could uncover errors in the software.

Testing has become an integral part of any system or project. When the software is developed, before it is given to the user it must be tested whether it is solving the purpose for which it is developed.

Testing is the process where the test data is prepared and is used for testing the modules individually and later validated.

The following is the description of testing strategies, which are carried out during the testing period. Code testing examines the logic of the program. The module testing, to locate error in each module is tested individually. This enables us to detect errors and correct it without affecting other modules. When the user finds no major problems with its accuracy the system passes to a final acceptance test. This test confirms that the system meets the original objectives, goals and requirements.

5.3 REFINEMENTS BASED ON FEEDBACK

Now you might ask whether the ability to talk to your computer is actually a good thing. For a start, it's not always the most efficient means of communication. It's also true that the interfaces we use to address the PC aren't geared up to speech--they're primarily visual rather than aural. And then there are the business and security issues: would you want to work in an office full of people talking to their PCs? It's also true that to get the best out of most packages you'll need to spend time with them before doing anything useful, teaching them how you speak.

However, for a large number of users who don't want to get to grips with an interface that's a long way from perfect, speech recognition could be a godsend. In particular, the omnipresence of PCs has caused some users to suffer repetitive strain injuries (RSI), which speech recognition is well placed to alleviate.

But speech recognition isn't confined to PCs. You're starting to see it all over the place, from automated phone systems to railway announcements. Maybe the next car you buy will include a form of speech recognition, allowing you to turn up the volume on your favourite CD track and ask for directions to your destination, all without having to take your hands off the wheel. Known as

embedded applications, you can expect this area to really take off over the next year or two.

What we're providing here is a start-to-finish guide to speech recognition, from providing you with the reasons why you should be thinking about how the technology can help you, to an exposition of how it actually works. We've even looked at the history of speech recognition, as well as using PC Magazine's expertise to look into the future, and see where this exciting technology is heading.

We're on the verge of a big breakthrough in terms of speech recognition technology and, more importantly, its application in many areas of our lives. This will give greater numbers of people access to the benefits of computers, without their even knowing that they're interacting with them. If that means people's lives are enriched, it can't be a bad thing.

It is important to understand the limits of current SR technology and how these limits affect system performance. Three of the most vital limitations of current SR technology are :

- Speaker identification
- Input recognition
- Recognition accuracy

The first hurdle for SR engines is determining when the speaker is addressing the engine and when the words are directed to someone else in the room. This skill is beyond the SR systems currently on the market. Your program must allow users to inform the computer that you are addressing the engine. Also, SR engines cannot distinguish between multiple speakers. With speaker-independent systems, this is not a big problem. However, speaker-dependent systems cannot deal well in situations where multiple users may be addressing the same system.

Even speaker-independent systems can have a hard time when multiple speakers are involved. For example, a dictation system designed to transcribe a meeting will not be able to differentiate between speakers. Also, SR systems fail when two people are speaking at the same time.

SR engines also have limits regarding the processing of identified words. First, SR engines have no ability to process natural language. They can only recognize words in the existing vocabulary and process them based on known grammar rules. Thus, despite any perceived "friendliness" of speech-enabled systems, they do not really understand the speaker at all.

SR engines also are unable to hear a new word and derive its meaning from previously spoken words. The system is incapable of spelling or rendering words that are not already in its vocabulary.

Finally, SR engines are not able to deal with wide variations in pronunciation of the same word. For example, words such as either (ee-ther or I-ther) and potato (po-tay-toe or po-tah-toe) can easily confuse the system. Wide variations in pronunciation can greatly reduce the accuracy of SR systems.

Recognition accuracy can be affected by regional dialects, quality of the microphone, and the ambient noise level during a speech session. Much like the problem with pronunciation, dialect variations can hamper SR engine performance. If your software is implemented in a location where the common speech contains local slang or other region-specific words, these words may be misinterpreted or not recognized at all.

Poor microphones or noisy office spaces also affect accuracy. A system that works fine in a quiet, well-equipped office may be unusable in a noisy facility. In a noisy environment, the SR engine is more likely to confuse similar-sounding words such as out and pout, or in and when. For this reason it is important to emphasize the value of a good microphone and a quiet environment when performing SR activities.

CONCLUSION

The Speech Enabling System is used to activate and access the system through our voice. Speech recognition represents a powerful way for a user to access information in the system. For users who work in an environment where speed and ease of access are critical, it holds enormous promise for future applications. As hardware continues to become more powerful and cheaper, speech recognition should continue to become more accurate and useful to increasingly wider.

SCOPE FOR FUTURE DEVELOPMENT

There are enormous practical utilities and advantages with voice user interface between user and machine. These systems with speech recognition could be used to automate the industries, thereby reducing the man power to minimum. A speech is the most natural and spontaneous way of communication with humans, it would be extremely advantageous to have speech communication with the machine. The data could be entered in to the computer through the speech communication between man and computer. Such communications increase the convenience of information transfer from man to machine. Isolated word recognizer could be used at 20 to 100 word per minute.(compared to 15 words per minute for a slow typist and about 45 words for a fast one), While continuous speech recognizers could operate at 150 to 200 words per minute handicapped person could operate various devices by spoken command, making him less dependent on others. Speech recognition systems could be used for security purpose and it is also used for language translators.

It's going to be a conversation-fuelled future. Very soon we'll not only be talking to our PCs, but also chatting to our TV remote controls, central heating boilers and lighting systems.

Speech recognition packages for PCs have already come a long way, but there are advances still to be made. Some are already planned, but need the power of PC hardware to catch up.

Many advances come in the areas of control and commands, rather than dictation. Intelligent phrase recognition takes speech input and turns it into commands for your PC and the applications it runs. We can already see this in a limited way on some speech recognition packages, but the power simply isn't there to provide support for more than a limited subset of an application's features. Fortunately, the technology involved in phrase recognition is largely input independent. This means that a system designed for use with text input, say from a Web page, can become speech enabled with the addition of memory and processing power. Large-vocabulary, application-independent phrase recognition is still some way off, because of this limitation. But for specific applications, the vocabulary involved is smaller, and less processing power is needed. We're likely to see this technology in use in embedded speech recognition as well as for controlling PC applications.

Speech feedback from your PC has already started to appear in limited forms. Some recognition packages include a text-to-speech engine that can read your dictation back to you. The

b same engine can be applied to the messages within dialog boxes, so it shouldn't be long before we start to see the first generation of speech user interfaces. Speech synthesis technology lags behind recognition in terms of sophistication, and the voices it produces still don't sound natural. In interactive voice response systems, the solution is to use recorded speech and play this back. While this makes the speech natural, it limits the system's vocabulary. A recorded speech system would never be practical for feedback from a PC, since the many applications available all have different functions and errors. With a high-quality speech synthesiser, it should be possible to interact with a computer on the same terms as you would another person.

BIBLIOGRAPHY

The following books and websites were referred for developing the Project.

Books for reference

- David J. Kruglinski, George Shepherd, and Scot Wingo, "**Programming Microsoft Visual C++**", 1998, Fifth edition.
- Peter Norton, Rob McGregor, "**Programming with MFC**", 1999, Fourth edition.
- John Paul Mueller, "**Visual C++ 5**", 1997, Fourth edition.
- Yashavant Kanetkar, "**Visual C++ Programming**", 1998, Third edition.
- Ian Sommerville, "**Software engineering**", 1999, Fifth edition.
- Roger S. Pressman, "**Software engineering**", 1999, Fourth edition.
- Richard Flair, "**Software Project management**", 1998, second edition.

Websites for references

<http://www.sunny-beach.net/vbvoicetelephony.htm>

<http://www.speech.cs.cmu.edu/comp.speech/Section1/Interfaces/ms.speech.api.htm>

<http://www.research.microsoft.com/srg/docs/>

<http://www.speech.cs.cmu.edu/comp.speech/Section1/Interfaces/asapi.html>

<http://www.apple.com/macos/speech/>

<http://www.speech.cs.cmu.edu/comp.speech/>

<http://www.speech.cs.cmu.edu/comp.speech/FAQ.Contents>

<http://svr-www.eng.cam.ac.uk/~ajr/SA95/node1.html>

<ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/recognition/>

<http://research.microsoft.com/research/srg/>>

<http://research.microsoft.com/research/srg/>SAPI Basics>

<http://www.voicexml.org/>

<http://www.research.microsoft.com/research/srg/>

<http://www.synapseadaptive.com/>

<http://www.synapseadaptive.com/recorders/recorders.htm>

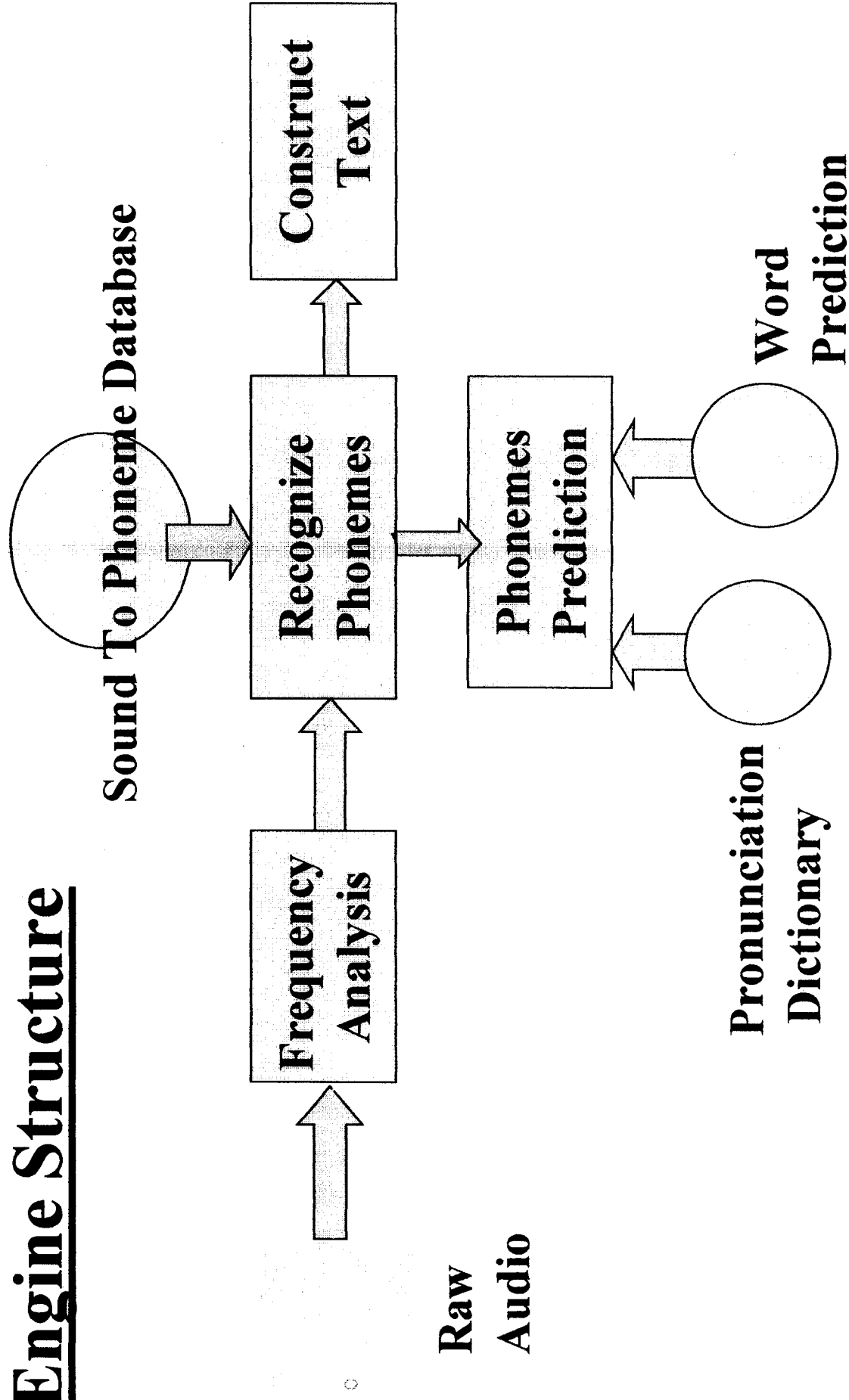
<http://www.isip.msstate.edu/projects/speech/software/index>

www.isip.msstate.edu/projects/speech/education/tutorials/

[isip_env/index.htm](http://www.isip.msstate.edu/projects/speech/education/tutorials/isip_env/index.htm)

<http://msdn.microsoft.com/library/periodic/period99/ppt2000.htm>

Engine Structure



Voice Training: Default Speech Profile



Get ready to start training the computer to your voice.



Make sure your room is quiet, and that you won't be disturbed for the next ten minutes.



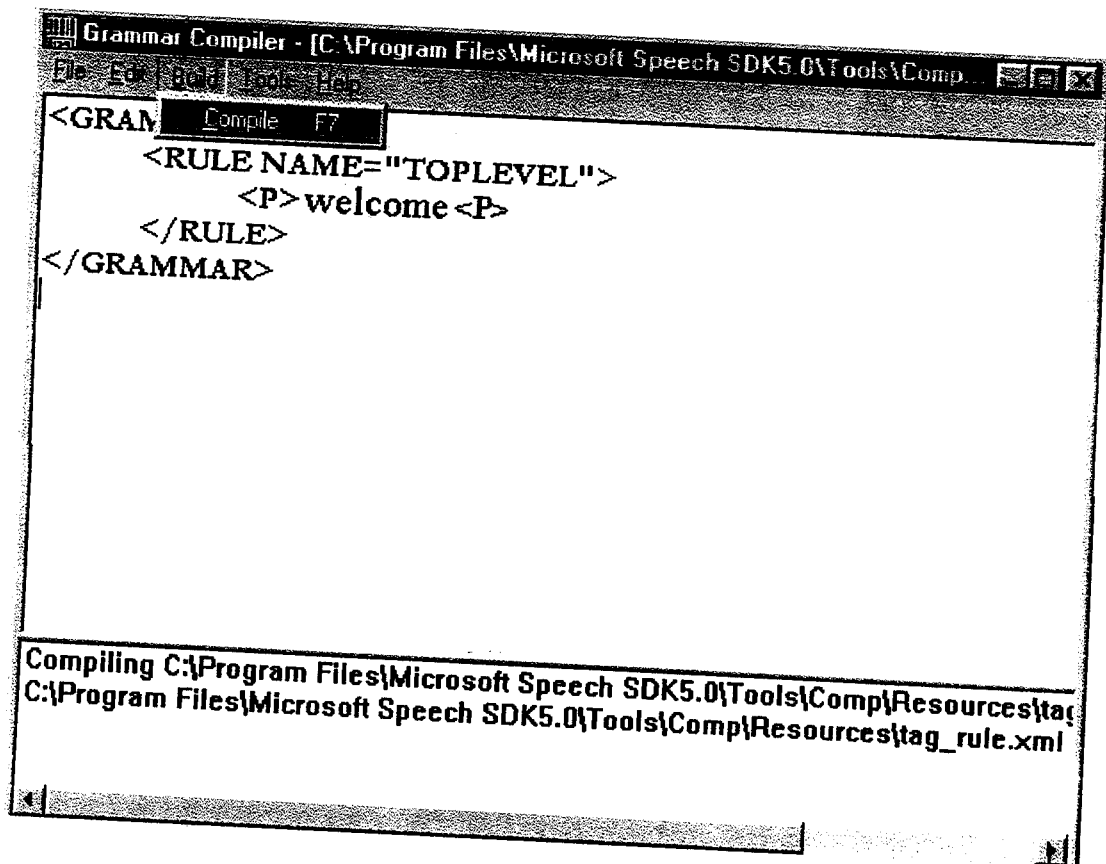
Make sure your microphone is positioned properly.

When you press "Next" training will begin.

< Back

Next >

Cancel



The image shows a window titled "Grammar Compiler - [C:\Program Files\Microsoft Speech SDK5.0\Tools\Comp...". The window contains a text editor with the following XML code:

```
<GRAMMAR>  
  <RULE NAME="TOPLEVEL">  
    <P>welcome</P>  
  </RULE>  
</GRAMMAR>
```

Below the code, a status bar or message box displays the text: "Compiling C:\Program Files\Microsoft Speech SDK5.0\Tools\Comp\Resources\tag_rule.xml".

