UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ELETRÔNICA E DE COMPUTAÇÃO

Sistema de reconhecimento automático de voz por telefone

Autor:
Guilherme Pastor Garnier

Orientador:
Fernando Gil Vianna Resende Junior, Ph.D.

Co-orientador:
Sérgio Barbosa Villas-Boas, Ph.D.

Examinador:
Márcio Nogueira de Souza, D.Sc.

DEL
Dezembro de 2004

Agradecimentos

Agradeço primeiramente aos meus pais, Gustavo e Elizabeth, por tudo que fizeram e fazem por mim, e por todo o apoio durante a minha vida.

À minha namorada, Juliana, pelo incentivo diante de cada obstáculo encontrado no caminho, principalmente nos momentos mais difíceis.

Aos meus orientadores, Fernando Gil e Sérgio Villas-Boas, por todo o apoio e dedicação ao longo deste projeto.

Aos colegas Pablo Salino Cunha e Rafael Teruszkin, que prestaram grande ajuda no início deste projeto, e a todos que me ajudaram, direta ou indiretamente.

Resumo

Neste trabalho foi desenvolvido um sistema de reconhecimento de voz através da linha telefônica. O sistema foi desenvolvido na linguagem C++, e realiza o acesso à linha telefônica através de qualquer placa de voice modem. Para a interface entre o sistema e o voice modem, foram utilizadas a interface TAPI e a biblioteca TFX, que encapsula a interface TAPI em classes, acrescentando a ela novas funcionalidades. O reconhecimento de voz utiliza a biblioteca Voice and Sound. A estrutura de menus utilizada no sistema é baseada em XML, e utiliza a biblioteca Xerces. O sistema detecta automaticamente o voice modem instalado no computador e monitora a linha telefônica, aguardando o recebimento de chamadas. Quando isto ocorre, o sistema atende a chamada e oferece menus de opções ao usuário, através de mensagens de voz. A seleção da opção desejada nos menus é realizada através da voz. Nos testes realizados, foi obtida uma taxa de 95,18 % de acerto no reconhecimento de voz.

Palavras-Chave

Integração Computador-Telefonia

TAPI

Voice Modem

Reconhecimento de Voz

XML

Waveform Audio

Lista de Acrônimos

ADSI: *Analog Display Services Interface*;

API: *Application Programming Interface*;

CTI: *Computer-Telephony Integration*;

DOM: *Document Object Model*;

DTMF: *Dual Tone Multi-Frequency*;

HMM: *Hidden Markov Models*;

HTML: *HyperText Markup Language*;

ISDN: *Integrated Services Digital Network*;

LAN: *Local Area Network*;

MFC: *Microsoft Foundation Classes*;

PABX: *Private Automatic Branch Exchange*;

PBX: *Private Branch Exchange*;

PCM: *Pulse Code Modulation*;

POTS: *Plain Old Telephone Service*;

PSTN: *Public Switched Telephony Network*;

SAX: *Simple API for XML*;

TAPI: *Telephony Application Programming Interface*;

TDD: *Telephony Devices for the Deaf*;

TFX: *Telephony Framework*;

TSP: *TAPI Service Provider*;

TSPI: *TAPI Service Provider Interface*;

WOSA: *Windows Open Services Architecture*;

XML: *Extensible Markup Language*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nas últimas décadas, o computador vem sendo cada vez mais utilizado em diversas atividades. O principal motivo da disseminação do uso dos computadores é a facilidade para a realização de diversas tarefas do dia-a-dia automaticamente, muitas delas sem a necessidade de intervenções humanas.

Entre as muitas aplicações possíveis dos computadores, há os sistemas de atendimento por telefone. Nestes sistemas, tradicionalmente, há uma ou mais linhas telefônicas, e, sem o uso de computadores, há funcionários responsáveis pelo atendimento das chamadas. Estes sistemas são bastante utilizados para atendimento de clientes em empresas. Assim que uma chamada é recebida, um funcionário deve atendê-la e fornecer informações ou receber reclamações de clientes.

Os computadores simplificam bastante o uso deste tipo de sistema, pois eliminam a necessidade de funcionários preparados para atender as chamadas. Em vez disto, mantém-se o computador ligado a uma linha telefônica, aguardando o recebimento de chamadas. Quando isto acontece, o computador deve atender a chamada e enviar mensagens de voz, previamente gravadas, através da linha.

Para receber informações do usuário, a maioria destes sistemas monitora os tons DTMF (Dual Tone Multi Frequency) do teclado telefônico da pessoa que fez a chamada. Desta maneira, o computador é capaz de descobrir quando e quais teclas são pressionadas. Para o uso deste tipo de sistema, as mensagens de voz enviadas pelo computador através da linha telefônica devem informar ao usuário que fez a chamada quais são as opções disponíveis no sistema, como um menu, associando cada opção a uma tecla do telefone. Assim, o usuário deve pressionar a tecla que foi associada à opção desejada, informando ao sistema a sua

escolha. Este tipo de sistema é bastante comum, sendo utilizado em diversas situações, como, por exemplo, em sistemas de informações de cinemas, que fornecem informações sobre os filmes em cartaz, horários de sessões, preços de ingressos e promoções, ou em sistemas de atendimentos a clientes de operadoras de telefonia, que permitem consultar saldos, acessar a caixa postal ou inserir créditos no telefone celular, entre outras funções.

O objetivo deste trabalho é o desenvolvimento de um sistema baseado em menus por telefone, como os exemplos citados anteriormente, porém controlado por voz. Neste sistema, elimina-se a necessidade do uso das teclas do telefone, pois seleciona-se a opção desejada através da voz, falando a palavra correspondente. Assim que uma mensagem de voz com as opções disponíveis é enviada pela linha telefônica, a voz do usuário é gravada, e o sistema de reconhecimento de voz detecta a opção escolhida. Isto permite o acesso às opções do sistema de forma mais simples e direta. Uma das vantagens deste sistema é permitir que deficientes visuais possam utilizá-lo facilmente, pois, após a discagem, não é necessário utilizar o teclado telefônico.

Na pesquisa por uma solução que permitisse o acesso à linha telefônica através do computador, uma das principais dificuldades deste trabalho, levou-se em conta alguns fatores, como a facilidade de uso e de configuração do voice modem. Após algum tempo de pesquisa na Internet, foi encontrada a interface TAPI, que permite o acesso direto ao modem, sem a necessidade de configurá-lo manualmente. Para o uso desta interface, foi utilizada a biblioteca TFX, desenvolvida em C++, que encapsula todas as funcionalidades da interface TAPI em classes, o que facilita ainda mais o seu uso. Além disso, a bilioteca TFX ainda adiciona novas funcionalidades, como uma classe que permite a gravação e reprodução da voz pela linha telefônica.

No reconhecimento de voz, foi utilizada a biblioteca Voice and Sound, também desenvolvida em C++, o que facilitou bastante a integração com a biblioteca TFX. O reconhecimento é baseado em coeficientes cepstrais e HMM's. A voz gravada é segmentada em trechos, para a detecção de início e fim da palavra. Em seguida, são extraídos parâmetros da voz, para comparação com os modelos previamente gravados em um treinamento, o que permite o reconhecimento da palavra que foi dita pelo usuário do sistema. Neste treinamento, cada palavra é gravada várias vezes por pessoas diferentes, permitindo altas taxas de acerto no reconhecimento de voz.

Este trabalho está organizado da seguinte forma: o Capítulo 2 traz uma visão geral

do sistema, apresentando sua organização e estrutura; no Capítulo 3, é descrita a integração entre o computador e a linha telefônica; o Capítulo 4 descreve como foram realizadas a gravação, detecção e reconhecimento de voz no sistema; no Capítulo 5, é apresentada a estrutura de menus do sistema, e como estes são organizados e modificados; o Capítulo 6 mostra a interface desenvolvida para o programa; o Capítulo 7 apresenta os resultados obtidos em testes com o sistema; e o Capítulo 8 mostra as conclusões do projeto e propostas para sua continuidade.

Capítulo 2

Estrutura e organização do sistema

O sistema desenvolvido [1] pode ser esquematizado como mostra a Figura 2.1. O programa que foi desenvolvido neste trabalho deve rodar num computador que esteja ligado a uma linha telefônica, através de uma placa de voice modem, e rodando sistema operacional Windows, versão 95 ou superior. O usuário do programa, tendo acesso ao computador onde o programa está instalado, pode realizar a configuração do sistema, que, ao iniciar, aguarda por chamadas através da linha telefônica.

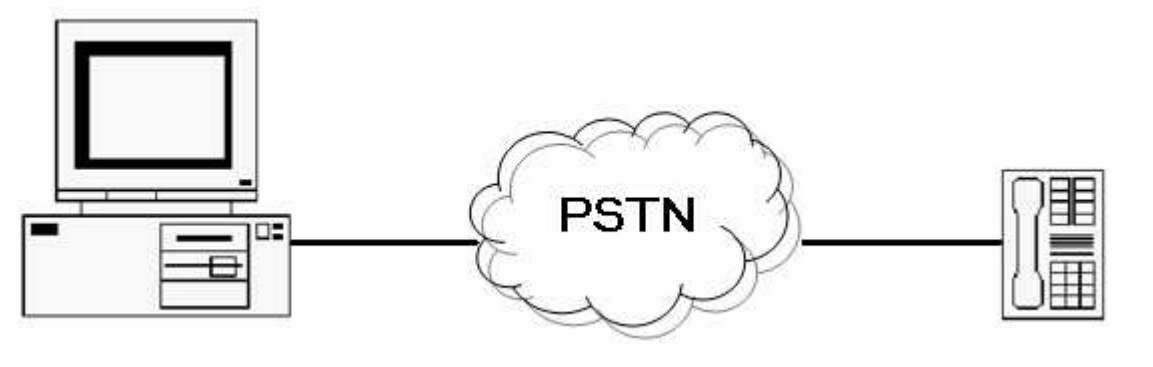

Figura 2.1: Esquematização do sistema

A linha telefônica ligada ao computador através do voice modem está ligada à PSTN (Public Switched Telephony Network), a rede pública de telefonia [2]. Em outra linha telefônica, há o usuário do sistema, que precisa apenas de um telefone para utilizá-lo. Ao fazer uma chamada para o número de telefone correspondente à linha ligada ao computador que tem o programa rodando, este irá atender a chamada, dando início à comunicação entre o sistema e o usuário, como ilustra a Figura 2.2.

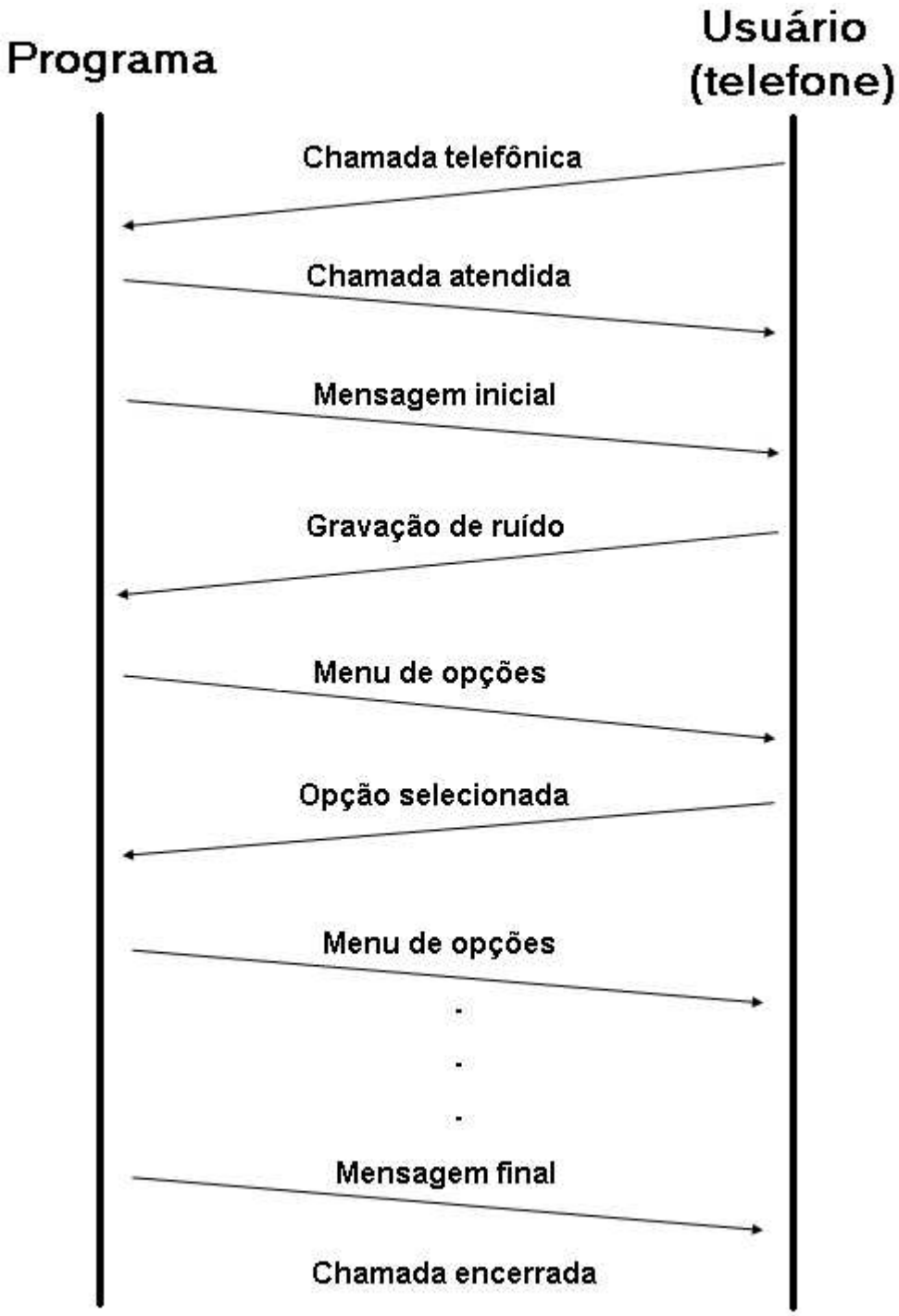

Figura 2.2: Comunicação entre sistema e usuário

Inicialmente, a comunicação deve ser iniciada através de uma chamada telefônica. Quando o programa atende a esta chamada, o sistema pode se comunicar com o usuário através de mensagens de voz. As mensagens enviadas pelo sistema estão armazenadas em arquivos no formato Waveform Audio (Wav) [3, 4]. Quando a chamada é atendida, é enviada

uma mensagem inicial do sistema. Em seguida, o ruído de linha é gravado por um período de tempo, para o cálculo de parâmetros que serão necessários para a detecção da voz. Após a gravação do ruído, uma mensagem correspondente a um menu é transmitida, informando as opções disponíveis. Ao final desta mensagem, o sistema inicia a gravação da voz do usuário através da linha telefônica. O usuário deve, então, falar a palavra que corresponde à opção que ele deseja.

Assim que a gravação da voz termina, o sistema armazena este sinal num arquivo no formato Wav. Em seguida, tem início a etapa de reconhecimento de voz. Esta etapa calcula, dentre as opções disponíveis, a probabilidade de cada uma ter sido falada pelo usuário. Isto é feito comparando-se o sinal de voz gravado com o treinamento do sistema. A palavra com maior probabilidade é reconhecida.

Após o reconhecimento de uma das opções do menu, o sistema envia nova mensagem de voz ao usuário, agora informando as opções do sub-menu cuja categoria foi selecionada anteriormente, e, novamente, a voz do usuário é gravada e reconhecida. As etapas de envio de mensagem de voz, gravação da voz do usuário e reconhecimento são repetidas até que uma das condições de término do sistema seja alcançada. Estas condições são as seguintes:

- Uma das opções finais do sistema foi selecionada pelo usuário, ou seja, a opção selecionada não leva a outro sub-menu. Quando isto acontece, o sistema envia ao usuário uma mensagem de voz com informações correspondentes à categoria selecionada;

- A chamada foi interrompida por algum motivo - a chamada caiu, o usuário desligou o telefone ou qualquer outro motivo.

Em qualquer das duas situações, o sistema finaliza a chamada e volta ao seu estado inicial, monitorando a linha telefônica à espera por novas chamadas.

A estrutura do sistema em camadas é apresentada na Figura 2.3.

A nível de hardware é utilizada uma placa de voice modem, para acesso à linha telefônica. Acima desta placa, há um driver que a controla. Este driver deve ser compatível com a interface TAPI [5, 6], como será descrito no Capítulo 3. Acima do driver, encontra-se o sistema operacional Windows. Na camada acima, localizam-se as interfaces (API's) do Windows. No caso deste trabalho, as interfaces utilizadas são a interface de telefonia (TAPI) e a interface multimídia. Acima destas duas interfaces, encapsulando-as em classes, há a biblioteca TFX [7, 8]. Neste mesmo nível, localiza-se a biblioteca Voice and Sound, utilizada

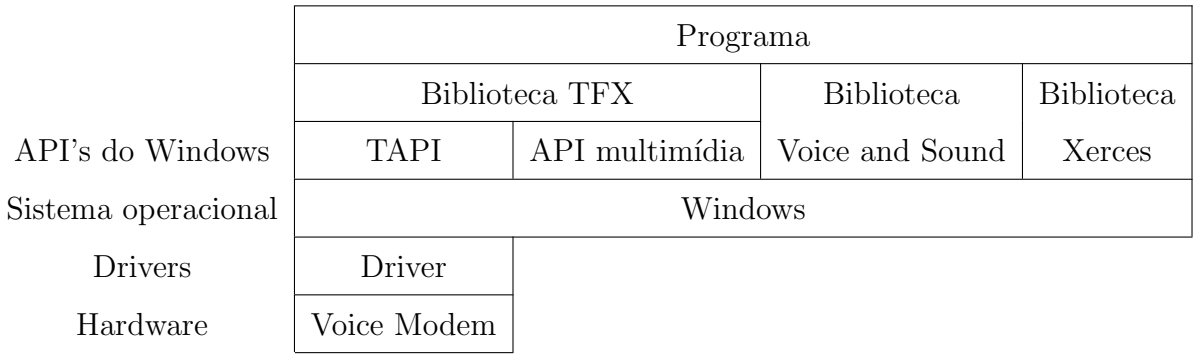

Figura 2.3: Estrutura do sistema em camadas

no reconhecimento de voz, como apresentado no Capítulo 4, e a biblioteca Xerces [9, 10], utilizada para acessar documentos XML [11], como descrito no Capítulo 5. Na camada superior, fica o programa desenvolvido neste trabalho, utilizando as bibliotecas TFX, Voice and Sound e Xerces. A interface do programa é apresentada no Capítulo 6. Com o uso destas bibliotecas, o programa torna-se independente de hardware, em função das diversas camadas localizadas entre os dois.

Capítulo 3

Integração entre programa e linha telefônica

A interface do programa, descrita no Capítulo 6, permite a configuração do sistema e dos menus. Para ter acesso ao voice modem e à linha telefônica, o programa utiliza a interface TAPI e a biblioteca TFX.

Uma das etapas mais importantes no desenvolvimento deste trabalho foi o estabelecimento de uma comunicação entre um programa rodando no computador e uma linha telefônica, que permitisse ao programa ter acesso à linha e manipular chamadas telefônicas. Um dos principais obstáculos para isso foi como configurar a placa de voice modem. Existem diversos modelos de modem disponíveis, de diferentes fabricantes e com diferentes características. A instalação de um modem no computador requer um driver específico para aquele modelo, apesar de o sistema operacional Windows fornecer um driver genérico, que permite o funcionamento de muitos modelos. Assim, a solução utilizada no programa deveria ser capaz de funcionar com qualquer modelo de modem, desde que este fosse um voice modem, ou seja, um modem com capacidade de envio e recepção de voz pela linha telefônica ligada a ele.

Além do driver, existem outras configurações que são necessárias para o acesso ao modem, como a porta de comunicação (COM) onde o modem está instalado - COM1, COM2, COM3 ou COM4. A seleção da porta incorreta pode causar problemas no funcionamento do sistema operacional, pois outros dispositivos, como o mouse, utilizam estas portas para se comunicarem com o computador. Além disso, uma mesma placa de modem, se for instalada em diferentes computadores, pode utilizar diferentes portas de comunicação, ou seja, esta

configuração deveria ser atualizada para cada placa de modem e cada computador que fosse utilizar o programa.

Uma solução mais adequada para o problema seria buscar os drivers e configurações necessários no sistema operacional, pois, a partir do momento que o modem está instalado e funcionando corretamente, o sistema operacional mantém armazenadas todas as informações necessárias para o acesso ao modem. Assim, torna-se desnecessária a configuração manual desta placa, além de manter o sistema compatível com qualquer placa de modem que seja compatível com o sistema operacional Windows. Esta solução é uma API (Application Programming Interface - Interface de Programação de Aplicação) chamada TAPI (Telephony API - API de Telefonia).

3.1 Interface TAPI

A interface TAPI [5, 6] foi desenvolvida pela Intel e pela Microsoft em 1992. Nesta época, a integração entre computador e telefonia (CTI - Computer-Telephony Integration) tornava-se cada vez mais forte, com o uso de computadores para gerenciar linhas telefônicas e call centers. Porém, não havia um padrão. Cada empresa desenvolvia seu próprio hardware de telefonia (modem, PABX, switch) e uma aplicação específica para ele, incompatível com outros modelos e fabricantes. As aplicações desenvolvidas eram compatíveis apenas com hardwares específicos, além de não terem compatibilidade garantida com hardwares desenvolvidos futuramente. O objetivo no desenvolvimento da interface TAPI foi exatamente a definição de um padrão que permitisse o desenvolvimento de aplicações para telecomunicações independentes de hardware, impulsionando o surgimento de novos equipamentos e serviços com CTI.

Com o desenvolvimento da interface TAPI, surgiu o padrão que permitiu a integração entre hardwares e aplicações voltadas para telecomunicações. Assim, qualquer hardware cujo driver fosse desenvolvido com a interface TAPI seria compatível com qualquer aplicação também desenvolvida com esta interface.

Para estimular ainda mais o uso dessa interface, a Microsoft passou a incluí-la em todas as versões de seu sistema operacional Windows. A Tabela 3.1 mostra as versões da interface TAPI incluídas em cada versão do Windows.

Cada versão da interface TAPI é compatível com todas as anteriores. Porém, uma

Tabela 3.1: Versões TAPI em cada versão do sistema operacional Windows

Sistema operacional	**Versão TAPI incluída**	**Última versão compatível**
Windows 3.1x	1.3	1.3
Windows 95	1.4	2.2
Windows 98	2.1	2.2
Windows CE 1.0	1.5	1.5
Windows NT 4.0	1.4	2.2
Windows 2000	2.2	2.2
Windows XP	2.2	2.2

versão da interface nem sempre será compatível com as versões mais recentes. Uma aplicação desenvolvida para a versão 2.0, por exemplo, pode não funcionar num sistema operacional com a versão 1.4 instalada, pois pode estar utilizando alguma funcionalidade que não estava disponível nesta versão. Para o funcionamento correto da aplicação neste exemplo, uma versão mais recente da aplicação poderia ser instalada no sistema operacional, desde que respeitadas as versões compatíveis, conforme mostra a Tabela 3.1.

A Microsoft desenvolveu dois drivers genéricos para modem, chamados Unimodem e Unimodem/V. Estes drivers são baseados na interface TAPI e acompanham o sistema operacional Windows. O Unimodem é compatível com a grande maioria dos modems. Porém, é bastante limitado em termos de funcionalidades, fornecendo suporte apenas para aplicações mais simples. Posteriormente, foi desenvolvido o Unimodem/V. Compatível apenas com voice modems, este driver adiciona diversas funcionalidades não disponíveis no Unimodem, como detecção de dígitos DTMF, atendimento de chamadas e a possibilidade de envio e recepção de voz através da linha telefônica. O Unimodem/V permite todas as funcionalidades utilizadas no programa desenvolvido neste trabalho. Em geral, recomenda-se o uso de drivers desenvolvidos especificamente para a placa utilizada. Porém, na ausência destes, o driver Unimodem/V pode ser utilizado. A primeira distribuição do Windows 95 inclui apenas o Unimodem - porém, o Unimodem/V pode ser instalado separadamente. A partir da distribuição OSR2 do Windows 95, todas as versões deste sistema operacional incluem o Unimodem e o Unimodem/V.

Outra vantagem da interface TAPI é que foi desenvolvida para várias linguagens de programação. Assim, as mesmas funções podem ser utilizadas em C++, Delphi e Visual

Basic, por exemplo, tornando simples a portabilidade de código entre estas linguagens.

Para utilizar a interface TAPI, existem várias funções disponíveis, mas a maioria delas requer vários parâmetros que, muitas vezes, dificultam a programação. Para facilitar o uso da interface TAPI, foi utilizada a biblioteca gratuita TFX [7, 8], desenvolvida na linguagem C++ [12, 13]. Essa biblioteca permite que se tenha controle do estado atual do modem e da linha telefônica, facilitando a programação do sistema. Além disso, como a biblioteca de reconhecimento automático de voz [14] utilizada neste programa também foi escrita em C++, a integração tornou-se muito mais simples.

3.1.1 Características da interface TAPI

A interface TAPI foi desenvolvida com base na Arquitetura de Serviços Abertos do Windows (WOSA - Windows Open Services Architecture). A idéia principal desta arquitetura é fornecer uma interface independente de hardware aos desenvolvedores de aplicações, e uma especificação de driver independente de aplicação aos desenvolvedores de hardware. Com base nesta arquitetura, a estrutura da interface TAPI é baseada em camadas, como mostra a Figura 3.1. A interface TAPI aparece numa camada abaixo das aplicações de telefonia que a utilizam, e acima da interface TSPI (TAPI Service Provider Interface - Interface de Provedor de Serviços TAPI). Esta interface realiza a comunicação direta entre o hardware utilizado - modem, PBX, ISDN - e a interface TAPI. No caso de voice modems, a interface TSPI é o driver Unimodem/V.

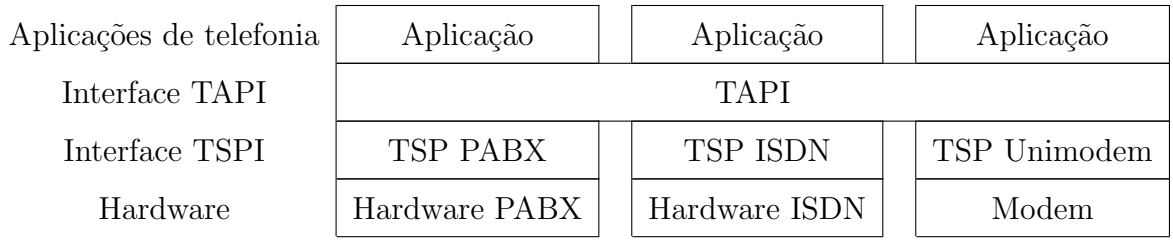

Figura 3.1: Arquitetura da interface TAPI

Assim, tendo como base a arquitetura WOSA, qualquer aplicação desenvolvida com TAPI é compatível com qualquer hardware que utiliza um driver desenvolvido com TSPI.

De acordo com o modelo WOSA, as principais características da interface TAPI são:

Controle de chamadas

Uma chamada telefônica passa por diversos estados desde a discagem até seu encerramento. Entre estes dois estados, há um período em que o terminal que fez a chamada aguarda o atendimento, o outro terminal pode estar ocupado, a chamada pode estar em curso e um dos terminais pode desconectar. As aplicações desenvolvidas com TAPI devem manipular corretamente todos os possíveis estados da chamada.

Acesso a dados através de interfaces padrão

O principal objetivo da interface TAPI é a integração, e, assim, ela deveria ser capaz de manipular diversos tipos de dados. Tendo isto em mente, foram desenvolvidas as classes de dispositivos para cada tipo de dados. Desta forma, a interface TAPI é capaz de verificar se um dispositivo é compatível com determinado tipo de dados. As classes de dispositivos definidas são mostradas na Tabela 3.2.

Tabela 3.2: Classes de dispositivos definidas pela interface TAPI

Classe de dispositivo	**Descrição**
comm	Porta de comunicações
comm/datamodem	Modem acessado através de uma porta de comunicações
comm/datamodem/portname	Nome do dispositivo onde o modem está conectado
wave/in	Dispositivo de entrada de áudio Wav
wave/out	Dispositivo de saída de áudio Wav
midi/in	Dispositivo de entrada Midi
midi/out	Dispositivo de saída Midi
ndis	Dispositivo de rede
tapi/line	Dispositivo de linha
tapi/phone	Dispositivo de fone
tapi/terminal	Dispositivo de terminal

No caso do programa desenvolvido neste projeto, as classes de dispositivo utilizadas são wave/in e wave/out, para recebimento e envio de voz através da linha telefônica, respectivamente.

Independência de rede

A interface TAPI permite a integração de diferentes tipos de redes [2], como PSTN, utilizando o serviço tradicional de telefonia (POTS - Plain Old Telephone Service), PBX (Private Branch Exchange, uma rede telefônica privada), ISDN (Integrated Services Digital Network - Rede Digital de Serviços Integrados), a rede de telefonia celular e até uma rede local de computadores (LAN - Local Area Network). A abstração fornecida pela interface TAPI permite o acesso a diversas funcionalidades dessas redes sem a necessidade que a aplicação tenha conhecimentos específicos relacionados ao dispositivo utilizado. Para o uso de funcionalidades específicas, a interface permite que as aplicações busquem uma descrição completa das capacidades do dispositivo.

Independência de modelo de conexão

A interface TAPI permite a programação de aplicações independentes de como o hardware está conectado ao computador. Os principais modelos de conexão são os descritos abaixo:

- Conexão centrada no telefone

  O ponto central da conexão neste modelo, mostrado na Figura 3.2, é o telefone. A linha telefônica é conectada diretamente ao telefone, e, através de algum tipo de conexão digital, o computador é ligado ao telefone.

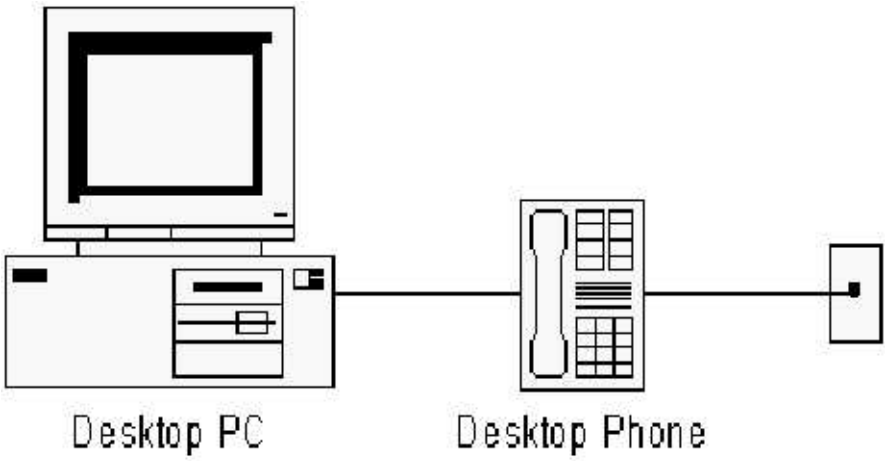

Figura 3.2: Modelo de conexão centrado no telefone

Neste modelo, o telefone realiza o controle das chamadas, podendo atuar independentemente do computador. Este modelo é mais utilizado com telefones que possuem características como identificação e transferência de chamadas.

- Conexão centrada no computador

  Este modelo, mostrado na Figura 3.3, é o mais utilizado em residências. A linha telefônica é conectada diretamente ao computador, através de um modem ou outro hardware de telefonia, e através deste mesmo hardware conecta-se o telefone. O computador passa a ser o ponto central da conexão, e o telefone age simplesmente como dispositivo de entrada e saída para a voz. Neste modelo, o computador pode substituir completamente o telefone, efetuando e recebendo chamadas, através de alguma aplicação desenvolvida com TAPI.

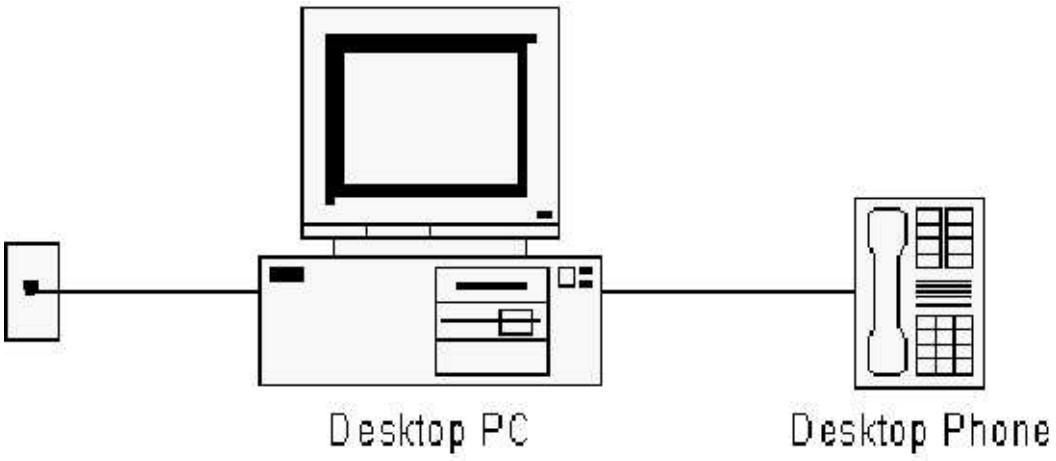

Figura 3.3: Modelo de conexão centrado no computador

- Conexão LAN

  No modelo de conexão LAN, que pode ser visto na Figura 3.4, o computador e o telefone não estão diretamente conectados. Em vez disso, um servidor na rede é conectado a um comutador telefônico (PABX). O computador conecta-se ao servidor através da rede, controlando as chamadas através dele. O servidor passa os comandos recebidos para o comutador, que controla o telefone, geralmente mantido próximo ao computador.

  Neste modelo de conexão, diferentemente dos anteriores, a aplicação não se comunica diretamente com o telefone, e sim com o servidor na rede. Este modelo tipicamente

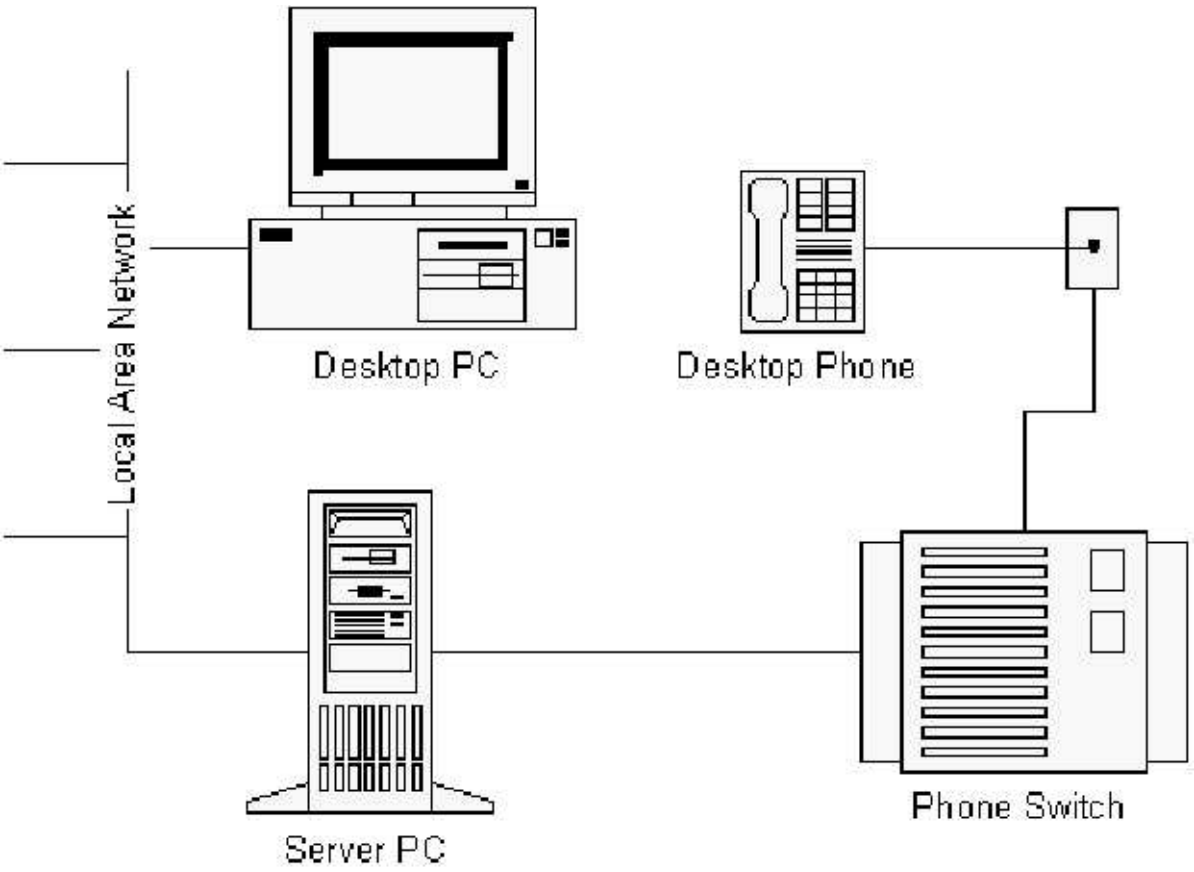

Figura 3.4: Modelo de conexão LAN

limita o acesso a dados na chamada. A banda necessária para a transmissão de dados e voz entre o comutador telefônico e o servidor e entre este e o computador é pequena; porém, quando o acesso aos dados numa chamada é multiplicado pelo número de pessoas utilizando seus telefones através da LAN e somado com o tráfego da rede, normalmente torna-se impraticável fornecer este recurso.

A principal vantagem deste modelo é o fato de não exigir qualquer hardware adicional no computador. Qualquer computador ligado à LAN pode controlar o telefone através de uma aplicação.

- Conexão cliente/servidor

  No modelo de conexão cliente/servidor, mostrado na Figura 3.5, um servidor é conectado a um comutador telefônico através de diversas linhas telefônicas. Assim, este servidor pode realizar o controle das chamadas em todas as linhas.

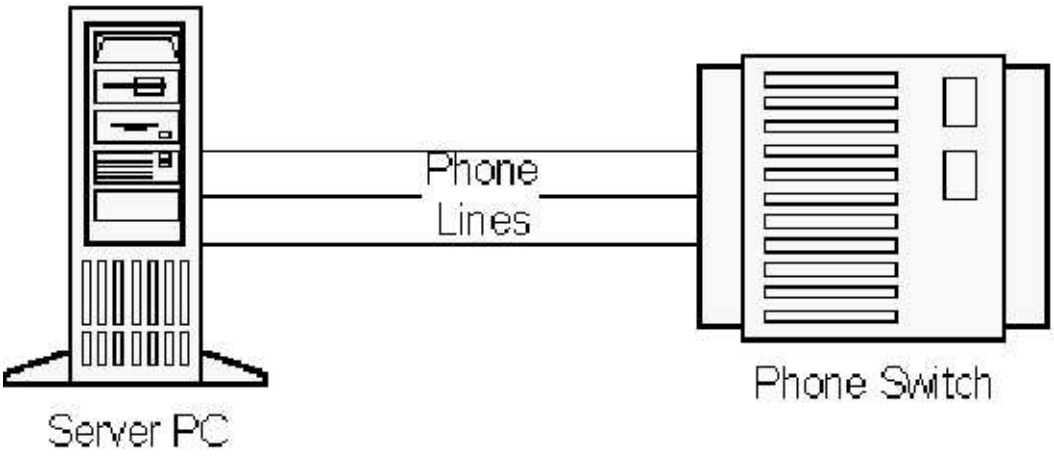

Figura 3.5: Modelo de conexão cliente/servidor

Independência de plataforma (quando possível)

O desenvolvimento da interface TAPI seria, idealmente, um padrão para programação de telefonia compatível com qualquer sistema operacional, o que é, inclusive, um dos princípios do modelo WOSA. Porém, os desenvolvedores da Microsoft e da Intel desenvolveram-no apenas para o sistema operacional Windows, apesar de ser compatível com todas as versões deste, desde o Windows 3.1.

Compartilhamento de linhas entre aplicações

A interface TAPI permite que aplicações diferentes realizem o compartilhamento da linha para diferentes modos de mídia. Assim, uma aplicação pode aguardar um fax enquanto outra realiza uma chamada, por exemplo. Para que esse compartilhamento seja possível, as aplicações devem definir privilégios da chamada, como mostrado na Tabela 3.3.

Uma aplicação com privilégio Owner deve especificar também o modo de mídia desejado. Os modos de mídia e suas aplicações típicas são apresentados na Tabela 3.4. Uma aplicação pode definir mais de um modo de mídia.

Quando uma chamada é recebida, a interface TAPI é responsável por determinar seu modo de mídia e passar o controle da chamada à aplicação que está aguardando chamadas com este modo. Se não houver nenhuma aplicação aguardando o modo de mídia determinado, a chamada é ignorada.

O principal problema na determinação do modo de mídia é no caso de chamadas

Tabela 3.3: Privilégios para compartilhamento da linha telefônica

Privilégio	**Descrição**
None	A aplicação só pode efetuar chamadas, não pode receber
Monitor	A aplicação só pode monitorar as chamadas efetuadas e recebidas
Owner	A aplicação pode efetuar e receber chamadas com o modo de mídia especificado
Owner + Monitor	A aplicação pode monitorar chamadas de outras aplicações

analógicas. Estas chamadas não podem ser identificadas até que sejam atendidas. Porém, uma chamada não pode ser atendida até que seja passada para a aplicação que irá atendê-la e controlá-la. Desta forma, todas as chamadas analógicas são recebidas com modo de mídia Unknown.

No caso do programa desenvolvido neste trabalho, a aplicação especifica o privilégio Owner, para ser capaz de atender a chamada e encerrá-la, e os modos de mídia Automated Voice e Unknown.

3.2 Biblioteca TFX

A interface TAPI é bastante abrangente. Porém, muitas tarefas que devem ser realizadas nos diversos estados do sistema são bastante repetitivas. Por exemplo, na inicialização, devem ser detectados os dispositivos de entrada e saída ligados ao computador. Depois, deve ser verificado qual destes dispositivos tem as capacidades necessárias ao uso do sistema - neste caso, a capacidade de monitorar a linha telefônica, atender a uma chamada telefônica e enviar e receber voz através da linha.

Como estas tarefas são comuns a qualquer aplicação que utiliza a interface TAPI, poucos desenvolvedores a utilizam diretamente. É comum o encapsulamento das funções TAPI em classes, simplificando bastante seu uso. Com este objetivo, foi desenvolvida a biblioteca TFX (Telephony Framework).

A biblioteca TFX foi desenvolvida na linguagem C++, e foi modelada com base nas

Tabela 3.4: Modos de mídia e suas aplicações

Modo de mídia	**Aplicações**
Interactive Voice	Controle de chamadas
Automated Voice	Aplicação automatizada (ex: secretária eletrônica)
Data Modem	Aplicação de comunicações
Group 3 Fax	Recebimento de fax
Group 4 Fax	Recebimento de fax
Digital Data	Aplicação de comunicações ou rede
Telephony Devices for the Deaf (TDD)	Aplicação visual
Teletex	Aplicação visual
Videotext	Aplicação visual
Telex	Aplicação visual
Mixed	Aplicação ISDN
Analog Display Services Interface (ADSI)	Aplicações envolvendo dados e voz
VoiceView	Aplicações envolvendo dados e voz
Unknown	Modo de mídia indeterminado (em geral, chamadas analógicas)

classes MFC (Microsoft Foundation Classes). Isto torna bastante simples o uso da biblioteca em aplicações desenvolvidas com MFC, pois reduz-se o trabalho no desenvolvimento de aplicações com TAPI.

Além de encapsular as funções da interface TAPI, definindo classes correspondentes aos dispositivos utilizados para a integração CTI, a biblioteca TFX adiciona novas funcionalidades, pois encapsula também a API multimídia do Windows, o que torna bastante simples a transmissão de voz pela linha telefônica.

3.2.1 Funções da biblioteca TFX

A biblioteca TFX define diversas funções que permitem a inicialização e o encerramento do sistema. Estas funções não foram encapsuladas em classes porque devem ser utilizadas independentemente das classes que a aplicação utilizar.

Antes de qualquer classe ser utilizada, a biblioteca deve ser apropriadamente iniciali-

zada. Para inicializar a linha telefônica, é utilizada a função TfxLineInitialize:

```
TRESULT TfxLineInitialize (CtAppSink* pAppSink=0, LPCSTR
szAppName=0, HINSTANCE hInst=0);
```

A função TfxLineInitialize chama internamente a função lineInitialize da interface TAPI. O primeiro parâmetro, pAppSink, define a aplicação responsável pela manipulação da linha telefônica. O parâmetro szAppName define o nome da aplicação, e hInst é a instância da aplicação. Os valores padrão dos parâmetros são iguais a zero, o que inicializa a linha sem associar seu uso a uma aplicação específica. O retorno da função é um código que indica se a operação foi realizada com sucesso. Possíveis falhas na inicialização são: a interface TAPI não está instalada no sistema operacional ou outro programa está utilizando a linha.

Quando o programa for encerrado, a linha telefônica deve ser fechada, para permitir que outros programas possam utilizá-la. Se esta etapa não for realizada, o computador deverá ser reinicializado para que o acesso à linha seja liberado. A função responsável por esta etapa é a TfxLineShutdown:

```
void TfxLineShutdown ();
```

A função TfxLineShutdown simplesmente libera os recursos alocados para o uso da linha telefônica, permitindo seu acesso por outros programas. A função não recebe parâmetros nem retorna valores.

Outra função bastante utilizada é a TfxGetNumLines:

```
DWORD TfxGetNumLines ();
```

Esta função não recebe parâmetros, e retorna o número de linhas detectadas no computador. Este valor deve ser maior ou igual a um para que seja possível utilizar o programa desenvolvido neste trabalho.

Além destas, a biblioteca possui outras funções, que, por serem pouco utilizadas, não serão mencionadas.

3.2.2 Classes da biblioteca TFX

Entre as classes da biblioteca TFX, três são as mais importantes, pois definem os principais dispositivos utilizados na integração CTI [15]. Estes dispositivos são: linha, chamada e áudio, representados na biblioteca TFX pelas classes CtLine, CtCall e CtWave, respectivamente.

O dispositivo de linha é o principal dispositivo utilizado nas aplicações. Esta abstração representa não apenas uma linha telefônica, mas qualquer dispositivo fisicamente conectado a uma linha telefônica, como uma placa de voice modem e um telefone. Os dispositivos de linha possuem diferentes características e capacidades, que podem ser descobertas em uma aplicação, e devem ser devidamente abertos antes de serem usados.

A chamada é um dispositivo dinâmico, ao contrário da linha, que é estática. Assim que uma chamada telefônica tem início, o dispositivo chamada é criado e alocado na memória; quando a chamada é encerrada, o dispositivo é destruído, e a memória alocada para ele é liberada.

O dispositivo de áudio modela a transferência de voz pela linha telefônica. Suas principais funções são permitir a reprodução e a gravação de áudio pela linha de maneira simples, abstraindo os detalhes do formato de áudio Wav, que é utilizado como padrão.

A hierarquia das principais classes da biblioteca TFX é apresentada na Figura 3.6.

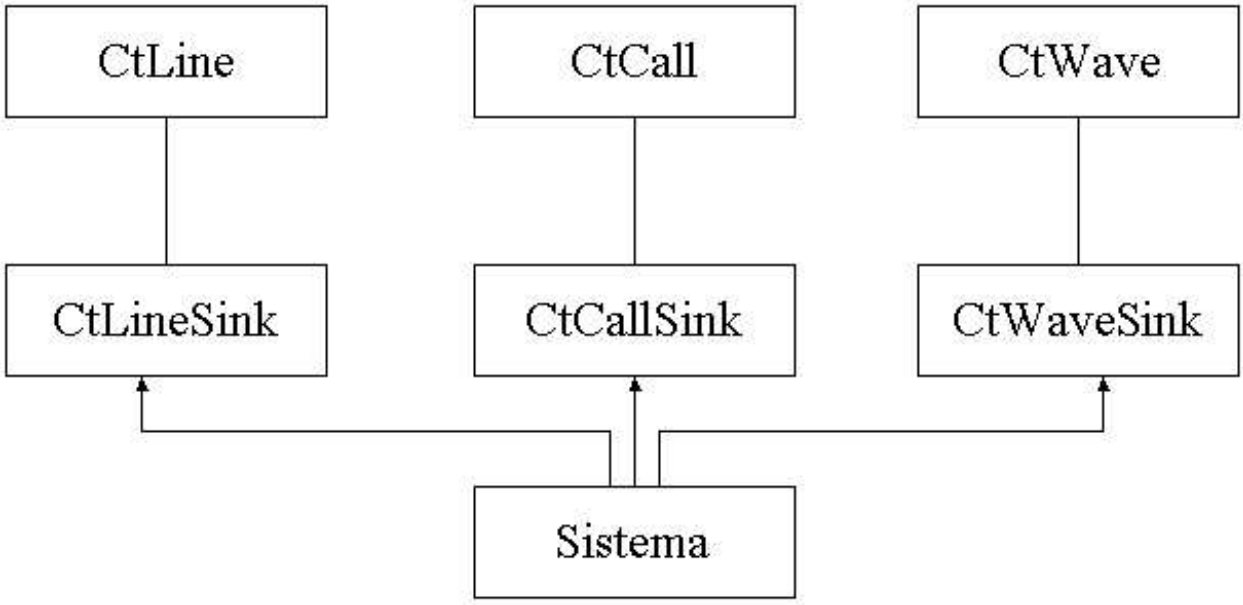

Figura 3.6: Hierarquia de classes da biblioteca TFX

Além das principais classes - CtLine, CtCall e CtWave -, são utilizadas as classes CtLineSink, CtCallSink e CtWaveSink. Estas são classes de interface, que recebem as mensagens de eventos relacionados à linha, à chamada e à transmissão de voz, respectivamente. Desta maneira, sempre que o estado da linha telefônica mudar, por exemplo, uma mensagem informando este evento será enviada pelo sistema operacional. Quando isto ocorre, a classe CtLineSink recebe esta mensagem e chama o método correspondente. As classes CtCallSink

e CtWaveSink trabalham de maneira análoga.

O programa desenvolvido neste trabalho define uma classe principal que é derivada de CtLineSink, CtCallSink e CtWaveSink, herdando todos os métodos destas classes. Desta forma, qualquer evento relacionado à linha telefônica, à chamada ou à transmissão de voz é imediatamente informado ao programa, que pode manipular cada evento de maneira adequada.

3.2.2.1 CtLine

A classe CtLine da biblioteca TFX modela a linha telefônica. Os métodos desta classe permitem abrir, fechar e configurar a linha, além de monitorar o estado da linha, o que permite que o programa saiba quando o estado da linha mudou. A Tabela 3.5 mostra os estados possíveis de uma linha telefônica.

Tabela 3.5: Estados de uma linha telefônica

Estado da linha	**Descrição**
LINEDEVSTATE_BATTERY	Bateria fraca (utilizado em telefones celulares)
LINEDEVSTATE_CAPSCHANGE	As capacidades da linha foram modificadas
LINEDEVSTATE_CLOSE	A linha foi fechada por outra aplicação
LINEDEVSTATE_CONFIGCHANGE	A configuração da linha foi modificada
LINEDEVSTATE_COMPLCANCEL	Identificação cancelada
LINEDEVSTATE_CONNECTED	Conectado
LINEDEVSTATE_DEVSPECIFIC	Indica modificação na configuração específica do dispositivo
LINEDEVSTATE_DISCONNECTED	Desconectado
LINEDEVSTATE_INSERVICE	Indica que a linha está conectada
LINEDEVSTATE_LOCK	A linha está bloqueada
LINEDEVSTATE_MAINTENANCE	A linha está em manutenção
LINEDEVSTATE_MSGWAITOFF	O indicador de mensagens está desligado
LINEDEVSTATE_MSGWAITON	O indicador de mensagens está ligado

Estado da linha	Descrição
LINEDEVSTATE_NUMCALLS	O número de chamadas na linha mudou
LINEDEVSTATE_NUMCOMPLETIONS	O número de chamadas completas mudou
LINEDEVSTATE_OPEN	A linha foi aberta por outra aplicação
LINEDEVSTATE_OTHER	Um item de status não listado foi modificado
LINEDEVSTATE_OUTOFSERVICE	Sem serviço
LINEDEVSTATE_REINIT	Modificações na configuração do dispositivo de linha
LINEDEVSTATE_REMOVED	Indica que o dispositivo foi removido
LINEDEVSTATE_RINGING	Uma chamada está sendo recebida (telefone tocando)
LINEDEVSTATE_ROAM	Mudança no modo de roaming (utilizado em telefones celulares)
LINEDEVSTATE_SIGNAL	Modificação significativa no nível do sinal (utilizado em telefones celulares)
LINEDEVSTATE_TERMINALS	Modificações na configuração do terminal
LINEDEVSTATE_TRANSLATECHANGE	Modificações na configuração da linha

No programa desenvolvido, o principal estado é LINEDEVSTATE_RINGING, que indica que uma chamada está sendo recebida. Neste momento, um objeto da classe CtCall deve ser alocado se a aplicação deseja gerenciar esta chamada.

Um dos métodos mais importantes da classe CtLine é o método Open, responsável por abrir a linha e permitir a comunicação:

```
TRESULT CtLine::Open (DWORD nLineID, CtLineSink* pInitialSink=0,
DWORD dwPrivileges=LINECALLPRIVILEGE_NONE, DWORD
dwMediaModes=LINEMEDIAMODE_INTERACTIVEVOICE);
```

Este método recebe como parâmetros nLineID, um número identificando a linha selecionada - este número deve ser entre 0 e o valor retornado pela função TfxGetNumLines

menos um. Cada número identifica uma linha detectada. Por exemplo, se o computador possui apenas uma linha telefônica, a função acima retornará um, e o parâmetro nLineID deve ser igual a zero. O parâmetro pInitialSink define a classe que irá receber as mensagens de eventos relativos à linha - no caso deste programa, a própria classe definida no programa. Os dois últimos parâmetros definem os privilégios e os modos de mídia.

3.2.2.2 CtCall

A classe CtCall é responsável por gerenciar uma chamada telefônica. Quando uma aplicação inicia, não há um objeto desta classe, pois não há uma chamada telefônica em curso. Um objeto de CtCall é criado para cada chamada recebida que a aplicação irá gerenciar.

Para saber quando uma chamada está sendo recebida, uma aplicação deve analisar o estado da linha telefônica. Quando este estado for igual a LINEDEVSTATE_RINGING, significa que uma chamada está sendo recebida. Se a aplicação deseja gerenciar a chamada, um objeto da classe CtCall é alocado, e o método Answer é responsável por atendê-la:

```
TRESULT CtCall::Answer (LPCSTR psUserInfo=0, DWORD nSize=0);
```

O primeiro parâmetro da função, psUserInfo, é uma string contendo informações sobre a chamada, que são passadas ao usuário que fez a chamada. Para que isto seja possível, a rede deve ter esta capacidade, o que não é o caso da rede pública de telefonia. Desta maneira, este parâmetro não é utilizado no programa desenvolvido. O parâmetro nSize informa o tamanho da string no parâmetro anterior, e também não é utilizado.

Após a alocação de memória para uma chamada, esta pode passar por vários estados, desde o atendimento até o encerramento. A Tabela 3.6 mostra os possíveis estados de uma chamada telefônica.

O estado LINESTATE_OFFERING ocorre quando uma chamada está sendo recebida, ou seja, antes de a chamada ser atendida. Assim que a chamada é atendida, o estado passa para LINESTATE_CONNECTED, o que significa que o programa pode iniciar a comunicação com o usuário que fez a chamada, enviando e recebendo voz pela linha telefônica. O estado LINESTATE_DISCONNECTED indica que a chamada foi interrompida. Logo em seguida, o estado passa para LINESTATE_IDLE. Quando isto ocorre, a memória alocada para a chamada, através de um objeto da classe CtCall, deve ser liberada.

Após os procedimentos de desalocação de memória, o programa pode voltar para seu estado inicial, aguardando novas chamadas.

Tabela 3.6: Estados de uma chamada telefônica

Estado da chamada	**Descrição**
LINECALLSTATE_ACCEPTED	A chamada foi aceita
LINECALLSTATE_BUSY	Chamada com sinal de ocupado
LINECALLSTATE_CONFERENCED	Chamada de conferência
LINECALLSTATE_CONNECTED	Conexão estabelecida
LINECALLSTATE_DIALING	Discando
LINECALLSTATE_DIALTONE	Recebendo tom de discagem
LINEDEVSTATE_DISCONNECTED	Desconectado
LINEDEVSTATE_IDLE	Não há chamada ativa
LINEDEVSTATE_OFFERING	Uma chamada está sendo recebida (telefone tocando)
LINECALLSTATE_ONHOLD	Chamada em estado de hold
LINECALLSTATE_ONHOLDPENDCONF	A chamada está sendo adicionada a uma conferência
LINECALLSTATE_ONHOLDPENDTRANSFER	A chamada está sendo transferida para outro número
LINECALLSTATE_PROCEEDING	Aguardando resposta
LINECALLSTATE_RINGBACK	Aguardando atendimento
LINECALLSTATE_SPECIALINFO	Recebendo uma informação especial (indica algum erro)
LINECALLSTATE_UNKNOWN	Estado desconhecido

3.2.2.3 CtWave

A interface TAPI fornece uma abstração para os objetos de telefonia (linha, fone e chamada telefônica). Através desta abstração, é possível realizar todo o controle e gerenciamento de chamadas. Porém, após o estabelecimento de uma chamada, a interface TAPI não tem responsabilidades sobre a transferência de informações pela linha. Cabe à aplicação ou ao usuário realizar esta tarefa. No caso de uma comunicação por fax, por exemplo, uma aplicação deve enviar ou receber os dados. No caso de uma chamada de voz com modo de mídia Interactive Voice, o próprio usuário é responsável por isso - seja através de um telefone, seja utilizando o próprio computador para receber voz através de caixas de som e enviar

voz através de um microfone. No programa desenvolvido neste projeto, o modo de mídia utilizado é Automated Voice, ou seja, a transferência de voz em uma chamada gerenciada pela aplicação é realizada pela própria aplicação.

Para que uma aplicação possa enviar voz pela linha, é necessário que haja uma mensagem gravada, armazenada em um arquivo. A aplicação deve ser capaz de ler um formato de arquivo de áudio para enviar a voz pela linha. Além disso, quando a voz é recebida pela linha, a aplicação deve ser capaz de armazenar a voz no mesmo formato de áudio.

A biblioteca TFX utiliza a API multimídia do Windows para trabalhar com o formato de áudio Wav. O encapsulamento das chamadas a esta API na classe CtWave abstrai os detalhes do formato Wav ao usuário da biblioteca. Desta maneira, o sistema simplesmente cria um objeto da classe CtWave e utiliza seus métodos para enviar e receber voz. O formato de áudio utilizado é PCM (Pulse Code Modulation) mono (um canal de áudio), com taxa de amostragem de 8 kHz e 16 bits por amostra.

Para poder enviar uma mensagem de voz armazenada em um arquivo no formato Wav, é necessário carregar este arquivo na classe CtWave, através do método Load:

```
TRESULT CtWave::Load (LPCSTR pszFileName);
```

O parâmetro pszFileName especifica o nome do arquivo Wav. Após ser carregado, o arquivo pode ser reproduzido pela linha telefônica, através do método Play:

```
TRESULT CtWave::Play (UINT nWaveOut, bool bLoop=false);
```

O parâmetro nWaveOut especifica o identificador do dispositivo de saída para o qual o áudio será enviado - neste caso, o voice modem. O parâmetro bLoop indica se o áudio deve ser repetido em um loop, até que a transmissão seja interrompida manualmente, ou reproduzido apenas uma vez.

A classe CtWave permite também a gravação de voz pela linha telefônica. Esta etapa é descrita em mais detalhes no Capítulo 4.

3.2.2.4 Classes de interface

As classes CtLineSink, CtCallSink e CtWaveSink são classes de interface. Quando ocorre qualquer evento relacionado à linha telefônica, à chamada e à transferência de áudio pela linha, respectivamente, estas três classes recebem as mensagens correspondentes a estes

eventos, chamando um método adequado para cada evento. Nestas três classes, todos os métodos relacionados a manipulação de eventos são virtuais, ou seja, estas classes não podem ser instanciadas. Qualquer aplicação que deseja manipular eventos relacionados à linha, à chamada ou à transferência de áudio pela linha deve ser derivada das classes de interface correspondentes, herdando suas propriedades. Assim, a aplicação deve definir os métodos que são virtuais nas classes de interface.

A classe CtLineSink define métodos virtuais para quaisquer eventos relacionados à linha telefônica. O método virtual OnLineNewCall é chamado quando a linha recebe uma nova chamada. O método OnLineDevState é disparado quando o estado da linha muda. Um dos parâmetros recebidos por esse método indica o novo estado da linha, que pode ser qualquer estado listado na Tabela 3.5.

Na classe CtCallSink, cada método representa um evento relacionado à chamada telefônica. O método OnCallState informa mudanças no estado da chamada. O novo estado da chamada é recebido como parâmetro neste método, e pode ser qualquer estado apresentado na Tabela 3.6. Outros métodos desta classe trazem informações como detecção e geração de dígitos DTMF.

A classe CtWaveSink é responsável pela interface da aplicação com eventos de transferência de áudio pela linha telefônica. Quando a aplicação envia áudio pela linha telefônica, o evento OnWaveOutOpen informa que o voice modem foi aberto como dispositivo de saída de áudio. O evento OnWaveOutDone é disparado quando o áudio terminou de ser enviado pela linha, e o evento OnWaveOutClose informa que o dispositivo foi fechado. Na gravação de voz pela linha, o evento OnWaveInOpen informa que o voice modem foi aberto como dispositivo de entrada de áudio, e os eventos OnWaveInData e OnWaveInClose são disparados quando a gravação termina e quando o dispositivo é fechado, respectivamente.

No caso de uma aplicação herdando estas três classes, não é necessário definir todos os métodos das classes de interface. Apenas os métodos que interessam, ou seja, os métodos que informam eventos que serão tratados, precisam ser definidos. No caso do programa desenvolvido neste trabalho, os únicos métodos herdados da classe CtLineSink que foram definidos são OnLineDevState, responsável pelo atendimento de uma chamada recebida, e OnLineNewCall, que aloca memória para uma chamada recebida. O único método herdado de CtCallSink que é utilizado é OnCallState, que analisa o estado atual de uma chamada, dando início à comunicação com o usuário em uma nova chamada e desalocando-a após seu

encerramento. Os métodos definidos a partir da classe CtWaveSink são OnWaveOutDone, OnWaveOutClose, OnWaveInData e OnWaveInClose, que informam o status da gravação e da reprodução de áudio através da linha telefônica.

Capítulo 4

Sistema de gravação, detecção e reconhecimento de voz

Para que o sistema possa interagir com o usuário através da linha telefônica, sua voz deve ser gravada através da linha, e, posteriormente, reconhecida, para a seleção das opções de cada menu. Este Capítulo descreve as etapas de gravação, detecção e reconhecimento da voz do usuário do sistema.

A gravação da voz através da linha telefônica utiliza a biblioteca TFX. Esta biblioteca possui a classe CtWave, que permite a transferência de sinais de áudio através da linha. Através desta classe, é possível transmitir um sinal gravado no computador e gravar a voz do usuário.

Para tornar o uso do sistema mais prático, foi implementado um algoritmo de detecção automática de extremos, através do qual é possível detectar em tempo real quando o usuário parou de falar. Assim, é possível interromper a gravação da voz instantaneamente, sem que seja necessário aguardar o término do tempo de gravação.

Para realizar o reconhecimento da voz gravada pela linha telefônica, foi utilizada a biblioteca Voice and Sound. Esta biblioteca foi totalmente desenvolvida na linguagem C++, é multiplataforma e permite o reconhecimento de voz em tempo real ou utilizando arquivos de som pré-gravados. Para o primeiro caso, a biblioteca permite a gravação da voz. Porém, as funções responsáveis pela gravação realizam esta tarefa através da placa de som detectada no computador, ou seja, a biblioteca não permite o uso do voice modem como dispositivo de entrada de áudio. Assim, a biblioteca Voice and Sound foi utilizada apenas no reconhecimento de voz.

4.1 Gravação de voz

A classe CtWave oferece vários métodos para interação com o usuário. Para a gravação da voz recebida pela linha, é utilizado o método Record:

```
TRESULT CtWave::Record (UINT nWaveIn, UINT nSecs);
```

O primeiro parâmetro, nWaveIn, é o identificador do dispositivo de entrada através do qual a voz será gravada - neste caso, o voice modem. O parâmetro nSecs indica o tempo de gravação, em segundos.

Após a gravação, o áudio recebido pela linha é mantido em um buffer na classe CtWave. Para armazená-lo em um arquivo no formato Wav, é utilizado o método Save:

```
TRESULT CtWave::Save (LPCSTR pszFileName);
```

O único parâmetro, pszFileName, especifica o nome do arquivo.

Como a interface TAPI não realiza o gerenciamento dos dados transferidos através da linha telefônica, a biblioteca TFX encapsula, através da classe CtWave, a API multimídia do Windows.

4.1.1 API multimídia do Windows

A API multimídia do Windows [3, 4] permite o uso de dispositivos de áudio - como placas de som ou placas de modem -, aos quais a interface TAPI não tem acesso direto. Essa API, apesar de permitir o controle de qualquer dispositivo de entrada ou saída de áudio, é bastante complexa, pois a realização de tarefas aparentemente básicas, como gravar voz ou reproduzir um arquivo de áudio, exige o uso de diversas funções, o que dificulta a integração da API com programas que a utilizam. Para utilizar a classe CtWave, não é necessário compreender como funciona a API multimídia do Windows, pois esta classe encapsula toda a complexidade, oferecendo ao usuário apenas os métodos necessários. Porém, para a implementação de um algoritmo de detecção de extremos, foi necessário utilizar diretamente as funções da API multimídia.

A arquitetura de áudio do Windows é esquematizada na Figura 4.1. As aplicações Windows utilizam as funções da API, através da biblioteca WINMM. Esta biblioteca acessa o driver de áudio, que cria um novo processo no sistema. Através deste processo, o driver

do kernel é aberto, e este realiza as operações assíncronas de leitura ou escrita no hardware, enviando eventos ao driver de áudio ao completar as operações.

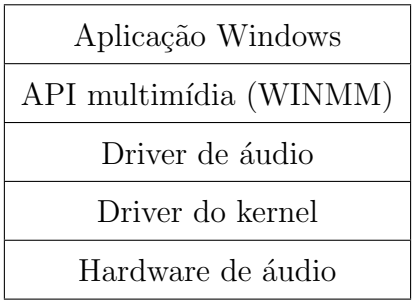

Figura 4.1: Arquitetura de áudio do Windows

A primeira etapa para aquisição de áudio é abrir o dispositivo desejado. As funções waveInOpen e waveOutOpen permitem abrir um dispositivo para entrada e saída de áudio, respectivamente.

```
MMRESULT waveInOpen (LPHWAVEIN phwi, UINT uDeviceID, LPWAVEFORMATEX pwfx,
DWORD dwCallback, DWORD dwCallbackInstance, DWORD fdwOpen);
```

```
MMRESULT waveOutOpen (LPHWAVEOUT phwo, UINT uDeviceID, LPWAVEFORMATEX pwfx,
DWORD dwCallback, DWORD dwCallbackInstance, DWORD fdwOpen);
```

Os parâmetros phwi e phwo são estruturas que irão apontar para o endereço do dispositivo enquanto este estiver aberto. O parâmetro uDeviceID define o índice do dispositivo que se deseja abrir. Os dispositivos disponíveis são numerados de zero ao número de dispositivos menos um. O número de dispositivos de entrada e saída de áudio pode ser obtido através das funções waveInGetNumDevs e waveOutGetNumDevs, respectivamente.

```
UINT waveInGetNumDevs ();
```

```
UINT waveOutGetNumDevs ();
```

O parâmetro pwfx das funções waveInOpen e waveOutOpen é uma estrutura que define o formato do áudio que será enviado ou recebido. Neste trabalho, o formato utilizado é: modulação PCM, 1 canal, taxa de amostragem de 8 kHz e 16 bits por amostra. A abertura de dispositivos de entrada e saída de áudio gera a notificação de eventos WIM_OPEN e WOM_OPEN, respectivamente.

Após a abertura do dispositivo, deve ser preparado um buffer para armazenar os dados que serão recebidos ou enviados. As funções waveInPrepareHeader e waveOutPrepareHeader são responsáveis por isto.

```
MMRESULT waveInPrepareHeader (HWAVEIN hwi, LPWAVEHDR pwh, UINT cbwh);
```

```
MMRESULT waveOutPrepareHeader (HWAVEOUT hwo, LPWAVEHDR pwh, UINT cbwh);
```

Os parâmetros hwi e hwo são as estruturas utilizadas nas funções waveInOpen e waveOutOpen. O parâmetro pwh é uma estrutura que define o tamanho do buffer. Antes de chamar estas funções, deve ser alocado um buffer com o tamanho desejado, em bytes. Como o parâmetro pwfx das funções waveInOpen e waveOutOpen definiu o formato do áudio, é possível calcular o número de bytes necessários para gravar ou reproduzir o tempo desejado. No caso da gravação, este buffer deve estar vazio, para que possa receber dados. Na reprodução, ele deve conter os dados que serão transmitidos.

Após a preparação do buffer, na reprodução de áudio, basta chamar a função waveOutWrite, que irá iniciar a transmissão dos dados do buffer para o dispositivo de saída que foi aberto.

```
MMRESULT waveOutWrite (HWAVEOUT hwo, LPWAVEHDR pwh, UINT cbwh);
```

Após o término da transmissão do buffer, um evento WOM_DONE é notificado.

No caso da gravação, a próxima etapa é enviar o buffer para o dispositivo de entrada, para que este possa utilizá-lo.

```
MMRESULT waveInAddBuffer (HWAVEIN hwi, LPWAVEHDR pwh, UINT cbwh);
```

Após a definição do buffer, a gravação pode ser iniciada. Para isto, é utilizada a função waveInStart.

```
MMRESULT waveInStart (HWAVEIN hwi);
```

Quando o buffer definido pela função waveInAddBuffer é completado, um evento WIM_DATA é notificado à aplicação.

Após o término da gravação ou reprodução, o dispositivo deve ser fechado. Para isto, são utilizadas as funções waveInClose e waveOutClose, respectivamente.

```
MMRESULT waveInClose (HWAVEIN hwi);

MMRESULT waveOutClose (HWAVEOUT hwo);
```

Ao fechar um dispositivo de entrada ou saída, é gerado um evento WIM_CLOSE ou WOM_CLOSE, respectivamente.

A classe CtWave encapsula todas as funções da API multimídia acima descritas. Os eventos notificados pelo dispositivo são encapsulados pela classe de interface CtWaveSink.

4.2 Detecção de voz

Apesar de permitir a gravação da voz pela linha telefônica, a classe CtWave é limitada, pois a gravação é com tempo fixo. Através do parâmetro nSecs do método Record, é calculado o tamanho do buffer necessário para armazenar os dados recebidos. Como a classe trabalha com um canal, taxa de amostragem de 8 kHz e 16 bits por amostra, o tamanho do buffer para gravação de nSecs segundos de voz é igual a:

$$\frac{8000\,amostras}{1\,segundo} \cdot \frac{16\,bits}{1\,amostra} \cdot nSecs\,segundos = 128000 \cdot nSecs\,bits = 16000 \cdot nSecs\,bytes$$

Da maneira em que a classe CtWave foi implementada, um buffer único de 16000 · nSecs bytes é alocado para armazenar a voz. Somente após a gravação de um buffer inteiro a classe de interface CtWaveSink dispara o evento OnWaveInData. Após este evento, os dados armazenados no buffer podem ser acessados pela aplicação. Desta forma, não é possível acessar a voz gravada até que a gravação tenha terminado.

Para sobrepor esta limitação da classe CtWave, foram implementados novos métodos para esta classe, de modo a torná-la mais completa, sem modificar seu funcionamento normal através dos métodos originais.

Como os dados de um buffer só podem ser acessados após este ser completamente utilizado - ou seja, após a notificação do evento WIM_DATA -, a solução utilizada para permitir a deteção em tempo real foi utilizar vários buffers menores, em vez de um único buffer. Assim, em cada momento que o evento WIM_DATA é recebido, um novo buffer pode ser acessado. Conseqüentemente, os dados gravados podem ser analisados.

Para a definição de vários buffers, basta alocar memória para todos os buffers que serão utilizados e chamar as funções waveInPrepareHeader e waveInAddBuffer para cada buffer. O código abaixo exemplifica este processo.

```
// Após abrir o dispositivo

LPSTR *data;
WAVEHDR **waveHdr;
DWORD nDataSize = bufferTime * pFormat->nSamplesPerSec *
            pFormat->wBitsPerSample/8;
waveHdr = new WAVEHDR*[numBuffers];

for (int i=0; i<numBuffers; i++) {
    waveHdr[i] = new WAVEHDR;
    data[i] = (LPSTR) (new BYTE[nDataSize]);
    waveHdr[i]->dwBufferLength = nDataSize;
    waveHdr[i]->lpData = data[i];
    waveInPrepareHeader(hWaveIn, waveHdr[i], sizeof(*waveHdr[i]));
    waveInAddBuffer(hWaveIn, waveHdr[i], sizeof(*waveHdr[i]));
}
```

Desta maneira, é definida uma fila de buffers. Assim que o primeiro buffer é completado, o evento WIM_DATA é notificado - e, conseqüentemente, o método OnWaveInData da classe CtWaveSink. Se houvesse apenas um buffer, a gravação seria encerrada. Porém, se houver outro buffer na fila, o dispositivo automaticamente utiliza-o e continua a gravação. Este processo continua até que a fila esteja vazia ou que a função waveInClose seja chamada. Como a troca de buffers é feita automaticamente pelo dispositivo, sem que o software precise interferir, não há perda de informação entre um buffer e o seguinte, o que permite uma gravação contínua, como se houvesse apenas um buffer.

Com as modificações descritas acima, foi implementado o método RecordBuffers na classe CtWave. Este método é semelhante ao método Record, porém utilizando uma fila de buffers para a gravação.

```
bool CtWave::RecordBuffers(UINT nWaveIn, int buffers, double bufferTime);
```

No método acima, o parâmetro buffers define o número de buffers utilizados, e bufferTime é o tempo que se deseja gravar em cada buffer, em milissegundos.

Com o método RecordBuffers, é possível definir buffers pequenos e acessá-los imediatamente após sua gravação. Assim, foi possível implementar um algoritmo para detecção de extremos, o que permite a interrupção da gravação da voz logo após a detecção de que o usuário terminou de falar.

O algoritmo que foi implementado [14] requer a segmentação do sinal de voz, para obter trechos de som quase estacionários, permitindo a posterior análise da voz. Como os trechos devem ser quase estacionários, não podem ser muito grandes. Em geral, trechos de 20 ms podem ser considerados quase estacionários. Para isto, basta gravar a voz utilizando buffers deste tamanho. Conseqüentemente, cada buffer corresponderá a um segmento de voz.

Com a voz segmentada, é possível realizar a detecção de extremos, também chamada de recorte da palavra. Esta detecção é realizada para retirar os trechos que contêm exclusivamente ruído, e que poderiam prejudicar a taxa de acerto no reconhecimento da palavra.

Para detectar se um trecho do som contém voz, é utilizada como parâmetro a energia das amostras. Se um determinado trecho contém amostras com energia acima de um determinado limiar, pode-se considerar este trecho como vozeado. Se a energia mantém-se próxima de zero, o trecho poderia ser considerado como contendo apenas ruído. Porém, nos fonemas fricativos, a energia mantém-se próxima do ruído ambiente. Assim, se fosse utilizada apenas a energia como parâmetro, estes fonemas poderiam ser identificados como ruído, e palavras que começam ou terminam com estes fonemas seriam recortadas incorretamente.

Para resolver este problema, é utilizado um outro parâmetro, a taxa de cruzamentos por zero. Nos fonemas fricativos, o espectro se concentra nas altas freqüências, e, assim, a taxa de cruzamentos por zero costuma apresentar valores mais altos que no ruído ambiente. Desta forma, este parâmetro permite a distinção entre os fonemas fricativos e o ruído, tornando a detecção de extremos da palavra mais precisa.

O algoritmo utilizado para o recorte calcula os valores de energia e taxa de cruzamentos por zero para o ruído ambiente e para cada trecho de som segmentado. Em seguida, realiza o recorte baseado na comparação dos valores obtidos. A gravação do ruído é realizada no início da chamada, e os parâmetros obtidos são armazenados para comparação com os segmentos gravados posteriormente.

Após a gravação do ruído, são calculados os parâmetros L_{e1}, L_{e2} e L_{zc}. O limiar L_{e1} é um limiar alto de energia, para uma estimativa não muito precisa, distinguindo apenas as interferências de média energia. O limiar baixo de energia (L_{e2}) faz uma estimativa mais

precisa da parte vozeada da palavra, e o limiar de taxa de cruzamentos por zero (L_{zc}) estima possíveis fonemas fricativos no início ou no final de uma palavra.

Com estes parâmetros calculados, o algoritmo de detecção pode ser aplicado. A detecção do início da palavra acontece em três passos. O primeiro passo consiste em comparar a energia de cada segmento com o limiar alto de energia (L_{e1}). Quando forem encontrados cinco segmentos consecutivos com a energia acima deste limiar, o primeiro deles é considerado o início da palavra. Assim, obtém-se uma primeira estimativa.

O segundo passo consiste em comparar os segmentos anteriores àquele que foi considerado como o início da palavra pelo primeiro passo do algoritmo com o limiar baixo de energia (L_{e2}). Todos os segmentos acima deste limiar são considerados como voz. Desta forma, comparando-se do último ao primeiro segmento, o primeiro com energia abaixo do limiar L_{e2} é considerado o último segmento antes da palavra - conseqüentemente, o segmento seguinte a este é considerado o início da palavra, com uma estimativa mais precisa.

O terceiro passo consiste em detectar fonemas fricativos no início da palavra. A partir do último segmento antes do início da palavra estimado no segundo passo, a taxa de cruzamentos por zero de cada segmento é comparada com o limiar L_{zc}. Se a taxa se mantiver abaixo do limiar L_{zc} por três segmentos consecutivos, o último segmento com taxa acima de L_{zc} é considerado o início da palavra, e o algoritmo termina. Se isto não ocorrer nos 25 segmentos anteriores ao segmento marcado como inicial no segundo passo, esta marcação é mantida e o algoritmo é encerrado.

Para a detecção de final da palavra, o processo é análogo. Após a detecção de cinco segmentos consecutivos com energia abaixo de L_{e1}, o último segmento antes destes é marcado como final. A partir daí, a energia dos segmentos é comparada com L_{e2}. Ao encontrar um segmento com energia abaixo de L_{e2}, o anterior é marcado como final. Em seguida, a comparação da taxa de cruzamentos por zero de cada segmento com L_{zc} procura por três segmentos consecutivos com esta taxa abaixo de L_{zc}. Quando encontrar, o último segmento anterior a estes é marcado como o final da palavra.

Com base no algoritmo de detecção de extremos descrito acima, foram implementados mais dois métodos na classe CtWave: noiseRecording, para o cálculo dos parâmetros do ruído, e voiceRecording, que é chamado após o recebimento de cada buffer gravado, ou seja, após o recebimento do evento OnWaveInData, na classe CtWaveSink. O método noiseRecording analisa o ruído gravado pela linha e calcula os limiares L_{e1}, L_{e2} e L_{zc}, e o método VoiceRe-

cording implementa o algoritmo de detecção de extremos. Ao receber um novo buffer, este método é chamado. Quando o início da palavra é detectado, todos os buffers anteriores a ele são descartados. Quando o final da palavra é detectado, a gravação da voz é imediatamente interrompida, e tem início a etapa de reconhecimento de voz.

4.3 Reconhecimento de voz

Para fazer o reconhecimento de voz, foi utilizada a biblioteca Voice and Sound [14]. Esta biblioteca foi desenvolvida em C++ com orientação a objetos, o que tornou bastante simples o uso da biblioteca e sua integração com o sistema desenvolvido neste trabalho. A biblioteca é baseada em coeficientes cepstrais e HMM's (Hidden Markov Models).

Antes de fazer o reconhecimento de voz, é necessário realizar o treinamento do sistema. Isto exige a gravação de cada palavra que faz parte do sistema o maior número possível de vezes. Quanto maior o número de repetições de cada palavra, maior a probabilidade de acerto. A biblioteca Voice and Sound possui funções que realizam o treinamento do sistema a partir de arquivos Wav, gerando como resultado arquivos de centróides e modelos. Estes arquivos devem ser carregados no programa para realizar o reconhecimento de voz.

Na biblioteca Voice and Sound, são definidas três classes: vs_wave, vs_speech e vs_voicewave. A classe vs_wave é responsável pela aquisição do som, além de manipular o formato do som, permitindo alterar-se suas características, como a taxa de amostragem e o número de canais. A classe vs_speech é a responsável pelo processamento e reconhecimento de um sinal de voz. Esta classe permite definir os modelos HMM utilizados, extrair parâmetros do sinal de voz e reconhecer uma palavra baseando-se no treinamento do sistema. A terceira classe, vs_voicewave, herda os métodos e propriedades da classe vs_wave, adicionando funcionalidades específicas ao reconhecimento de voz.

A classe vs_wave é utilizada para a manipulação direta do som. Ela permite a aquisição e reprodução do som, além da alteração de suas propriedades. Porém, como esta classe não permite a aquisição e reprodução do som através de uma placa de voice modem - permite apenas através de uma placa de som do computador -, estas tarefas não são realizadas pela classe vs_wave neste trabalho, e sim pela classe CtWave da biblioteca TFX.

A classe vs_speech realiza o processamento de um sinal de voz, extraindo parâmetros e comparando o sinal, através destes mesmos parâmetros, com os modelos definidos no trei-

namento do sistema. Os métodos desta classe permitem definir parâmetros como número de centróides, estados e modelos utilizados no reconhecimento da palavra, e realizar o reconhecimento de uma palavra, a partir de um treinamento realizado anteriormente.

A classe vs_voicewave possui métodos voltados para a manipulação da voz. Como é derivada da classe vs_wave, herda todos os métodos desta. Os métodos implementados na classe vs_voicewave permitem definir o tamanho da janela utilizado na segmentação do sinal de voz e realizar o recorte da palavra em tempo real, detectando seus extremos. O algoritmo de detecção é o mesmo que foi implementado neste trabalho, porém a classe vs_voicewave só permite utilizá-la em tempo real na gravação a partir de um microfone. Por isso, a detecção de extremos neste trabalho é realizada pela classe CtWave.

4.3.1 Utilizando a biblioteca Voice and Sound

A utilização da biblioteca Voice and Sound é bastante simples, pois todas as funcionalidades estão encapsuladas em classes. Desta forma, não é preciso compreender a teoria de reconhecimento de voz para utilizá-la, bastando saber utilizar os métodos adequados de cada classe.

Para utilizar a biblioteca Voice and Sound neste trabalho, foi implementado um método no programa principal que, chamado após o término da gravação da voz, cria uma instância da classe vs_speech e uma instância da classe vs_voicewave. Não é necessário instanciar a classe vs_wave.

Primeiramente, é necessário abrir o arquivo Wav no qual a voz foi gravada, utilizando a classe vs_voicewave:

```
vs_voicewave myvoicewave (1, 8000, bufferTime);
```

```
myvoicewave.open (fileName);
```

No construtor da classe vs_voicewave, os parâmetros são o número de canais de áudio, a taxa de amostragem, em Hz, e o tempo de gravação de um segmento, em segundos. Neste trabalho, o tempo utilizado para o segmento é o mesmo do buffer de gravação - por padrão, 0,02 s. O método open recebe como parâmetro o nome do arquivo Wav que será aberto.

Em seguida, os arquivos gerados no treinamento do sistema devem ser carregados na classe vs_speech:

```
vs_speech myspeech;
```

```
myspeech.set_hmm (128, 4, numWords, centrosFileName, modelosFileName);
```

O método set_hmm da classe vs_speech é responsável por carregar os arquivos de treinamento. Os dois primeiros parâmetros são o número de centróides e de estados. Estes valores devem ser os mesmos que foram utilizados no treinamento. O terceiro parâmetro é o número de palavras que fazem parte daquele treinamento, ou seja, o número de palavras que compõem aquele menu. Os dois últimos parâmetros são os nomes dos arquivos de treinamento com os centróides e os modelos, respectivamente.

Após carregar o arquivo Wav com a voz e os arquivos de treinamento, a biblioteca é capaz de fazer o reconhecimento de voz:

```
short resposta = myspeech.recognize(myvoicewave);
```

O método recognize da classe vs_speech recebe como parâmetro o objeto da classe vs_voicewave que contém as amostras do sinal de voz, e retorna o índice da palavra que foi reconhecida. Este índice é baseado na ordem na qual os arquivos Wav de cada palavra foram carregados no treinamento.

Com o índice da palavra retornado pelo método acima, é possível saber qual foi a palavra reconhecida pelo sistema, e assim, selecionar a opção correspondente no menu. Este processo é repetido a cada palavra gravada pelo sistema.

Capítulo 5

Sistema de menus

Para facilitar o acesso às informações, as opções disponíveis no sistema desenvolvido neste trabalho são estruturadas em menus, com as opções agrupadas em menus específicos. Para simplificar a manipulação e o gerenciamento, a estrutura de menus foi organizada em um documento XML, e uma opção do programa permite a edição deste documento. Este Capítulo descreve a organização da estrutura de menus em arquivos XML e a manipulação destes arquivos no sistema.

5.1 A linguagem XML

A organização da estrutura de menus é extremamente importante, pois com um grande número de opções disponíveis, torna-se difícil gerenciar os menus no programa e acessar as opções durante seu uso. Por isso, optou-se por organizar esta estrutura utilizando um arquivo XML.

O XML (Extensible Markup Language) [11] é uma linguagem aberta que surgiu, inicialmente, como um complemento à linguagem HTML (HyperText Markup Language). A estrutura da linguagem HTML trata da formatação das informações, ou seja, da maneira como as informações contidas em um documento serão apresentadas visualmente. A linguagem HTML não trata da estruturação das informações, ou seja, da maneira na qual os dados contidos em um documento são organizados.

Analogamente ao HTML, o XML trata apenas da organização do conteúdo de um documento, não preocupando-se com a formatação dos dados - esta tarefa cabe à aplicação que irá utilizar o XML, seja um arquivo HTML, seja um programa em C++. A principal

vantagem disto é que um mesmo documento pode ser utilizado de maneiras diferentes em aplicações diferentes, sem que um interfira no outro - desde que seja mantida a estrutura do arquivo XML.

Um dos principais objetivos da linguagem XML foi tornar esta linguagem simples, e ao mesmo tempo mantê-la poderosa. Sua habilidade para organizar, descrever e estruturar dados pode ser utilizada em diversas situações. Um arquivo XML pode ser utilizado por diferentes aplicações, como um formato comum entre elas. O uso de XML num arquivo HTML permite separar o conteúdo da apresentação visual - e, assim, a atualização das informações pode ser feita no arquivo XML sem modificar o arquivo HTML.

No surgimento do XML, seus criadores definiram uma série de objetivos para esta linguagem. Entre eles, permitir o uso direto da linguagem na Internet, suportar diversas aplicações, facilitar a criação de documentos e ser uma linguagem clara, o que permite a leitura de um documento tanto por programas quanto por pessoas, devido à simplicidade da estrutura da linguagem.

5.2 Interfaces XML

As interfaces do XML permitem o acesso a um documento XML em uma aplicação, sem que esta tenha que manipular diretamente o arquivo. Um mesmo arquivo XML pode ser manipulado por diferentes interfaces, dependendo da aplicação. As interfaces podem ser de dois tipos: orientadas a eventos ou orientadas a modelos de objetos.

Numa interface orientada a eventos, conforme cada elemento da estrutura XML é lido, ocorre a notificação de um evento. Normalmente, há um tipo de evento para cada tipo de elemento encontrado na estrutura. O evento recebido deve tratar da manipulação daquele elemento específico. Assim, conforme a estrutura de um arquivo XML vai sendo analisada pela interface, a aplicação correspondente recebe eventos que notificam o tipo de elemento encontrado. A interface orientada a eventos mais utilizada é o SAX (Simple API for XML).

Uma interface orientada a modelos de objetos trata o documento de maneira diferente. Neste tipo de interface, o documento é primeiramente lido em sua totalidade e representado através de objetos. Após isto, a aplicação pode manipulá-lo através dos objetos que foram criados. A principal interface deste tipo é o DOM (Document Object Model).

A escolha do tipo de interface XML depende de como se deseja utilizar o documento.

Os dois tipos apresentam vantagens e desvantagens. As interfaces orientadas a eventos são mais simples e consomem menos memória, porém são limitadas a uma leitura seqüencial do documento, o que pode não ser adequado em determinadas situações. As interfaces orientadas a modelos de objetos têm como vantagem o fato de armazenarem a estrutura inteira do documento em objetos, e pemitirem sua manipulação livremente. Porém, como toda a estrutura é mantida na memória, aumentam o tempo de processamento e o consumo de memória, o que pode ser crucial em aplicações que utilizam documentos muito grandes.

No sistema desenvolvido neste trabalho, a estrutura de menus mantida num arquivo XML não é acessada seqüencialmente. Como depende das opções escolhidas pelo usuário, não é possível saber previamente qual será o próximo elemento que deve ser lido, podendo ser, inclusive, um elemento anterior ao atual na estrutura. Desta forma, optou-se por utilizar o tipo de interface orientada a modelos de objetos, o que permite a livre manipulação do documento durante a execução do sistema.

5.3 Estrutura de menus do sistema em XML

Conforme descrito anteriormente, a linguagem XML foi escolhida para organizar a estrutura de menus do programa por ser uma linguagem simples para criar, organizar, estruturar e gerenciar documentos. Assim, o acesso às opções dos menus no programa tornou-se simples. Além disso, também foi possível criar uma opção no programa para manutenção da estrutura de menus.

A Figura 5.1 apresenta a estrutura de menus que foi criada para este sistema.

A organização dos menus é em uma estrutura de árvore, para a qual a linguagem XML é bastante adequada. A estrutura da linguagem XML utiliza tags semelhantes ao HTML. Porém, os tags do XML identificam o tipo de informação contida no documento. Abaixo é apresentado um trecho do arquivo XML descrevendo a estrutura de menus da Figura 5.1.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<menuRoot>
    <soundFile>C:\mensagem.wav</soundFile>
    <menu>
        <soundFile>C:\menu1.wav</soundFile>
        <centrosFile>C:\centros1.dat</centrosFile>
        <modelosFile>C:\modelos1.dat</modelosFile>
```

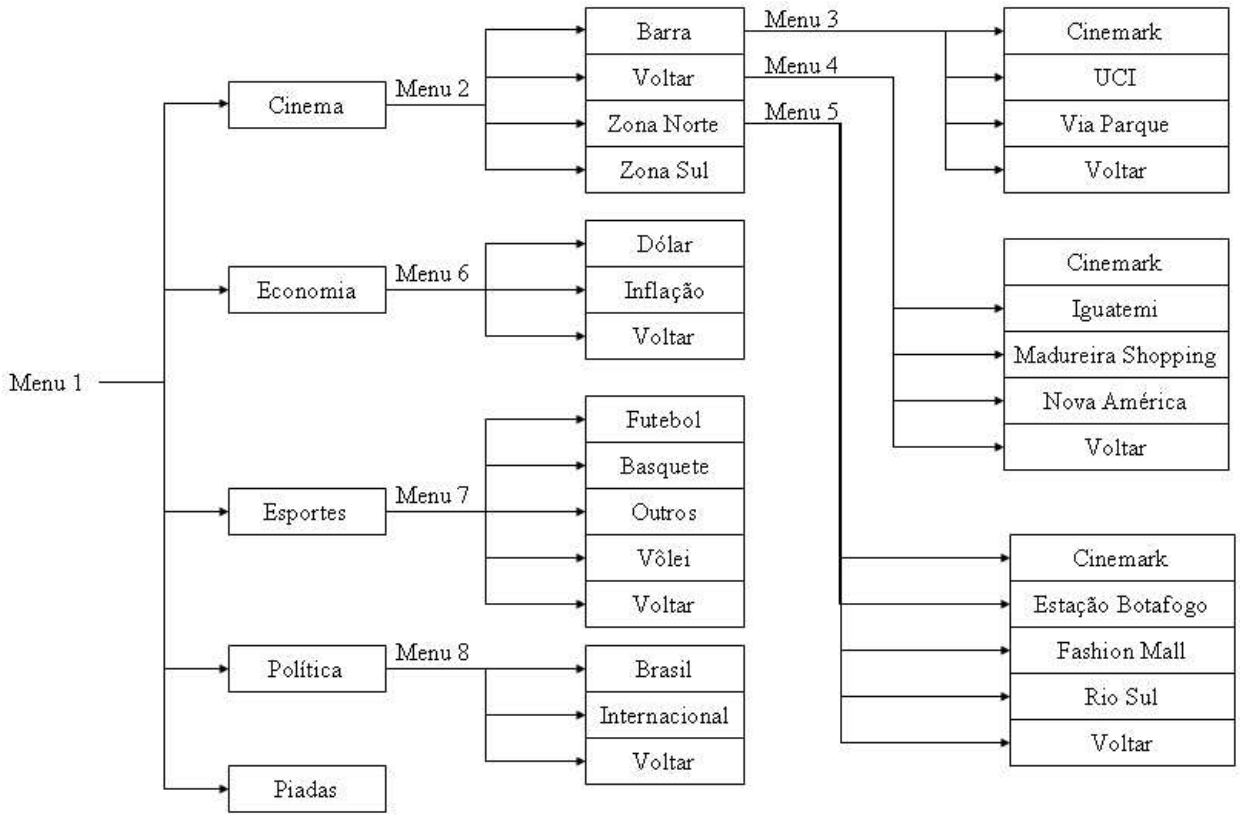

Figura 5.1: Estrutura de menus do sistema

```
<menuItem name="Cinema">
    <menu>
        <soundFile>C:\menu2.wav</soundFile>
        <centrosFile>C:\centros2.dat</centrosFile>
        <modelosFile>C:\modelos2.dat</modelosFile>
        <menuItem name="Barra">
            <menu>
                <soundFile>C:\menu3.wav</soundFile>
                <centrosFile>C:\centros3.dat</centrosFile>
                <modelosFile>C:\modelos3.dat</modelosFile>
                <menuItem name="Cinemark">
                    <soundFile>C:\cinemark.wav</soundFile>
                </menuItem>
                <menuItem name="UCI">
                    <soundFile>C:\uci.wav</soundFile>
                </menuItem>
```

```
                    <menuItem name="Via Parque">
                        <soundFile>C:\via_parque.wav</soundFile>
                    </menuItem>
                    <menuItem name="Voltar">
                        <Voltar/>
                    </menuItem>
                </menu>
            </menuItem>

                    .
                    .
                    .

        <menuItem name="Piadas">
            <soundFile>C:\piadas.wav</soundFile>
        </menuItem>
        <menuItem name="Política">
            <menu>
                <soundFile>C:\menu8.wav</soundFile>
                <centrosFile>C:\centros8.dat</centrosFile>
                <modelosFile>C:\modelos8.dat</modelosFile>
                <menuItem name="Brasil">
                    <soundFile>C:\brasil.wav</soundFile>
                </menuItem>
                <menuItem name="Internacional">
                    <soundFile>C:\internacional.wav</soundFile>
                </menuItem>
                <menuItem name="Voltar">
                    <Voltar/>
                </menuItem>
            </menu>
        </menuItem>
    </menu>
</menuRoot>
```

A primeira linha do arquivo XML informa que se trata de um arquivo XML, identificando a versão e a codificação. As linhas seguintes descrevem o conteúdo do documento.

No primeiro nível da estrutura, há um tag menuRoot, que identifica o menu raiz da estrutura. O menuRoot contém um tag SoundFile, que identifica o nome do arquivo Wav com a mensagem inicial do programa - a mensagem que é reproduzida no momento em que uma nova chamada é atendida pelo sistema. O tag menuRoot contém também um menu, que é o menu principal do sistema.

Cada tag menu deve ter um tag SoundFile, indicando o nome do arquivo Wav com a mensagem de voz que apresenta as opções daquele menu, além de um tag centrosFile e um tag modelosFile, que indicam os nomes dos arquivos de centróides e de modelos, respectivamente, que são criados no treinamento do sistema, conforme descrito no Capítulo 4. Estes arquivos são necessários para o reconhecimento de voz. Cada menu é composto por diversos tags menuItem, sendo cada menuItem uma opção.

Cada tag menuItem deve ter um atributo name que o identifica. Além disso, um menuItem pode ter outro menu. Os menuItems que não contêm menu podem ser as opções finais do sistema, e, neste caso, ter um SoundFile, que é o nome do arquivo Wav que contém a mensagem de voz correspondente a esta opção, ou ter um tag vazio Voltar, que indica que a seleção desta opção retorna ao menu anterior.

5.4 Utilizando XML no sistema

Para simplificar o acesso e a manipulação de arquivos XML no sistema, foi utilizada a biblioteca Xerces [9, 10]. Esta biblioteca fornece todas as funções para ler, escrever e manipular dados em XML. Esta biblioteca tem como principais vantagens ser gratuita e multiplataforma, além de possuir versões em diversas linguagens, como C++, Java e Perl.

A biblioteca Xerces permite o uso das interfaces SAX, orientada a eventos, e DOM, orientada a modelos de objetos. Conforme descrito anteriormente, optou-se por utilizar uma interface orientada a modelos de objetos. Diversas classe na biblioteca Xerces encapsulam os tipos de objetos do DOM. A classe DOMDocument representa o documento XML inteiro. Através dela, é possível acessar todos os dados do documento, assim como criar um novo documento. A classe DOMNode representa um nó da estrutura de árvore, ou seja, um elemento do documento. Cada tag, atributo, texto ou comentário em um arquivo XML

é representado por um DOMNode, que pode possuir filhos, organizando-se, assim, uma estrutura de árvore.

Para integrar a biblioteca Xerces com o programa desenvolvido, foi criada a classe VBMenu, que torna ainda mais simples o acesso ao XML no programa.

5.4.1 A classe VBMenu

A classe VBMenu utiliza a biblioteca Xerces, e encapsula o acesso às suas classes, realizando todo o processamento específico ao documento criado para armazenar a estrutura de menus, e fornecendo ao programa os métodos necessários para percorrer os nós da árvore, acessando os dados relativos a cada nó.

Além da biblioteca Xerces, a classe VBMenu utiliza a biblioteca VBLib [13], que define, entre outras, a classe de string VBString.

Os principais métodos da classe VBMenu utilizados no programa são apresentados a seguir. Para carregar um arquivo XML na classe VBMenu, é utilizado o método loadMenu:

```
void VBMenu::loadMenu (const char* menuFileName);
```

O parâmetro menuFileName indica o nome do arquivo XML. Para descarregar um arquivo, liberando os recursos da classe, utiliza-se o método unloadMenu:

```
void VBMenu::unloadMenu ();
```

Tendo um documento XML carregado, o método execute é utilizado para processar o menu atual:

```
int VBMenu::execute ();
```

O método execute lê os nós do menu atual, armazenando o número de itens e os nomes dos arquivos de voz, centróides e modelos deste menu. O valor de retorno deste método é o número de itens do menu atual.

Para acessar o nome de um menuItem no menu atual, utiliza-se o método getMenu-ItemName:

```
VBString VBMenu::getMenuItemName (int index);
```

O parâmetro index é o índice do menuItem no menu atual, de zero até o número de menuItems menos um. O valor retornado é uma string com o nome do menuItem.

Para verificar se um item de um menu é uma opção para voltar ao menu anterior - ou seja, para verificar se um tag menuItem contém um tag Voltar - é utilizado o método isMenuItemVoltar:

```
bool VBMenu::isMenuItemVoltar (int index);
```

O parâmetro index é o índice do menuItem. O método retorna true se o menuItem contém um tag Voltar, e false em caso contrário.

Quando um menuItem selecionado corresponde à opção de voltar ao menu anterior, utiliza-se o método goToParentMenu:

```
void VBMenu::goToParentMenu ();
```

Com o método acima, retorna-se ao menu anterior, ou seja, retorna-se ao menu pai do menu atual na estrutura de árvore.

Se um menuItem não for a opção voltar, o método utilizado é o goToMenuItem:

```
void VBMenu::goToMenuItem (int index);
```

O parâmetro index define o índice do menuItem selecionado. Após utilizar o método goToParentMenu ou goToMenuItem, basta utilizar o método execute, já apresentado, para processar o novo menu.

Após o processamento de um menu, é possível acessar os nomes dos arquivos correspondentes a este menu. Os métodos utilizados para isto são os seguintes:

```
VBString VBMenu::getSoundFileOfCurrentMenuItem ();

VBString VBMenu::getCentrosFileOfCurrentMenuItem ();

VBString VBMenu::getModelosFileOfCurrentMenuItem ();
```

Os três métodos acima retornam os nomes dos arquivos Wav, de centróides e de modelos, respectivamente.

Para acessar o número de itens do menu atual, é utilizado o método getNumberOfItemsInCurrentMenu:

```
unsigned VBMenu::getNumberOfItemsInCurrentMenu ();
```

O método isMenuItemFinal é utilizado para saber se o menuItem atual é uma opção final do sistema - ou seja, um menuItem que possui um tag soundFile, mas não possui os tags menu e Voltar:

```
bool VBMenu::isMenuItemFinal ();
```

A classe VBMenu possui muitos outros métodos. Porém, os acima apresentados são os métodos utilizados para percorrer a estrutura de menus do documento e acessar todos os dados necessários no programa. Os demais métodos são mais específicos, utilizados apenas na edição de um documento XML.

A utilização de um documento XML no sistema, como descrito na acima, necessita de um arquivo XML completo, com os menus, itens e nomes de arquivos utilizados. Para criar um novo documento XML ou editar um documento existente, foi criada uma opção no programa que permite realizar todas as operações necessárias. Esta opção apresenta visualmente a estrutura de menus em uma árvore, e é descrita no Capítulo 6 em mais detalhes.

Capítulo 6

Interface do programa

O programa desenvolvido neste trabalho possui diversas opções necessárias ao funcionamento do sistema. Este Capítulo descreve com detalhes a interface e as opções disponíveis no programa.

A interface do programa foi desenvolvida utilizando MFC (Microsoft Foundation Classes) [3, 16]. Como esta biblioteca é padrão da Microsoft, a integração do programa com as interfaces de telefonia e multimídia do Windows, descritas nos Capítulos 3 e 4, respectivamente, tornou-se bastante simples.

6.1 Tela principal do programa

A Figura 6.1 mostra a tela principal do programa, que é exibida assim que este é executado.

A tela principal mostra um controle do tipo List Box no centro, que funciona como um log do programa. Este controle apresenta mensagens informativas, de acordo com o funcionamento do sistema. Qualquer mensagem relativa à chamada - como chamada recebida, atendida ou desconectada -, ao reconhecimento de voz - como as opções selecionadas em cada menu -, ou à gravação ou reprodução de voz é exibida neste controle. A barra de status na parte de baixo da tela mostra a mensagem "Desconectado", que é o estado inicial do programa. Quando o voice modem é aberto, a barra de status mostra "Aguardando chamadas", e quando uma chamada está em curso, a mensagem varia de acordo com o status atual do sistema.

Na parte superior da tela, há quatro menus com as opções do programa, que serão

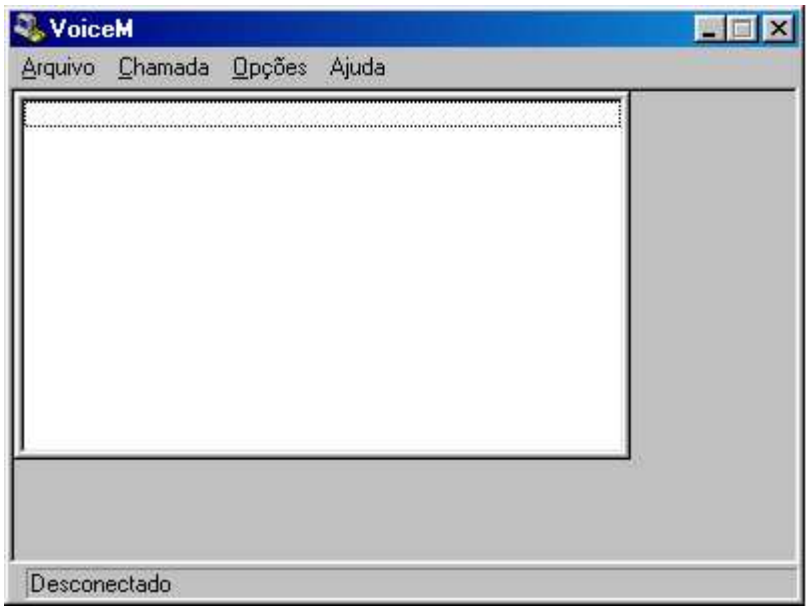

Figura 6.1: Tela principal do programa

descritas com mais detalhes nas próximas Seções.

6.1.1 Menu Arquivo

O menu Arquivo oferece opções relativas a arquivos de entrada e saída do programa. A Figura 6.2 mostra as opções do menu Arquivo.

Figura 6.2: Opções do menu Arquivo

A primeira opção do menu Arquivo é “Selecionar Arquivo XML”. Esta opção abre uma tela para seleção do arquivo XML que será carregado. Ao selecionar um arquivo nesta tela, este arquivo é carregado na classe VBMenu, e quando uma chamada for atendida pelo programa, os dados carregados serão utilizados. Isto permite que sejam criados diversos arquivos XML com diferentes estruturas de menus. Para alternar entre arquivos, basta selecionar o arquivo desejado nesta opção. O programa só permite ao usuário abrir a linha telefônica para atender chamadas depois que um arquivo XML tiver sido carregado. Quando uma chamada está em curso, esta opção fica desabilitada no menu Arquivo.

A opção “Salvar Log” permite armazenar as mensagens do log em um arquivo. Ao selecioná-la, surge uma tela que permite selecionar o arquivo de destino, onde as mensagens serão armazenadas. O arquivo armazena também a data da última chamada atendida. Se o log estiver vazio, a seleção desta opção exibirá a mensagem “Nenhuma mensagem no Log”.

A opção “Limpar Log” apaga todas as mensagens do log. Esta opção é útil quando muitas chamadas foram atendidas pelo programa, pois o log pode estar muito cheio, dificultando a visualização das últimas mensagens.

A última opção do menu Arquivo é “Sair”, que é utilizada para encerrar o programa. Quando esta opção é selecionada, se uma chamada estiver em curso, ela é encerrada; a linha é fechada, caso esteja aberta, e as configurações do programa são armazenadas numa chave no registro do Windows. Em seguida, são realizadas todas as operações de liberação de memória, desalocando-se todos os objetos utilizados no programa.

6.1.2 Menu Chamada

O menu Chamada oferece opções relativas ao gerenciamento da linha telefônica e da chamada. A Figura 6.3 mostra as opções do menu Chamada.

A opção “Abrir linha” permite a seleção um dispositivo de linha para que o programa possa receber chamadas telefônicas. Esta opção é descrita em mais detalhes na Seção 6.3.

A segunda opção do menu Chamada é “Fechar linha”. Esta opção fica inicialmente desabilitada, pois quando o programa tem início, a linha telefônica está fechada. Após a abertura da linha, esta opção fica disponível. A seleção desta opção encerra uma chamada, caso haja uma chamada atual, e em seguida, libera os recursos alocados para o dispositivo de linha. Após isto, esta opção volta a ficar desabilitada, e a opção “Abrir linha” fica novamente disponível.

Figura 6.3: Opções do menu Chamada

A última opção do menu Chamada é "Finalizar chamada". Se o programa estiver atendendo uma chamada atualmente, esta opção estará habilitada, e irá encerrar a chamada. Em seguida, o sistema volta a aguardar novas chamadas. Esta opção fica desabilitada quando não há uma chamada em curso.

6.1.3 Menu Opções

O menu Opções oferece configurações relativas ao sistema. A Figura 6.4 mostra as opções deste menu.

A opção "Edição de Menus XML" permite ao usuário editar um documento XML para armazenar uma estrutura de menus. Esta opção é descrita com mais detalhes na Seção 6.2.

A opção "Configurações" mostra uma tela como a Figura 6.5.

A tela de configurações do programa é dividida em duas partes: configurações básicas e configurações avançadas. As configurações básicas são relativas à chamada. É possível definir o número de toques que o sistema deve aguardar antes de atender uma chamada - o valor padrão é um toque - e o tempo máximo de gravação da voz do usuário através da linha telefônica. Ao gravar a voz do usuário, caso o algoritmo de detecção de extremos não consiga detectar o início ou o fim da palavra, a gravação será interrompida após este tempo.

Figura 6.4: Opções do menu Opções

Figura 6.5: Tela de configurações do programa

Além de digitar o valor, é possível selecionar em uma Combo Box a unidade, segundos ou milissegundos. O valor padrão é de 2 s.

As configurações avançadas do programa permitem definir o tempo de gravação do ruído e o tamanho do buffer de gravação. Ambos permitem digitar um valor e selecionar a unidade, segundos ou milissegundos. O tempo de gravação de ruído é o tempo em que o sistema grava o áudio através da linha no início de uma chamada, após reproduzir a

mensagem inicial e antes de reproduzir a mensagem relativa ao primeiro menu. A gravação do ruído é necessária para o cálculo de parâmetros necessários para a detecção de extremos da palavra, como descrito em mais detalhes no Capítulo 4. O valor padrão é de 1 s. O tamanho do buffer de gravação define o tamanho de cada buffer utilizado na gravação de voz. Este tamanho é o mesmo utilizado para o segmento no algoritmo de detecção de extremos. O valor padrão é de 20 ms.

As opções definidas na tela de configurações são armazenadas no registro do Windows depois que o programa é encerrado. Na próxima vez que o programa for executado, estas configurações serão carregadas a partir do registro.

6.1.4 Menu Ajuda

As opções do menu Ajuda são apresentadas na Figura 6.6.

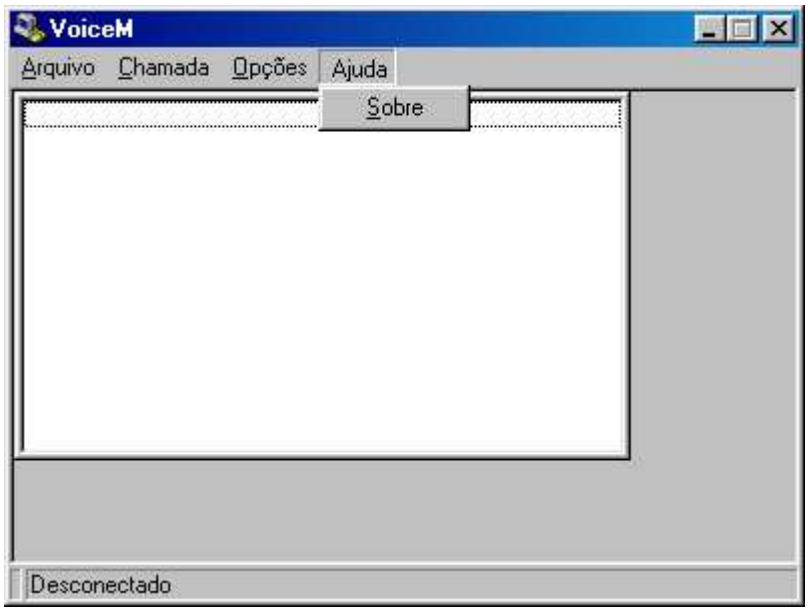

Figura 6.6: Opções do menu Ajuda

O menu Ajuda mostra apenas a opção "Sobre", que exibe informações relativas ao programa, como versão e autor.

6.2 Edição de Menus

A opção "Edição de Menus XML", no menu Opções, apresentada na Seção 6.1.3, permite ao usuário do programa editar um documento XML para armazenar uma estrutura

de menus que será utilizada neste programa. É possível editar um documento XML existente ou criar um novo documento. A seleção desta opção exibe uma tela como a da Figura 6.7.

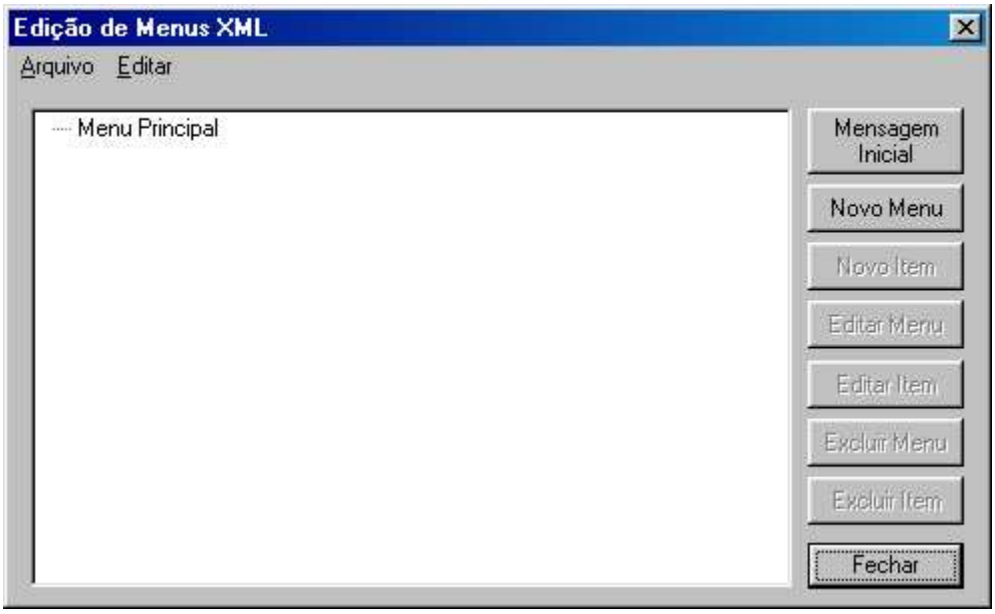

Figura 6.7: Tela de edição de menus XML

Esta tela possui um controle do tipo Tree Control, que mostra a estrutura de menus em árvore - inicialmente, há apenas o nó raiz, chamado de Menu Principal - e botões que permitem a edição do documento. O menu Arquivo desta tela oferece as seguintes opções: "Novo", para criar um novo documento XML; "Abrir", para selecionar um arquivo XML que será editado; "Salvar", que salva o documento XML com o nome atual; "Salvar Como", que salva o documento atual com um novo nome; e "Fechar", que fecha esta tela e retorna à tela principal do programa. A Figura 6.8 mostra a tela de edição de menus com a estrutura de menus da Figura 5.1 carregada no programa.

O menu Editar oferece opções para alterar o documento XML. Todas as opções deste menu também estão disponíveis em botões à direita da tela. A primeira opção é "Mensagem Inicial", que permite selecionar um arquivo Wav com a mensagem inicial do sistema. Ao selecionar esta opção, uma tela é exibida, como mosta a Figura 6.9.

A tela de edição da mensagem inicial possui um botão Selecionar, que permite a seleção de um arquivo Wav do computador.

A segunda opção de edição é "Novo Menu". Esta opção cria um novo menu dentro do item atualmente selecionado. Se este item já possuir um menu, esta opção estará desabilitada. Ao selecionar esta opção, é exibida a tela apresentada na Figura 6.10.

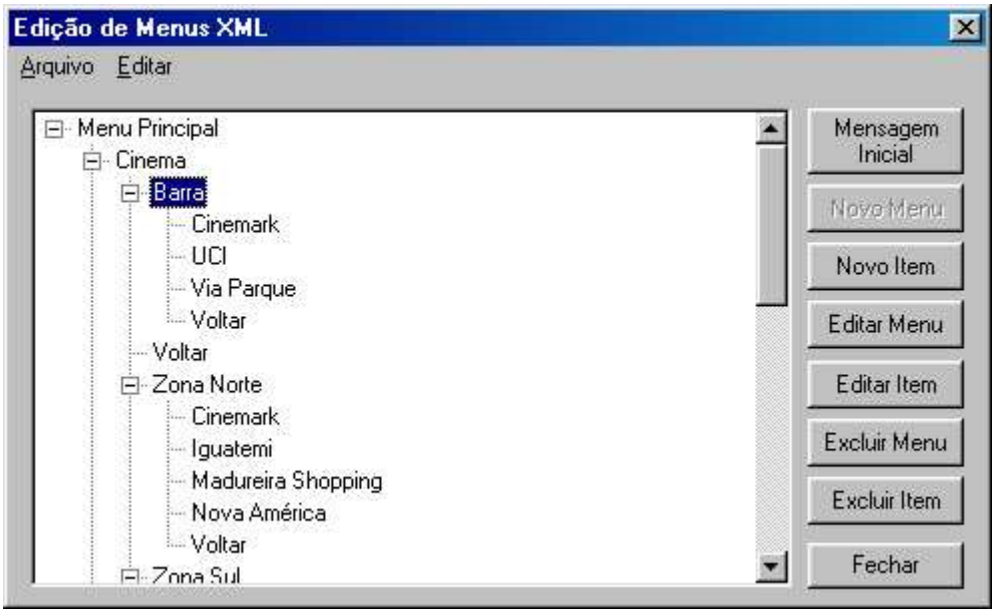

Figura 6.8: Tela de edição de com uma estrutura carregada

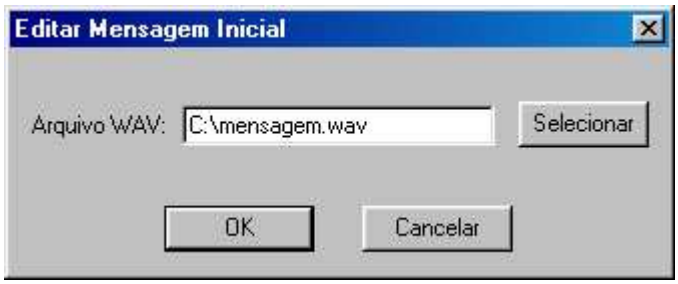

Figura 6.9: Tela de edição da mensagem inicial

A tela de edição de menus é semelhante à tela de edição da mensagem inicial, porém, além do arquivo Wav, é possível editar os arquivos de centróides e modelos correspondentes ao novo menu.

Após definir os arquivos correspondentes ao novo menu e clicar no botão OK, é exibida uma nova tela, para a criação de um novo item no novo menu. Esta tela é apresentada na Figura 6.11.

A tela de edição de itens permite definir o nome do novo item, o arquivo Wav correspondente ao item, caso este seja uma opção final do sistema, e um Check Box, que define se o novo item corresponde à opção voltar ao menu anterior. Se o Check Box for selecionado, as opções de edição de nome e arquivo Wav ficarão desabilitadas, pois um item que retorna ao menu anterior terá o nome “Voltar”, e não possuirá um arquivo Wav associado a ele.

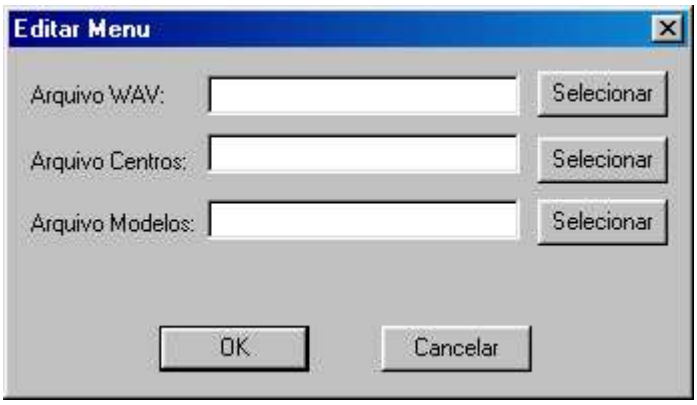

Figura 6.10: Tela de edição de menus

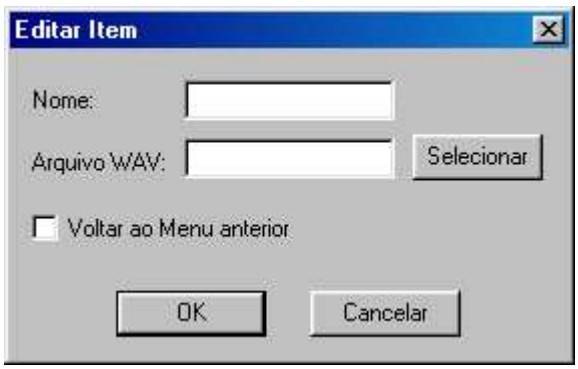

Figura 6.11: Tela de edição de itens

A opção de edição "Novo Item" insere um novo item logo após o item atualmente selecionado na tela. A tela de edição é a mesma mostrada na Figura 6.11.

A opção "Editar Menu" permite alterar os nomes dos arquivos Wav, centróides e modelos do menu contido no item atualmente selecionado. Se este item não possuir um menu, esta opção estará desabilitada. A tela de edição é a mesma mostrada na criação de um novo menu, como apresentado na Figura 6.10.

A opção "Editar Item" é utilizada para modificar o item atualmente selecionado. A tela é a mesma mostrada na Figura 6.11, e permite a edição do nome e do arquivo Wav do item, além da Check Box que define se o item corresponde à opção Voltar.

A opção "Excluir Menu" tem a função de excluir o menu dentro do item selecionado. Ao excluir um menu, todos os itens e menus contidos nele também são excluídos. Esta opção estará desabilitada caso o item selecionado não possua um menu.

A última opção é "Excluir Item", que exclui o item atual. Se este item possuir um

menu, este também será excluído.

6.3 Utilizando o programa

Para utilizar o programa para receber chamadas, é necessário, primeiramente, criar um documento XML com a estrutura de menus que será utilizada. A opção de edição de menus foi descrita com detalhes na Seção anterior. Após a edição de um documento XML, é necessário selecioná-lo, através da opção "Selecionar Arquivo XML", no menu Arquivo da tela principal do programa.

A próxima etapa é abrir um dispositivo de linha, para que o programa possa receber chamadas. Para isto, é utilizada a opção "Abrir linha" do menu Chamada. Ao selecionar esta opção, surge uma tela para seleção do dispositivo de linha que será utilizado, como mostra a Figura 6.12.

Figura 6.12: Tela de seleção do dispositivo de linha

Esta tela mostra um controle Combo Box, com a lista de todos os dispositivos de linha detectados no computador. Ao selecionar um dispositivo na Combo Box, o ícone daquele dispositivo é exibido à esquerda. Se nenhum dispositivo de linha for detectado no computador, a opção OK estará desabilitada nesta tela. O botão Propriedades exibe a tela de propriedades do dispositivo atualmente selecionado na lista. A tela de propriedades do dispositivo é mostrada na Figura 6.13.

A tela de propriedades do dispositivo apresenta configurações como a porta de comunicações que está sendo utilizada, o volume do alto-falante e diversas propriedades avançadas de conexão, como velocidade, bits de dados e configurações de porta.

O botão OK na tela de seleção de dispositivos fecha esta tela. Em seguida, o programa abre o dispositivo selecionado, e passa a exibir seu nome e ícone no título da tela principal.

Figura 6.13: Tela de propriedades do dispositivo

Com o dispositivo aberto, o sistema passa a monitorar a linha, aguardando por chamadas telefônicas - a mensagem "Aguardando chamadas" é exibida na barra de status -, e a opção "Abrir linha" no menu Chamada fica desabilitada até que a linha seja fechada.

Com um dispositivo de linha aberto, o programa é capaz de realizar todas as tarefas para seu funcionamento - atender uma chamada recebida, acessar a estrutura de menus do arquivo que foi selecionado anteriormente e reconhecer a voz gravada pela linha. A chamada poderá ser encerrada depois que o usuário selecionar uma opção final do sistema e escutar a mensagem correspondente ou através da opção "Finalizar chamada" no menu Chamada. Após o encerramento da chamada, o sistema volta a aguardar novas chamadas.

Capítulo 7

Resultados

Para avaliar o sistema desenvolvido, foram realizados vários testes, tendo como objetivo verificar a porcentagem de acerto de cada palavra que faz parte dos menus do sistema.

A estrutura de menus utilizada para os testes foi a mesma apresentada na Figura 5.1. Cada palavra que compõe estes menus foi gravada diversas vezes através da linha telefônica. Foram realizados três testes com os menus.

O primeiro teste é dependente de locutor, ou seja, a voz de uma única pessoa foi utilizada nos treinamentos e testes. Cada palavra que compõe os menus foi gravada 40 vezes, sendo 20 vezes para o treinamento e 20 vezes para os testes. O segundo teste é semelhante ao primeiro, também dependente de locutor, porém utilizando a voz de outra pessoa, e em um ambiente diferente - diferentes locais, computadores, linhas telefônicas e voice modems.

O terceiro teste é independente de locutor. Foram utilizadas vozes de diversas pessoas para o treinamento e vozes de outras pessoas para o teste. Neste teste, foram utilizadas 13 pessoas diferentes, sendo 8 homens e 5 mulheres. Cada palavra foi gravada 5 vezes por cada pessoa. No treinamento, foram utilizadas 9 pessoas - 6 homens e 3 mulheres - e nos testes, 4 pessoas - 2 homens e 2 mulheres.

7.1 Primeiro teste dependente de locutor

Os resultados do menu 1 no primeiro teste dependente de locutor são apresentados na Tabela 7.1. A taxa média de acerto foi de 98 %. Nos menus 2 a 8, apresentados nas Tabelas 7.2 a 7.8, as taxas de acerto foram de, respectivamente, 96,25 %, 95 %, 98 %, 98 %, 78,33 %, 96 % e 95 %. A taxa média de acerto do sistema neste teste foi de 95,15 %.

Tabela 7.1: Resultados para o menu 1 no primeiro teste dependente de locutor

	Cinema	**Economia**	**Esportes**	**Piadas**	**Política**	**Resultados**
Cinema	19	0	0	0	1	95 %
Economia	0	20	0	0	0	100 %
Esportes	0	0	20	0	0	100 %
Piadas	0	0	0	19	1	95 %
Política	0	0	0	0	20	100 %

Tabela 7.2: Resultados para o menu 2 no primeiro teste dependente de locutor

	Barra	**Voltar**	**Zona Norte**	**Zona Sul**	**Resultados**
Barra	18	2	0	0	90 %
Voltar	0	20	0	0	100 %
Zona Norte	0	0	19	1	95 %
Zona Sul	0	0	0	20	100 %

Tabela 7.3: Resultados para o menu 3 no primeiro teste dependente de locutor

	Cinemark	**UCI**	**Via Parque**	**Voltar**	**Resultados**
Cinemark	18	0	1	1	90 %
UCI	0	19	0	1	95 %
Via Parque	0	0	19	1	95 %
Voltar	0	0	0	20	100 %

Tabela 7.4: Resultados para o menu 4 no primeiro teste dependente de locutor

	Cinemark	**Iguatemi**	**Madureira Shopping**	**Nova América**	**Voltar**	**Resultados**
Cinemark	18	0	0	0	2	90 %
Iguatemi	0	20	0	0	0	100 %
Madureira Shopping	0	0	20	0	0	100 %
Nova América	0	0	0	20	0	100 %
Voltar	0	0	0	0	20	100 %

Tabela 7.5: Resultados para o menu 5 no primeiro teste dependente de locutor

	Cinemark	**Estação Botafogo**	**Fashion Mall**	**Rio Sul**	**Voltar**	**Resultados**
Cinemark	18	0	0	0	2	90 %
Estação Botafogo	0	20	0	0	0	100 %
Fashion Mall	0	0	20	0	0	100 %
Rio Sul	0	0	0	20	0	100 %
Voltar	0	0	0	0	20	100 %

Tabela 7.6: Resultados para o menu 6 no primeiro teste dependente de locutor

	Dólar	**Inflação**	**Voltar**	**Resultados**
Dólar	11	0	9	55 %
Inflação	0	16	4	80 %
Voltar	0	0	20	100 %

Tabela 7.7: Resultados para o menu 7 no primeiro teste dependente de locutor

	Basquete	**Futebol**	**Outros**	**Vôlei**	**Voltar**	**Resultados**
Basquete	20	0	0	0	0	100 %
Futebol	0	18	0	0	2	90 %
Outros	0	0	19	0	1	95 %
Vôlei	0	0	0	19	1	95 %
Voltar	0	0	0	0	20	100 %

Tabela 7.8: Resultados para o menu 8 no primeiro teste dependente de locutor

	Brasil	**Internacional**	**Voltar**	**Resultados**
Brasil	17	0	3	85 %
Internacional	0	20	0	100 %
Voltar	0	0	20	100 %

7.2 Segundo teste dependente de locutor

Para o segundo teste dependente de locutor, os resultados obtidos são apresentados nas Tabelas 7.9 a 7.16. As taxas de acerto para os menus de 1 a 8 foram de, respectivamente, 100 %, 97,5 %, 100 %, 99 %, 97 %, 100 %, 100 % e 100 %, e a taxa média foi de 99,12 %.

Tabela 7.9: Resultados para o menu 1 no segundo teste dependente de locutor

	Cinema	**Economia**	**Esportes**	**Piadas**	**Política**	**Resultados**
Cinema	20	0	0	0	0	100 %
Economia	0	20	0	0	0	100 %
Esportes	0	0	20	0	0	100 %
Piadas	0	0	0	20	0	100 %
Política	0	0	0	0	20	100 %

Tabela 7.10: Resultados para o menu 2 no segundo teste dependente de locutor

	Barra	**Voltar**	**Zona Norte**	**Zona Sul**	**Resultados**
Barra	20	0	0	0	100 %
Voltar	0	20	0	0	100 %
Zona Norte	0	0	20	0	100 %
Zona Sul	0	1	1	18	90 %

Tabela 7.11: Resultados para o menu 3 no segundo teste dependente de locutor

	Cinemark	**UCI**	**Via Parque**	**Voltar**	**Resultados**
Cinemark	20	0	0	0	100 %
UCI	0	20	0	0	100 %
Via Parque	0	0	20	0	100 %
Voltar	0	0	0	20	100 %

7.3 Teste independente de locutor

O último teste, independente de locutor, obteve taxas de acerto de 91 %, 85 %, 84 %, 83,75 %, 89 %, 91,67 %, 94 % e 88,33 %, para os menus de 1 a 8. Os resultados são apresentados nas Tabelas 7.17 a 7.24. A taxa média de acerto foi de 91,26 % neste teste.

Tabela 7.12: Resultados para o menu 4 no segundo teste dependente de locutor

	Cinemark	**Iguatemi**	**Madureira Shopping**	**Nova América**	**Voltar**	**Resultados**
Cinemark	20	0	0	0	0	100 %
Iguatemi	0	20	0	0	0	100 %
Madureira Shopping	0	0	20	0	0	100 %
Nova América	0	0	0	19	1	95 %
Voltar	0	0	0	0	20	100 %

Tabela 7.13: Resultados para o menu 5 no segundo teste dependente de locutor

	Cinemark	**Estação Botafogo**	**Fashion Mall**	**Rio Sul**	**Voltar**	**Resultados**
Cinemark	20	0	0	0	0	100 %
Estação Botafogo	0	17	0	3	0	85 %
Fashion Mall	0	0	20	0	0	100 %
Rio Sul	0	0	0	20	0	100 %
Voltar	0	0	0	0	20	100 %

Tabela 7.14: Resultados para o menu 6 no segundo teste dependente de locutor

	Dólar	**Inflação**	**Voltar**	**Resultados**
Dólar	20	0	0	100 %
Inflação	0	20	0	100 %
Voltar	0	0	20	100 %

Tabela 7.15: Resultados para o menu 7 no segundo teste dependente de locutor

	Basquete	**Futebol**	**Outros**	**Vôlei**	**Voltar**	**Resultados**
Basquete	20	0	0	0	0	100 %
Futebol	0	20	0	0	0	100 %
Outros	0	0	20	0	0	100 %
Vôlei	0	0	0	20	0	100 %
Voltar	0	0	0	0	20	100 %

Tabela 7.16: Resultados para o menu 8 no segundo teste dependente de locutor

	Brasil	**Internacional**	**Voltar**	**Resultados**
Brasil	20	0	0	100 %
Internacional	0	20	0	100 %
Voltar	0	0	20	100 %

Tabela 7.17: Resultados para o menu 1 no teste independente de locutor

	Cinema	**Economia**	**Esportes**	**Piadas**	**Política**	**Resultados**
Cinema	19	0	0	0	1	95 %
Economia	0	20	0	0	0	100 %
Esportes	0	0	20	0	0	100 %
Piadas	0	0	0	14	6	70 %
Política	0	0	2	0	18	90 %

Tabela 7.18: Resultados para o menu 2 no teste independente de locutor

	Barra	**Voltar**	**Zona Norte**	**Zona Sul**	**Resultados**
Barra	16	3	0	1	80 %
Voltar	0	18	2	0	90 %
Zona Norte	0	0	20	0	100 %
Zona Sul	0	1	3	16	80 %

Tabela 7.19: Resultados para o menu 3 no teste independente de locutor

	Cinemark	**UCI**	**Via Parque**	**Voltar**	**Resultados**
Cinemark	17	0	3	0	85 %
UCI	3	17	0	0	85 %
Via Parque	0	0	15	5	75 %
Voltar	2	0	0	18	90 %

Tabela 7.20: Resultados para o menu 4 no teste independente de locutor

	Cinemark	**Iguatemi**	**Madureira Shopping**	**Nova América**	**Voltar**	**Resultados**
Cinemark	17	0	3	0	0	85 %
Iguatemi	0	16	4	0	0	80 %
Madureira Shopping	0	0	20	0	0	100 %
Nova América	3	0	0	17	0	85 %
Voltar	1	0	0	0	19	95 %

Tabela 7.21: Resultados para o menu 5 no teste independente de locutor

	Cinemark	**Estação Botafogo**	**Fashion Mall**	**Rio Sul**	**Voltar**	**Resultados**
Cinemark	17	3	0	0	0	85 %
Estação Botafogo	0	20	0	0	0	100 %
Fashion Mall	0	0	20	0	0	100 %
Rio Sul	1	0	4	15	0	75 %
Voltar	1	7	0	0	12	60 %

Tabela 7.22: Resultados para o menu 6 no teste independente de locutor

	Dólar	**Inflação**	**Voltar**	**Resultados**
Dólar	17	1	2	85 %
Inflação	0	20	0	100 %
Voltar	1	1	18	90 %

Tabela 7.23: Resultados para o menu 7 no teste independente de locutor

	Basquete	**Futebol**	**Outros**	**Vôlei**	**Voltar**	**Resultados**
Basquete	19	1	0	0	0	95 %
Futebol	0	20	0	0	0	100 %
Outros	0	0	20	0	0	100 %
Vôlei	0	3	0	17	0	85 %
Voltar	0	2	0	0	18	90 %

Tabela 7.24: Resultados para o menu 8 no teste independente de locutor

	Brasil	**Internacional**	**Voltar**	**Resultados**
Brasil	18	2	0	90 %
Internacional	1	19	0	95 %
Voltar	1	3	16	80 %

Capítulo 8

Conclusões

8.1 Conclusões do projeto

O sistema desenvolvido neste projeto teve como objetivo permitir que um programa rodando em um computador com uma placa de voice modem fosse capaz de monitorar uma linha telefônica e comunicar-se através dela, sem que houvesse a necessidade de um usuário utilizando o programa a partir do momento em que ele estivesse configurado. De um modo geral, este objetivo foi alcançado. O sistema funciona em computadores com sistema operacional Windows, e, a partir do momento em que a linha telefônica está sendo monitorada, funciona sem a intervenção de um usuário.

A escolha da linguagem C++, com orientação a objetos, foi um fator crucial para o bom funcionamento do sistema. O principal motivo é que as bibliotecas utilizadas neste projeto - Voice and Sound, para o reconhecimento de voz, TFX, para acessar o voice modem e a linha telefônica, e Xerces, para utilizar documentos XML - foram desenvolvidas nesta linguagem. Assim, a integração destas bibliotecas com o programa desenvolvido foi bastante facilitada.

A utilização do XML para armazenar a estrutura de menus também se mostrou acertada, pois esta é uma linguagem aberta e bem estruturada, o que permitiu a organização dos dados necessários em um documento que pode ser modificado conforme se queira adicionar ou remover menus ou itens no sistema.

Os resultados obtidos foram bons, apesar de alguns erros no reconhecimento de determinadas palavras. Nos testes dependentes de locutor, percebe-se claramente uma diferença nos resultados - a taxa média de acerto foi de 95,15 % no primeiro teste e de 99,12 % no

segundo teste. Os dois testes foram realizados em ambientes diferentes, com pessoas diferentes em locais diferentes, utilizando computadores, placas de voice modem e linhas telefônicas diferentes. Este fato pode ter provocado as diferenças nos resultados dos dois testes. O fator que mais provavelmente influenciou os resultados foi o ruído, tanto na linha telefônica quanto no ambiente onde o primeiro teste foi realizado. No teste independente de locutor, as taxas de acerto foram menores que nos outros testes, o que já era esperado, visto que as vozes utilizadas no treinamento são diferentes daquelas utilizadas nos testes.

8.2 Propostas para continuidade do projeto

Apesar dos bons resultados obtidos neste projeto, surgem pontos onde o sistema pode ser melhorado, visando solucionar alguns problemas.

Um dos principais pontos que permitiria melhorar o funcionamento do sistema seria torná-lo multiplataforma. Atualmente, o sistema só funciona no sistema operacional Windows, pois a interface gráfica foi desenvolvida em MFC, e a biblioteca TFX utiliza as interfaces TAPI e multimídia do Windows. Como as demais bibliotecas são multiplataforma, bastaria substituir as duas primeiras. Apesar de o MFC possuir substitutos que são multiplataforma, o mesmo não ocorre com o TFX, principalmente devido às dificuldades na configuração de modems no sistema operacional Linux.

Outra melhoria importante seria a capacidade de reproduzir e gravar voz simultaneamente, o que permitiria a um usuário que já conhece as opções do sistema selecionar a opção desejada sem ter que escutar a mensagem de voz que informa as opções disponíveis em um menu.

Quanto ao reconhecimento de voz, um algoritmo robusto poderia aumentar as taxas de acerto do sistema, devido ao problema de ruído na linha telefônica. O aumento da base de vozes utilizada no treinamento também contribuiria para esta melhoria.

Em relação à utilização de XML no programa para armazenar a estrutura de menus, um possível complemento seria utilizar um arquivo de validação XML ao abrir um documento.

Nos testes realizados com o programa, verificou-se que em alguns casos o voice modem não é detectado, ou este não consegue detectar as chamadas recebidas. Em todos os casos, este problema ocorreu com o sistema operacional Windows XP. Uma análise mais detalhada poderia permitir a solução deste problema.

Outra possível melhoria seria permitir o uso de mais de uma linha telefônica simultaneamente, com um voice modem ligado em cada linha. Desta maneira, o programa poderia monitorar diversas linhas e atender as chamadas recebidas simultaneamente. Para isso, seria importante realizar uma medida de consumo de CPU e memória pelo programa durante o gerenciamento de uma chamada, para que fosse possível verificar quantas chamadas um processador seria capaz de receber simultaneamente.

Referências Bibliográficas

[1] GARNIER, G. P., CUNHA, P. S., VILLAS-BOAS, S. B., *et al.*, "Sistema de Reconhecimento Automático de Voz por Telefone". In: *Semana da Eletrônica*, UFRJ, Rio de Janeiro, RJ, Brasil, Setembro 2003.

[2] TANENBAUM, A. S., *Computer Networks*. 3 ed. Prentice Hall, 1996.

[3] "MSDN - Microsoft Developers Network", http://msdn.microsoft.com/.

[4] ALMEIDA, J. C. R. L. D., "Componente de comunicação via modem para reconhecimento automático de voz". In: *Projeto Final de Curso*, Rio de Janeiro, Brasil, Maio 2001.

[5] AMUNDENSEN, M. C., *MAPI, SAPI & TAPI Developers Guide*. Sams Publishing, 1996.

[6] MARGULIES, E., *Secrets of Windows Telephony*. CMP Books, 1997.

[7] "Biblioteca TFX", http://members.bellatlantic.net/ vze3tcmc/csells.htm.

[8] SELLS, C., *Windows Telephony Programming - A Developer's Guide to TAPI*. Addison-Wesley, 1998.

[9] "Biblioteca Xerces-C++", http://xml.apache.org/xerces-c/.

[10] ARCINIEGAS, F., *C++ XML*. Pearson Education, 2002.

[11] PARDI, W. J., *XML in Action*. Microsoft Press, 1999.

[12] STROUSTRUP, B., *A Linguagem de Programação C++*. 3 ed. Bookman, 2002.

[13] VILLAS-BOAS, S. B., *C/C++ e Orientação a Objetos em Ambiente Multiplataforma*. Rio de Janeiro, RJ, Brasil, 2001.

[14] TERUSZKIN, R., JUNIOR, F. G. V. R., VILLAS-BOAS, S. B., *et al.*, "Biblioteca Orientada a Objeto para Reconhecimento de Voz e Aplicação em Controle de Robô". In: *Congresso Brasileiro de Automática*, Rio Grande do Norte, Brasil, Setembro 2002.

[15] EDGAR, B., *PC Telephony*. 4 ed. Flatiron Publishing, 1997.

[16] BLASZCZAK, M., *Professional MFC with Visual C++ 6*. Wrox Press, 1999.