MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
CAMPUS SÃO JOÃO EVANGELISTA

Curso Superior de Sistemas de Informação

Allan Ribeiro dos Santos
Daniela Couto de Oliveira

DESENVOLVIMENTO DE FERRAMENTA DE CONVERSÃO DE TEXTO EM FALA

São João Evangelista
2013

Allan Ribeiro dos Santos
Daniela Couto de Oliveira

DESENVOLVIMENTO DE FERRAMENTA DE CONVERSÃO DE TEXTO EM FALA

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia Minas Gerais – IFMG - Campus São João Evangelista, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Ma. Karina de Carvalho Dutra Lemos

São João Evangelista
2013

FICHA CATALOGRÁFICA
Elaborada pelo Serviço Técnico da Biblioteca do
Instituto Federal Minas Gerais – Campus São João Evangelista

O48d OLIVEIRA, Daniela Couto de, 1992 -

Desenvolvimento de ferramenta de conversão de texto em fala./ Allan Ribeiro dos Santos; Daniela Couto de Oliveira. São João Evangelista, MG: IFMG - Campus São João Evangelista, 2013.
59 p.: il.

Trabalho de Conclusão de Curso - TCC (graduação) apresentado ao Instituto Federal Minas Gerais – Campus São João Evangelista – IFMG, Curso de Bacharelado em Sistemas de Informação, 2013.
Orientador: Prof. Ma. Karina Dutra de Carvalho Lemos

1. Sistema de informação. 2. Software livre . 3. Software. 4. Ensino. I. Instituto Federal Minas Gerais – Campus São João Evangelista. Curso de Bacharelado em Sistemas de Informação. II. Título.

CDD 006

Allan Ribeiro dos Santos
Daniela Couto de Oliveira

DESENVOLVIMENTO DE FERRAMENTA DE CONVERSÃO DE TEXTO EM FALA

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia Minas Gerais – IFMG - Campus São João Evangelista, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Prof. Ma. Karina Dutra de Carvalho Lemos - (Orientadora) IFMG – Campus São João Evangelista

Prof. Esp. Fernando Henrique Mafra - IFMG – Campus São João Evangelista

Prof. Dayler Vinicius Miranda Alves - IFMG – Campus São João Evangelista

São João Evangelista, 14 de novembro de 2013

Dedico esse trabalho aos meus familiares, amigos e professores pelo incentivo e carinho.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus pelo dom da vida e por ter me ungido todos os dias dessa caminhada.

Aos meus pais, que lutaram para que este sonho torna-se realidade.

Agradeço também a todos os professores que me acompanharam durante a graduação, em especial à professora Karina Dutra, pela orientação, pelo aprendizado e apoio em todos os momentos necessários.

Aos meus amigos, pelas orações e pensamentos positivos para que eu pudesse alcançar meus objetivos.

A todos que, de alguma forma, contribuíram para esta construção.

"Não confunda derrotas com fracasso nem vitórias com sucesso. Na vida de um campeão sempre haverá algumas derrotas, assim como na vida de um perdedor sempre haverá vitórias. A diferença é que, enquanto os campeões crescem nas derrotas, os perdedores se acomodam nas vitórias." (ROBERTO SHINYASHIKI, 2013).

RESUMO

Este trabalho aborda o desenvolvimento de uma ferramenta para conversão texto-fala capaz de obter informações textuais e transformar tais informações em áudio. A conversão texto-fala ou TTS (*Text-To-Speech*) são sistemas capazes de converter uma linguagem escrita em fala. Essa possibilidade de armazenar informações na forma escrita, fornecendo saídas através de voz, amplia o uso de computadores para diversas aplicações como acesso a bancos de dados através de telefone, sistemas de correio eletrônico em correio por voz, e principalmente a facilidades providas para pessoas com deficiências vocais e visuais, através de máquinas de auxílio à fala e máquinas de leitura para cegos. Diante do exposto, objetivou-se com o trabalho o desenvolvimento de uma ferramenta que permitiu a conversão de um texto irrestrito em fala, permitindo o armazenamento dos arquivos em áudio para que possam ser acessados posteriormente. Para desenvolver a ferramenta proposta, torna-se cogente estudo de técnicas e procedimentos utilizados em sistemas de conversão texto-fala. O sistema foi desenvolvido utilizando a ferramenta de programação *Microsoft Visual Studio 2010 Professional,* a linguagem de programação C# e a biblioteca *System.Speech* responsável pelo processo de sintetize de texto em fala. O sistema foi apto a processar voz contínua e retorna a resposta imediata pelo programa. Os comandos e suas funções são fornecidos pelo usuário, adequando-se a sua necessidade prática. Diversos estudos podem ser realizados futuramente a respeito do trabalho, como o estudo mais aprofundado a respeito naturalidade da fala e entoação das pronúncias, utilizando para isso um analisador morfo-sintático. Os resultados obtidos ao término do trabalho foram satisfatórios, pois a ferramenta desenvolvida conseguiu obter informações textuais e realizar a síntese em fala conforme proposto.

Palavras-chave: Conversão Texto-Fala. Processamento textual. *System.Speech*. Síntese de texto.

ABSTRACT

This paper discusses the development of a tool to convert text -to-speech capable of obtaining textual information and transform that information into audio. The text-to-speech or TTS (Text- To-Speech) are systems able to convert written language into speech. This possibility of storing information in written form, providing outputs via voice, expands the use of computers for various applications such as access to databases via phone, e-mail systems for voice mail, and in particular the facilities provided for people with vocal and visual disabilities, through machines aid to speech and reading machines for the blind. According to the above, the research objective was to study the development of a tool that allowed the conversion of unrestricted text to speech, allowing storage of audio files so they can be accessed later. To develop the proposed tool, become necessary the study of techniques and procedures used in systems for converting text-to-speech. The system was developed using the programming tool Microsoft Visual Studio 2010 Professional, the C # programming language and the library System.Speech responsible for the process of summarizing text to speech. The system was able to process continuous voice and returns the immediate response by the program. The commands and their functions are provided by the user, adapting its practical necessity. Several studies can be made regarding the future work, as the further study about the naturalness of speech and intonation of the pronunciations, using for this a morpho-syntactic analyzer. The results obtained at the completion of the work were satisfactory, because the tool developed was able to obtain textual information and perform in speech synthesis as proposed.

Keywords: Conversion Text -To-Speech. Text processing. System.Speech. Synthesis text

LISTA DE SIGLAS

GC – *Garbage Collector*

HMM – *Hidden Markov Models*

HTML – *Hyper Text Markup Language*

IDE – *Integrated Development Environment*

JSAPI – *Java Speech Application Programing Interface*

JSGF – *Java Speech Grammar Form*

MBROLA – *Multi Band Resynthesis Over Lap Add*

PHP – *Hypertext Preprocessor*

RF – Requisito Funcional

RNF – Requisito Não Funcional

SO – Sistema Operacional

TTS – *Text-To-speech*

UML – *Unified Modeling Language*

VB – *Visual Basic*

XML – *Extensible Markup Language*

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A utilização da linguagem escrita tem predominado em sistemas computacionais desde o surgimento dos primeiros computadores, por ser a forma eficiente ao transmitir e armazenar informações. (CHBANE, 1994, p.12). Porém, com o avanço da tecnologia e com a crescente tendência de interfaces homem-máquina amigáveis, o uso da linguagem falada em computadores torna-se cada vez mais conveniente, na medida em que a fala é uma forma mais natural de comunicação.

Dados estatísticos de Chaponis, citados por Young e Fallside (1989), mostraram que a comunicação da informação através da voz em situações de interação homem-máquina é em média duas vezes mais eficiente do que qualquer outra forma de comunicação. Ainda segundo eles, a maior eficiência da comunicação verbal, apoiada na crescente evolução das técnicas de processamento de sinais digitais, tem feito com que sistemas de compreensão da fala e síntese de voz difundam-se cada vez mais como meios de entrada e saída de informações em computadores.

Chbane (1994) descreve que "Atualmente os sistemas de compreensão da fala estão restritos a algumas aplicações especiais, enquanto que os sistemas de síntese de voz vêm sendo largamente utilizados". Tais sistemas apresentam particular importância, pois proporcionam a produção de voz a partir de um texto de entrada, unindo a eficiência do armazenamento de dados na forma escrita com a comunicação através da fala.

Esses sistemas possuem diversas abrangências permitindo, por exemplo, acesso a bancos de dados através de telefone, como os sistemas de consulta de saldo bancário, atualmente bastante difundidos, e abrem perspectivas para transformar os atuais sistemas de correio eletrônico em correio por voz. Merece destaque ainda, as facilidades advindas do uso desses sistemas para pessoas com deficiências vocais e visuais, através de máquinas de auxílio à fala e máquinas de leitura para cegos, pois a maioria das interfaces entre computadores e humanos funciona através da linguagem textual, mas este tipo de interação para pessoas cegas torna-se um grande desafio.

Dentre os benefícios que a fala sintetizada apresenta destaca-se a naturalidade e agilidade de sistemas computacionais, a interação com sistemas de telecomunicações e a acessibilidade aos deficientes visuais. Considerando tais benefícios surge o seguinte questionamento: "Como essa interatividade torna-se possível já que a principal forma de interação homem-máquina é através da escrita?". A resposta a esta indagações é o uso dos sistemas de conversão texto-fala ou TTS (*Text-To-Speech*). Luiz Latsch afirma que "Um

conversor TTS é um sistema que a partir de um texto irrestrito, produz fala sintetizada correspondente à leitura". Ainda segundo ele os sistemas TTS realizam duas etapas para a conversão de texto em fala: primeiro, faz uma análise do texto para saber o que ele deve falar e, segundo, faz a produção do som propriamente dito. (LATSCH, 2002, p. 15).

Avanços consideráveis têm sido feitos nesta área de aplicações TTS, visando tornar estes sistemas cada vez mais próximos da fala natural. Leandro Gomes ressalta que "A meta principal destes sistemas é a maximização da inteligibilidade da fala sintetizada". (GOMES, 1998, p. 4). Entretanto, existem algumas dificuldades no desenvolvimento de um sistema de conversão texto-fala que não estão diretamente ligadas às técnicas de conversão. Surgem cada vez mais novas tecnologias e novas linguagens que permitem o desenvolvimento de *softwares* das mais variadas formas, porém esse fato pode dificultar a obtenção de informações de quaisquer sistemas computacionais, para então converter estas informações em fala.

Diversos fatores motivaram o desenvolvimento desse trabalho, pois a comunicação máquina-homem ou homem-homem através da voz já vem sendo utilizada em várias áreas, mas não amplamente. A aplicação do TTS na comunicação entre a máquina e o homem, pode auxiliar deficientes visuais no acesso à informação. Na comunicação homem-homem, deficientes vocais e/ou auditivos conseguem se comunicar através do som através de sistemas TTS, e ainda, utilizando tradutores automáticos, seria possível se comunicar em outras línguas. Além disso, o desenvolvimento de um sistema TTS implica na participação de várias áreas de conhecimento, como a linguística, inteligência artificial, processamento de sinais, análise de sistemas, entre outros, o que faz com que o assunto seja classificado como multidisciplinar. Apesar de existir várias aplicações possíveis e já utilizáveis para o sistema TTS, ainda é uma tecnologia em desenvolvimento que ainda requer avanços para ser aceita e absorvida pelo mercado.

Em virtude do exposto, o objetivo do trabalho foi criar uma ferramenta de conversão texto-fala, permitindo o armazenamento dos arquivos em áudio para que os mesmos possam ser acessados posteriormente. Para isto o sistema realiza processamento dos textos, digitados ou importados de dentro do sistema, em áudio em tempo de execução com a ajuda de um sintetizador de fala desenvolvido por terceiros, fornecendo a possibilidade do usuário salvar os textos, convertidos em forma de áudio para acessos posteriores. Os processamentos dos textos estão disponíveis nos idiomas inglês, espanhol e português.

O trabalho está estruturado em capítulos, sendo este a introdução que apresenta o tema da pesquisa descrevendo sua problemática, a relevância do desenvolvimento do *software*, a abrangência do estudo realizado e os objetivos. O segundo capítulo envolve a fundamentação

teórica, onde apresenta-se conceitos e técnicas de sistemas TTS, instrumentos e procedimentos necessários para realização do *software* e trabalhos correlatos. No capítulo três encontra-se a metodologia, apresentando a especificação, a implementação e o funcionamento do sistema de conversão texto-fala proposto. Esse capítulo trata isoladamente cada etapa implementada, regras e estratégias desenvolvidas para o tratamento do texto, algoritmos e técnicas utilizadas. No capítulo quatro são apresentados os resultados obtidos ao término do trabalho, assim como uma discussão a respeito dos mesmos. O quinto capítulo trata das conclusões provenientes do desenvolvimento do sistema bem como as possíveis extensões do mesmo.

2 FUNDAMENTAÇÃO TEÓRICA

De acordo como Marconi e Lakatos (2010), a fundamentação teórica são os principais conceitos teóricos necessários ao desenvolvimento de um trabalho. É o suporte teórico para os estudos, análise e reflexões, sobre os dados e/ou informações coletadas, tendo forte base teórica e pode ser decisiva na pesquisa jurídico teórica.

Em virtude do exposto, este capítulo trata dos assuntos que serviram como referencial para o trabalho, cujo objetivo foi revisar os conceitos e técnicas que foram utilizados na elaboração da aplicação TTS. Abrange na primeira seção deste capítulo uma breve explicação sobre os sistemas de conversão texto-fala e a segunda seção aborda como deve ser o processamento textual. Apresenta-se na terceira seção exemplificações de aplicações TTS, e na quarta seção discorre de breve apresentação da gramática da língua inglesa, portuguesa e espanhola. Na quinta seção aborda-se o processo de modelagem de sistemas utilizando UML (*Unified Modeling Language* – Linguagem de Modelagem Unificada), e na sexta seção são apresentadas as tecnologias adotadas para desenvolvimento da ferramenta proposta. Na sétima seção é abordado o processo de documento de elicitação. Por fim, na oitava seção são correlatados trabalhos realizados no Brasil envolvendo procedimentos TTS.

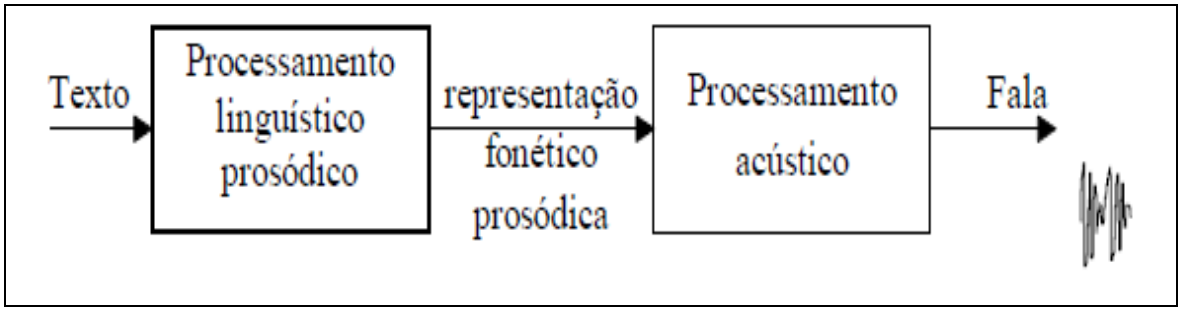
2.1 Sistemas de conversão texto-fala

De acordo com Chbane (1994), a síntese de fala é feita, por alguns sistemas, através da concatenação de unidades como frases ou palavras pré-gravadas, armazenadas em banco de dados, e estes são restritos ao vocabulário ou requerem enorme banco de dados em memória. Portanto, um sistema TTS possui duas características principais: relacionar a língua escrita e a língua falada e sintetizar os sons da língua falada.

Na síntese TTS, um computador recebe como entrada um arquivo de texto ou texto digitado e produz, por meio de alto-falantes, a leitura em voz alta do mesmo texto. Devido à inerente complexidade de sistema TTS, é indispensável dissecar o problema de síntese de voz a partir do texto em problemas menores que podem ser resolvidos satisfatoriamente.

De acordo com Silva (2004), tais sistemas têm uma organização modular que facilita construção dos módulos que fazem parte do bloco de processamento linguístico e prosódico e são extremamente dependentes da linguagem alvo. O modelo genérico da arquitetura desse tipo de sistema abrange alguns procedimentos que envolvem as etapas para o processamento textual. Esse modelo é apresentado na Figura 1.

Figura 1- Digrama de blocos genérico de um sistema



Fonte: TEIXEIRA, 1998

Diversos autores possuem denominações diferentes para os dois blocos básicos apresentados Figura 1, mas a funcionalidade descrita para cada um deles é praticamente a mesma nas diversas referências. Tiago Oechsler, afirma que "Os requisitos de um sintetizador são: a capacidade de produzir sinais com as características espectrais do sinal de fala e ainda ser capaz de alterar a estrutura temporal e a frequência fundamental sem produzir distorções apreciáveis". (OECHSLER, 2009, p. 13). Ainda segundo ele, um sistema TTS para ter boa aceitabilidade precisa ser capaz de pronunciar corretamente um texto e produzir fala inteligível e natural.

Segundo Silva (2004), outros fatores como velocidade de resposta, controlabilidade, limite em memória e capacidade de ser configurado e modificado, também contribuem para a aceitabilidade de um sistema. Tais fatores são determinados diretamente pela estrutura de dados e estrutura de software, no qual requer significativa importância.

2.2 Processamento textual

O processo de síntese de voz a partir de texto inicia-se com o processamento linguístico para a determinação da estrutura fonológica das sentenças dos textos de entrada. Essa estrutura, constituída pelos fonemas que formam as palavras, é analisada para a determinação dos parâmetros acústicos que serão utilizados posteriormente no controle do sintetizador de voz, o qual produzirá a fala. (CROCHIERE; FLANAGAN, 1990).

O trabalho de Crochiere e Flanagan (1990) elucida algumas etapas a serem seguidas para se conseguir realizar o processo completo de conversão de texto em fala. O primeiro passo é um pré-processamento do texto, que visa eliminar caracteres e palavras desconhecidas que a ferramenta não está preparada para interpretar. Nesta fase, o texto é separado palavra por palavra para que possa ser dada continuidade no processamento.

Nesta etapa que são substituídas as abreviações, as siglas e os símbolos especiais em sua forma extensa. Os algarismos também podem receber tratamento, sendo convertidos em formato por extenso, mas com pronúncia literal. A segunda etapa do processamento é realizar uma análise linguística sobre as palavras já separadas.

Ainda segundo Crochiere e Flanagan (1990), esta análise visa encontrar a classe gramatical correta de cada palavra existente no texto. O terceiro passo é fazer a identificação dos fonemas que formam cada palavra. Ou seja, cada palavra deverá ser transformada de uma sequência de letras em uma sequência de fonemas. Também deve ser identificada a sílaba tônica de cada palavra. O quarto passo do processamento textual para a conversão texto-fala é o processamento prosódico. O último passo do processamento textual consiste na síntese do texto em áudio. É o passo responsável por utilizar o resultado das quatro etapas anteriores e produzir um som que represente o texto de entrada.

"A prosódia trata do estudo da entonação, da acentuação, do ritmo de fala presente em palavras e orações". (CROCHIERE; FLANAGAN, 1990, p.25). Este processamento especial que tem a missão de tentar tornar o texto mais próximo do natural, colocando pausas onde são necessárias, ou acelerando certas partes da pronúncia, assim como adicionando tons de exclamação ou interrogação.

2.3 Aplicações de sistemas TTS

Inúmeras aplicações de sistemas TTS já foram desenvolvidas. Abaixo estão listados alguns exemplos da aplicabilidade dessa tecnologia.

a) Serviços de telecomunicações: Sistemas TTS tornam possível acessar informação textual por meio do telefone. Sabendo que cerca de 70% das chamadas de telefone atualmente requerem muito pouca interatividade, um prospecto é digno de ser considerado (DUTOIT, 1997). Textos podem abranger desde pequenas mensagens, como eventos culturais locais (cinemas, teatros) até enormes bases de dados que dificilmente poderiam ser lidas e armazenadas como fala digitalizada.

b) Ensino de linguagens: Sintetizadores TTS de alta qualidade podem com ajuda de um sistema de aprendizagem promover ferramentas de ensino a novas linguagens. De acordo com Dutoit (1997), esse tipo de projeto tem amplo mercado, no entanto, ainda não é plenamente implementado devido à baixa de qualidade disponível em sistemas comercias de síntese de voz.

c) Ajuda a pessoas deficientes: Incapacidades de voz originam-se em problemas nas sensações mentais ou motores. Máquinas podem invariavelmente produzir, em último caso, com a ajuda de um teclado especialmente projetado e um programa de rápida montagem de sentenças, voz sintética em poucos segundos para remediar esses impedimentos. (CAIRO, 1995).

d) Livros e brinquedos falantes: O mercado de brinquedos tem se aproximado cada vez mais aos recursos de síntese de voz. Muitos brinquedos falantes têm sido criados, porém limitações de qualidade invariavelmente interferem na ambição educacional dos produtos. De acordo com Cairo (1995) "Os sintetizadores de alta qualidade melhoram essas situação, mas são bastante caros para se agregar ao valor de brinquedos".

e) Monitoramento vocal: Em alguns casos, informação oral é mais eficiente do que mensagens escritas. O apelo é mais forte enquanto a atenção pode ainda focar em outra fonte visual de informação. "Consequentemente a ideia de incorporar sintetizadores de voz no gerenciamento ou controle de sistemas, como em cabines de avião, para prevenir os pilotos a serem alertados não somente com informação visual". (CAIRO, 1995).

f) Sistemas de voz aplicados a navegação por GPS (Global Positioning System): instalado em viaturas, para que as orientações visuais sejam acompanhadas por informações auditivas. (BRAGA; DIAS, 2009).

A maioria dos sistemas texto-fala comerciais da atualidade têm obtido bons resultados com a concatenação de unidades temporais, utilizando banco de dados com unidades diferenciada sem tamanho como trifones e difones e diferenciadas em frequência fundamental.

Os maiores esforços tem se dado na determinação da entoação prosódica através de características sintáticas do texto. Existem sintetizadores que utilizam dicionários léxicos, gramáticas e conjuntos de regras linguísticas para determinação de duração, frequência fundamental e intensidade.

Esta abordagem tem utilizado produtos da Linguística Computacional e da Inteligência Artificial. Em consequência do desenvolvimento da tecnologia no processamento da fala, os sistemas operacionais têm procurado incluir sistemas de voz, síntese e reconhecimento, como interface. Desenvolvedores para plataformas Unix vêm aprimorando os sistemas Festival e *Mbrola(Multi Band Resynthesis Over Lap Add)*, que oferecem suporte para a geração de novos banco de dados para diferentes vozes ou línguas.

2.4 Gramáticas das línguas portuguesa, inglesa e espanhola

Segundo Dimas Chbane "A linguagem natural pode ser expressa em variados idiomas por todo o mundo. Cada idioma é uma forma de linguagem natural irrestrita". Entretanto, cada idioma possui um conjunto de normas e regras que determina sua estrutura e sua melhor utilização para a comunicação.

A gramática nada mais é do que este conjunto de regras e normas que rege um idioma. Selmini citado por Infante e Cipro (1998, p.16) afirma ainda que "a gramática normativa estabelece a norma culta, ou seja, o padrão linguístico que socialmente é considerado modelar e é adotado para o ensino nas escolas e para a redação dos documentos oficiais.".

O estudo das gramáticas, tanto da língua portuguesa como da língua inglesa e espanhola, envolve o estudo da fonologia, da morfologia, da sintaxe e da semântica, e é essencial para a construção de um sistema de conversão texto-fala que deva atender aos três idiomas. "A morfologia analisa a estrutura, a formação e os mecanismos de flexão das palavras, além de dividi-las em classes gramaticais." (INFANTE; CIPRO, 1998, p. 17). Já a sintaxe, "[...] é a parte da gramática que estuda a disposição das palavras na frase e das frases no discurso." (GRUPO VIRTUOUS, 2012, p. 22).

2.5 Modelagem de sistemas utilizando UML

A UML é uma linguagem de modelagem que serve para a realização de uma padronização no desenvolvimento *de software*. Para Booch, Rumbaugh e Jacobson (2000), a UML é uma linguagem muito significativa, compreendendo todas as visões imprescindíveis ao desenvolvimento e implantação de sistemas de informação corporativos distribuídos a aplicações fundamentadas em *Web* e até mesmo sistemas complexos de tempo real. "A UML não é uma linguagem de programação e sim uma linguagem de modelagem, cujo objetivo é auxiliar os engenheiros de software a definir as características do software". (PRESSMAN, 2005, p.74).

Para que a realização do desenvolvimento de um projeto de software seja bem sucedida, a UML estabelece alguns diagramas e padrões de desenvolvimento que tem por objetivo a especificação, a visualização, a construção e a documentação dos artefatos de um sistema de software.

Uma linguagem de programação fornece um vocabulário e regras para que se desenvolvam aplicações utilizando a mesma. Já uma linguagem de modelagem tem sua visão

voltada para o aspecto conceitual e físico de um sistema. Portando, uma linguagem como a UML, é uma linguagem-padrão para a preparação da composição de projetos de *software* (PRESSMAN, 2005, p.76).

Por meio de diagramas, a UML possibilita a representação sistemas de *software* de diferentes pontos de vista. Facilitando assim a comunicação entre todas as pessoas envolvidas no desenvolvimento de um sistema como gerente, analista de sistema, coordenadores, programadores, entre outros, apresentando um vocabulário de entendimento facilitado. (PRESSMAN, 2005, p.76). Segundo Booch, Rumbaugh e Jacobson (2000) os diagramas da UML estão divididos em estruturais e comportamentais. Os diagramas estruturais são aspectos estáticos, ou seja, representação da estrutura de um sistema e são classificados em:

a) diagramas de classe;

b) diagrama de objetos;

c) diagrama de pacotes;

d) diagrama de estrutura composta;

e) diagrama de componentes;

f) diagrama de implementação.

Os diagramas comportamentais são aqueles onde o comportamento das classes é representado e seus principais diagramas são:

a) diagrama de caso de uso;

b) diagrama de sequência;

c) diagrama de colaboração;

d) diagrama de estados;

e) diagrama de atividade.

2.6 Tecnologias adotadas

Esta seção contempla as ferramentas de programação necessárias para realização para da ferramenta de conversão texto em fala. Tais tecnologias apresentam os requisitos necessários para desenvolvimentos de sistemas TTS.

2.6.1 Linguagem de programação C#

Segundo Maya (2000), a sintaxe da linguagem C# (*Sharp*) é orientada a objetos sendo baseada na linguagem de programação C++ e inclui muitas influências de outras linguagens

de programação como *Object Pascal* e *Java*. "A linguagem permite um novo grau de intercâmbio entre linguagens (componentes de software de diferentes linguagens podem interagir), com isso os desenvolvedores podem empacotar até software antigo, para trabalhar com novos programas C#". (MAYA, 2000, p. 4).

A sintaxe utilizada pelo C# é relativamente simplificada, o que diminui o tempo de aprendizado. A linguagem é fortemente tipada, o que evita erros por manipulação imprópria de tipos e atribuições incorretas. Segundo Maya (2000), o C# apresenta características essências sendo elas:

a) Simplicidade: De acordo com os projetistas de C# a linguagem é prestigiosa quanto o C++ e tão simples quanto o *Visual Basic*;

b) Gera código gerenciado: Assim como o ambiente *Microsoft*. *Net Framework* é gerenciado, assim também é o C#;

c) Controle de versões: Cada *assembly* gerado tem informação sobre a versão do código, permitindo a coexistência de dois *assemblies* homônimos, mas de versões diferentes no mesmo ambiente;

d) Flexibilidade: se o desenvolvedor precisar usar ponteiros, o C# permite, mas ao custo de desenvolver código não gerenciado, chamado "*unsafe*";

e) Linguagem gerenciada: os programas desenvolvidos em C# executam num ambiente gerenciado, o que significa que todo o gerenciamento de memória é feito pelo runtime via o GC[1] *(Garbage Collector* – Coletor de lixo).

2.6.2 A ferramenta ManagedSpy

A *ManagedSpy* é uma ferramenta utilizada para realizar testes diversos em interfaces de aplicações *Windows Forms* desenvolvidas com o *Microsoft .Net Framework*. "Esta ferramenta identifica aplicações compatíveis que estejam executando simultaneamente e permite obter propriedades dos componentes de interface destas aplicações". (WULFE, 2006). Ela é capaz de identificar, por exemplo, que uma determinada aplicação possui um componente do tipo *Panel*, que por sua vez possui um componente do tipo *TextBox* e um componente do tipo *Button*. Dos componentes descobertos, a ferramenta obtém diversas informações, como tamanho, posição na tela ou legenda.

[1]Coletor de lixo (em inglês: garbage collector, ou o acrônimo GC) é um processo usado para a automação do gerenciamento de memória. Com ele é possível recuperar uma área de memória inutilizada por um programa, o que pode evitar problemas de vazamento de memória, resultando no esgotamento da memória livre para alocação (MAEBE; RONSSE; BOSSCHERE; 2004).

Essa ferramenta possui uma biblioteca nomeada como *ManagedSpyLi*, que consiste é uma biblioteca escrita na linguagem C#, que permite acessar os componentes *Windows Forms*. Conceitualmente, esta classe funciona realmente como um proxy[2], na forma como ela manipula as trocas de mensagens entre um componente de uma aplicação e uma segunda aplicação que busca informações deste componente.

De acordo com Wulfe (2006) com essa biblioteca, torna-se possível buscar e setar propriedades destes componentes, além de interceptar os eventos disparados. Ela funciona transferindo informações entre os dois processos (o processo que tem seus componentes interpretados e o processo que utiliza estes componentes) através de um arquivo de memória mapeada[3].

2.6.3 A biblioteca System.Speech

O pacote principal para utilização da biblioteca *System.Speech*e é o pacote *Speech SDK* 5.1. Segundo Uzai (2010) esse pacote oferece recurso para transição de texto para fala e de reconhecimento de voz e é estruturado na criação de uma gramática, no reconhecimento da gramática, na manipulação da entrada de áudio, na manipulação da saída de áudio, na conversão de texto em áudio e na fragmentação da entrada de áudio.

A biblioteca *System.Speech* é uma biblioteca nativa da *Microsoft .Net Framework* que é capaz de fazer tanto síntese de voz, como reconhecimento de voz. Uzai (2010) afirma que a biblioteca *System.Speech* "É um conjunto de componentes que permitem a uma aplicação utilizar a voz como interface no sistema Windows, tanto por reconhecimento quanto por síntese da voz".

De acordo com Macoratti (2010) a biblioteca *System.Speech* possui propriedades e métodos que permitem sintetizar em voz humana uma *string*, usando a linguagem padrão e a voz do sistema operacional. Os componentes são utilizáveis por diferentes linguagens, como por exemplo, C++, Java ou *Visual Basic*, e em diferentes níveis de simplicidade ou complexidade.

---

[2]Um proxy é um serviço de rede que costuma rodar em um servidor e que serve basicamente para controlar a navegação de usuários de uma rede. Ele controla a comunicação entre os usuários de uma rede e os recursos que estes desejam acessar. Esses servidores têm uma série de usos, como filtrar conteúdo, providenciar anonimato, entre outros. (NASCIMENTO, 2009).

[3]Um arquivo de memória mapeada é um arquivo cujo conteúdo fica na memória virtual. Este mapeamento entre um arquivo e um espaço de memória permite que múltiplos processos leiam e modifiquem informações diretamente neste arquivo em memória. Uma vez presente esta correlação entre o arquivo e o espaço de memória permite ao programa manipular a porção mapeada como se fosse seu espaço primário de memória. (MICROSOFT CORPORATION, 2011).

É possível também usar como entrada um arquivo no formato *Speech Synthesis Markup Language* (SSML), que é uma adaptação da *eXtensible Markup Language* (XML) para conter informação necessária para a síntese de voz. O formato SSML permite descrever as palavras em formas de fonemas, assim como adicionar informação prosódica e outros detalhes. De acordo com Macoratti (2010) os principais namespaces[4] da biblioteca são:

a) *System.Speech.Synthesis*: *Namespace* responsável por conter classes para manipular o sintetizador de voz, capaz de transformar texto em áudio. Esse namespace contém classes para inicializar e configurar um mecanismo de síntese de fala para criar prompts para gerar a fala, para responder a eventos e modificação de características de voz.
b) *System.Speech.AudioFormat*: *Namespace* responsável por conter classes para gerar o arquivo de áudio e definir a qualidade do mesmo. Ele consiste em uma única classe, que contém informações sobre o formato de áudio que está sendo inserida para o mecanismo de reconhecimento de fala ou sendo saída do mecanismo de síntese de fala.
c) *System.Speech.Recognition*: Contém classes *e namespaces* responsáveis por fazer o efeito inverso do proposto, ou seja, reconhecer a voz e transformá-la em texto. Os aplicativos usam o *System.Speech.Recognition* para acessar e estender essa tecnologia de reconhecimento de fala básico e contém tipos de tecnologia de fala do *Windows Desktop* para o reconhecimento de fala.

2.6.4 Pacote de vozes necessárias

O pacote de *Voz Microsoft Zira Desktop* se encontra na maioria dos Sistemas Operacionais (SO) Windows. Esse pacote de voz é utilizando em grande escala em sistemas inteligentes de telefonia e sistemas avançados TTS. Umas das principais características dessa voz é a inelegibilidade na transcrição de texto e agilidade no processamento.

Como o padrão de voz do Windows é no idioma inglês torna-se necessário acoplar ao sistema o reconhecimento de palavras da língua portuguesa. Para isso existe o pacote de voz Raquel em português (*Raquel Voice Brazilian Portuguese*). Esse pacote de voz foi desenvolvido especialmente para o idioma português com pronúncia brasileira e já vem sendo utilizado em diversos leitores de tela no Brasil.

---

[4]O namespace é o equivalente a um “pacote” no C#

O pacote de voz *Microsoft Helena Desktop*, consiste em um padrão de voz do SO *Windows* que utiliza o idioma em espanhol. Essa voz permite a transcrição de textos inteligíveis e em tempo real. Este pacote de voz é utilizado em alguns sistemas TTS como o sistema *Microsoft Agent 2.0,* que consiste em uma tecnologia que o computador ler o que está escrevendo.

2.6.5 Microsoft Visual Studio 2010

Para realização da ferramenta torna-se necessário o uso do ambiente de desenvolvimento integrado (IDE) *Microsoft Visual Studio* 2010 *Professional.* " Esse IDE é uma coleção de ferramentas de desenvolvimento expostas por meio de uma interface de usuário comum". (MAYA, 2009, p.11).

De acordo com Maya (2000, p. 6) "A *Microsoft Visual Studio* 2010 *Professional* é um pacote de programas da empresa da *Microsoft* para desenvolvimento de software especialmente dedicado ao *Microsoft*. *Net Framework* e as linguagem Visual Basic (VB), C, C++, C# (C Sharp) e J# (J Sharp)".

Maya ainda afirma que, "*A Microsoft Visual Studio* fornece as ferramentas para projetar, desenvolver, depurar e implantar aplicativos Web, Web Services XML e aplicações cliente tradicionais." (MAYA, 2000, p. 7).

2.7 Documento de elicitação

Através desse documento são estabelecidos os requisitos de *software*. *Es*se processo de especificação do problema não é uma atividade simples, principalmente se considerarmos que a declaração de um problema tipicamente contém ambiguidades, contradições, falta de informações e/ou informações irrelevantes.

Segundo Costa, Guerrero e Mendonça "Cabe à elicitação a tarefa de identificar os fatos que compõem os requisitos do Sistema, de forma a prover o mais correto e mais completo entendimento do que é demandado daquele software". (COSTA; GUERRERO; MENDONÇA, 2008, p.10).

A elicitação de requisitos envolve a compreensão do domínio da aplicação, o problema específico a serem resolvidas, as necessidades e limitações organizacionais e as facilidades especificas, necessárias para as partes interessadas.

2.8 Trabalhos correlatados

Todos os trabalhos correlatos apresentados são trabalhos acadêmicos desenvolvidos no Brasil e envolvem de algum modo às técnicas utilizadas em sistemas de conversão texto-fala na língua portuguesa.

2.8.1 Sphinx4

Segundo Sourceforge (2008), o *Sphinx4* é uma ferramenta escrita em Java que suporta JSAPI (*Java Speech Application Programing Interface*), neste caso a Java *Speech Grammar Form* – JSGF. Sua função é a de reconhecimento de comandos por voz. Isso se deve a questões de desempenho, já que seu objetivo não é o de realizar um reconhecimento de voz, como por exemplo, aplicações que identificam a voz de uma pessoa utilizando inteligência artificial.

Ainda de acordo com Sourceforge (2008), o reconhecedor de fala do *Sphinx4* é baseado no modelo estatístico *Hidden Markov Models* (HMM). A característica de modelos baseados em HMM é que os fonemas são representados por modelos estatísticos que representam os dados deste fonema, que nada mais são que informações que diferenciam um fonema de outro.

2.8.2 FurbTTS

O *FurbTTS* é o protótipo de um sistema conversor texto-fala desenvolvido por Oechsler (2009). O protótipo processa textos escritos em português, fazendo um pré-processamento do texto onde números, siglas e abreviações são transformados em texto puro, para que possam ser sintetizados. É feita também uma análise linguística para atribuir a classe gramatical correta para cada palavra.

Após esta análise é feita a identificação dos fonemas que formam cada palavra, utilizando um dicionário de exceções, se necessário. Este dicionário de exceções pode ser alimentado pelos usuários durante a utilização do protótipo, fazendo com que o mesmo aprenda a pronunciar corretamente novas palavras. É feita também a identificação das sílabas tônicas de cada palavra, antes de passar o texto por um processamento prosódico, para tratar a entonação e a duração de cada segmento da fala.

De acordo com Oechsler (2009, p. 38), "Um dos objetivos desse trabalho é gerar um arquivo de saída com a lista de fonemas que devem ser sintetizados. O formato de saída foi especificado obedecendo ao formato de entrada do sintetizador MBROLA". Nesta lista de fonemas consta o tempo de duração de cada fonema e informações prosódicas.

2.8.3 Sistemas para conversão de textos em fonemas no idioma português

Desenvolvido por Chbane (1994), este sistema faz a transcrição de um texto em português para uma representação fonológica correspondente. Este sistema não faz a conversão de texto em fala propriamente dita, mas faz um estudo aprofundado de como deve ocorrer à transcrição das palavras para seus respectivos fonemas.

Nesse trabalho são abordados alguns aspectos no que se refere aos fonemas, sua classificação quanto a parâmetros de base acústica e articulatória. O texto inicial é processado, removendo partes indevidas e substituindo certas partes como datas, números, siglas e abreviações, para que esses possam ser sintetizados de maneira coerente.

Dessa forma é feita uma análise das palavras para determinar a classe gramatical de cada palavra, através de um dicionário de palavras e uma série de regras que tratam as exceções que não estão presentes no dicionário de palavras. (CHBANE, 1994, p.10). Após estas etapas, é feita a identificação dos fonemas que formam as palavras, utilizando também um dicionário e uma série de regras para a conversão de letras em fonemas. É feito também um processamento prosódico para encontrar o ponto correto de entonação de cada palavra.

Embora o sintetizador de voz em si não tenha sido implementado, os conceitos envolvendo a síntese da voz foram estudados para que o resultado final fosse o mais próximo possível de um texto pronto para ser sintetizado. "Os dicionários utilizados durante o processo se valem de estruturas de dados criadas pelo próprio autor, voltadas para a otimização da performance do processo". (CHBANE, 1994, p. 10).

2.8.4 Sistema de conversão texto-fala com modelagem de prosódia

Esse projeto foi desenvolvido com intuito da concepção de um protótipo para conversão de textos em linguagem natural para uma linguagem com gramática formal. Para alcançar tal objetivo o texto a ser convertido carece ser processado definindo regras referentes às normas gramaticais e gerar o arquivo com o código em língua formal.

Com tudo Vagnes Luis (2011) propôs complementar o projeto propiciando a complementação futura do *software* com o módulo de fala. Segundo Vagnes Luis (2011), as etapas do processo de conversão texto-fala são ordinalmente cinco: pré-processamento, análise linguística, transcrição fonética, processamento prosódico e síntese. Entretanto foca nas quatro primeiras etapas, inclusive não aplica a última etapa em seu projeto que seria o processamento acústico.

2.8.5 Sistema para conversão texto-fala para a língua portuguesa

Gomes (1998) desenvolveu um sistema de conversão texto-fala, divido em três módulos: um módulo de processamento textual, um módulo de processamento prosódico e um módulo sintetizador de voz.

O processamento do texto faz o pré-processamento do texto, a classificação gramatical das palavras e a divisão das palavras em sílabas para posteriormente transcrever as palavras para sua forma fonética.

O processamento prosódico do texto inclui "módulos para determinação de fronteiras prosódicas, geração de padrões de entonação e obtenção de durações de segmentos". (GOMES, 1998, p. 1).

A síntese da voz é feita utilizando o sintetizador de formantes de Klatt (KLATT, 1980) e a abordagem de síntese por regras. É utilizada como entrada para a síntese de voz a própria transcrição fonética obtida pela etapa do processamento textual, já com a informação prosódica necessária.

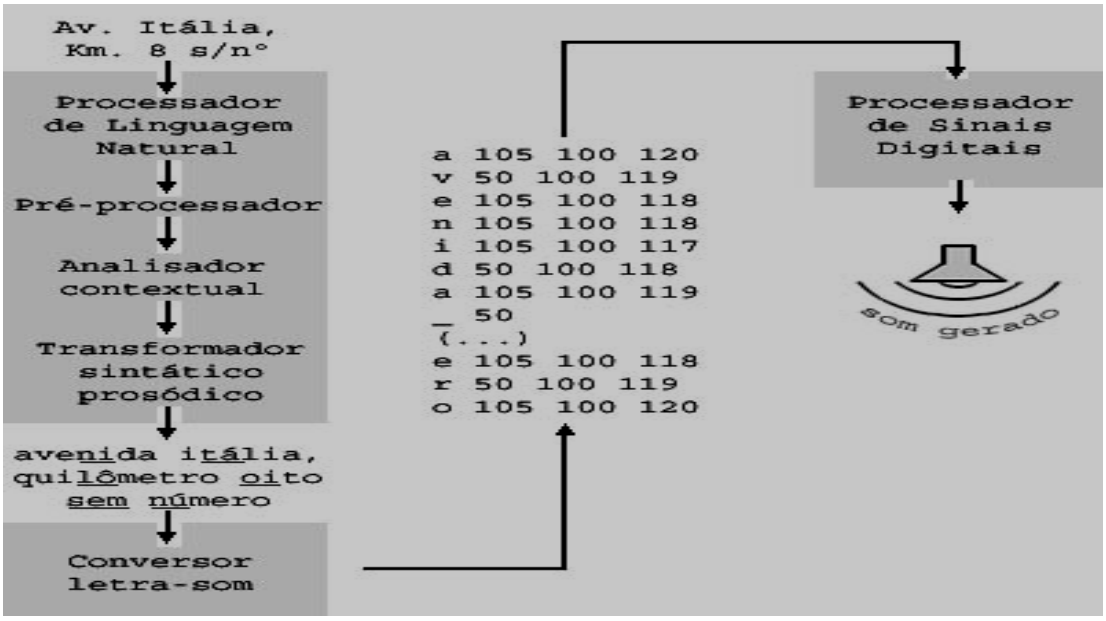
2.8.6 Nambiquara

Trata-se de um sistema de Texto-Fala livre, que se utiliza da abordagem da síntese concatenativa, pois busca alcançar bons resultados com custo computacional razoável (Texeira, 1998). Para isso a equipe desenvolveu uma camada de aplicação que se utiliza o sistema concatenativo MBROLA.

Trata-se de um projeto acadêmico, que visa o aprimoramento do projeto através da filosofia do software livre, o qual apresenta código fonte aberto. O sistema foi implementado utilizando linguagem de programação Php (*Hypertext Preprocessor*), funcionando sobre servidor apache, auxiliados por formulários html (*Hyper Text Markup Language* - Linguagem de Marcação de Hipertexto) e *scripts JavaScript*. Como alguns módulos necessitam consultar

tabelas de exceções, como no caso da substituição de siglas e abreviaturas, (por exemplo, em av. deve ser substituído pela palavra avenida), esse banco foi desenvolvido no MySQL. A estrutura do projeto pode ser vista na Figura 2, onde é mostrada a estrutura implementada.

Figura 2 – Sistema Nambiquara



Fonte: TEIXEIRA, 1998

Foi destacada no projeto a dificuldade em se modelar características emocionais adaptadas ao contexto, o que parece ser uma questão geral da maioria das estratégias de síntese concatenativa (Teixeira, 1998). Com isso o projeto buscou realizar um sistema de fala neutra, onde se preocupa com a pronúncia regular da palavra com a devida acentuação.

3 METODOLOGIA

De acordo como Marconi e Lakatos (2010), a metodologia do trabalho científico é a parte em que é feita uma descrição minuciosa e rigorosa do objeto de estudo e das técnicas utilizadas nas atividades de pesquisa.

Diante disso, esse capítulo trata-se da metodologia utilizada no trabalho. A primeira seção descreve os procedimentos adotados no trabalho. Nas seções subsequentes são apresentados os procedimentos para o desenvolvimento do software e a operacionalidade do sistema.

3.1 Procedimentos adotados

São apresentados os procedimentos adotados para o desenvolvimento da ferramenta proposta. Tais procedimentos têm como pilar sustentador a definição dos aspectos concernentes ao sistema.

3.1.1 Requisitos funcionais e não funcionais do sistema

Para realização da ferramenta de conversão de texto em fala, tornou-se necessário inicialmente realizar levantamento dos requisitos que a ferramenta envolveria. Para isso estabeleceu os requisitos funcionais (RF) e não funcionais (RNF) da ferramenta, realizou-se através da documentação de elicitação. Após essa especificação determinou-se que ferramenta envolveria os seguintes requisitos:

a) Permitir que o usuário configure qual idioma deverá ser utilizado: espanhol, inglês ou português - RF;

b) obter informação textual de determinados componentes de tela do *Microsoft .Net Framework* - RF;

c) fazer a conversão texto-fala das informações textuais obtidas das aplicações - RF;

d) permitir a importação de arquivo .txt – RF;

e) permitir que o usuário execute testes de conversão texto-fala, digitando o texto a ser sintetizado e selecionando se a ferramenta deve gerar a saída em áudio ou mostrar o resultado do processamento do texto - RF;

f) executar a conversão de texto-fala de maneira assíncrona - RF;

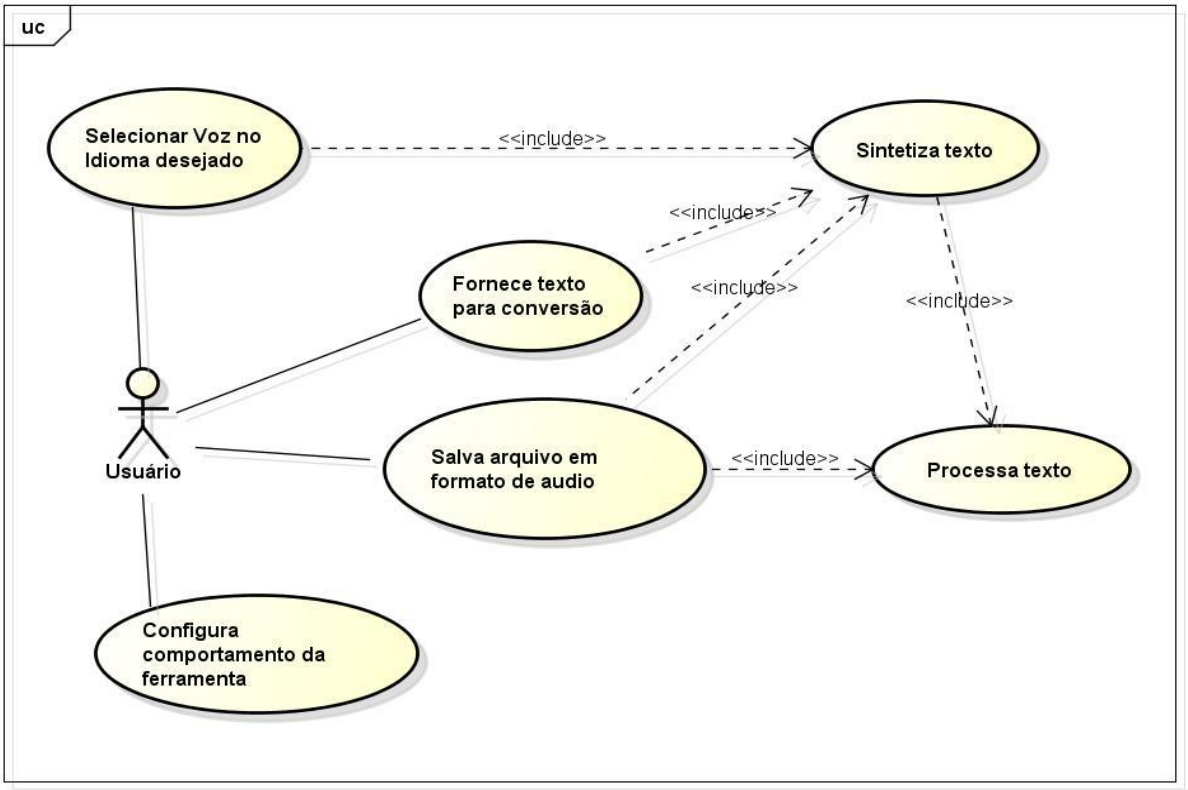
g) funcionar com o sistema operacional Windows XP ou superior - RNF;

h) demonstrar o status da interface - RNF.

3.1.2 Diagramas de casos de uso

Após a etapa de levantamento dos requisitos da ferramenta, realizou-se a criação dos diagramas de caso de uso, com a finalidade de estabelecer as principais funcionalidades da ferramenta proposta. Este a diagrama é descrito na Figura 3.

Figura 3 – Diagrama de caso de uso



Fonte: Elaborado pelos autores

O primeiro caso de uso descreveu as configurações que o usuário pode aplicar na ferramenta. Há uma dependência entre os casos de uso, pois primeiro o usuário deve selecionar o idioma dentre os disponíveis no sistema para depois, realizar a sintetize do texto em fala. Sem esta seleção a ferramenta torna-se impossibilitada de converter os textos em áudio. Ao obter informações textuais da aplicação, as mesmas são lidas pelo sintetizador de áudio, através do caso de uso "Sintetiza texto", que por sua vez utiliza o caso de uso "Processa texto" para processar qualquer texto antes de sintetizá-lo.

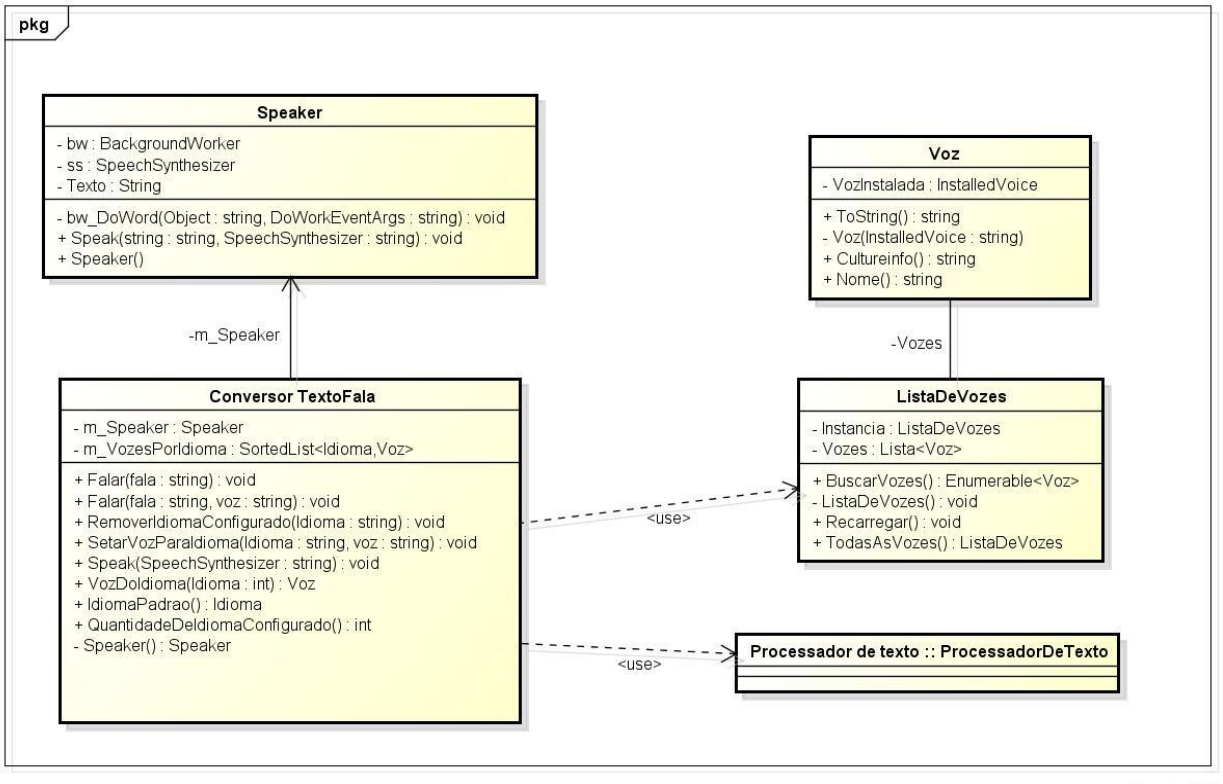
O caso de uso “Configura comportamento da ferramenta”, representa as configurações que o usuário pode aplicar na ferramenta. O usuário pode configurar o idioma a ser utilizado pela ferramenta (português, inglês e espanhol), sendo que esta configuração impacta diretamente na forma como a ferramenta processa os textos. Também é possível configurar a velocidade e volume da execução da ferramenta.

No caso de uso Sintetiza texto, é apresentado o funcionamento do sintetizador de texto. A síntese é feita de forma paralela e assíncrona. Ou seja, enquanto um texto é sintetizado as demais funcionalidades da ferramenta podem ser utilizadas paralelamente. Caso o sintetizador de áudio receba um novo texto para sintetizar, enquanto outro texto está sendo sintetizado, este segundo texto recebido será simplesmente ignorado.

3.1.3 Diagramas de classes

Para auxílio na implementação do sistema, realizou-se os diagramas de classes, com a finalidade de identificar as principais classes que compõem a ferramenta e a forma como estas estão estruturadas e interligadas. Na Figura 4, são mostrados três diagramas de classes, que apresentam as classes envolvendo as principais funcionalidades da ferramenta.

Figura 4 – Diagrama de classe



Fonte: Elaborado pelos autores

A primeira destas três funcionalidades é o processamento de textos. A classe principal neste contexto é a classe "ProcessadorDeTexto". Esta classe recebe um texto para ser processado através de seu único método público "ProcessarTexto". No processamento de texto, determinaram-se classes auxiliares onde terá as principais variáveis para sintetização do texto.

O sintetizador de textos tem como classe principal a classe "ConversorTextoFala". Esta classe recebe um texto para ser sintetizado e utiliza a classe "ProcessadorDeTexto" para processá-lo antes da síntese. Esta classe também guarda uma lista de idiomas (representados pela enumeração idioma), cada um com sua respectiva voz associada. Estas vozes que podem ser associadas ao idioma, são as vozes instaladas no próprio SO. Para a síntese do texto é utilizada a classe "*Speaker"*. Esta classe implementa a chamada à classe *Speech.Syntesizer* da *Microsoft*. *Net Framework* que vai pronunciar o texto.

3.1.4 Componentes de interface suportados

Após a fase de especificação do software e criação dos diagramas de caso de uso realizou a seleção de componentes que seriam suportados pela ferramenta, uma vez que uma aplicação *Windows Forms* desenvolvida utilizando a *Microsoft .Net Framework* pode valer-se de diversos componentes de interface já implementados no próprio *framework*.

Além disso, a *Microsoft Visual Studio (Integrated Development Enviroment - IDE*), mais comumente usado para desenvolvimento deste tipo de aplicação, permite que o usuário crie seus próprios componentes de forma fácil e rápida. Dentre desta diversidade de componentes disponíveis, além da possibilidade de se criar novos componentes, foi listado no Quadro 1 os componentes escolhidos.

Quadro 1 - Relação dos componentes interpretados pela ferramenta

COMPONENTES			
Button	GroupBox	RadioButton	SaveFileDialog
CheckBox	Label	TableLayoutContainer	OpenFileDialog
CkeckedListBox	ListBox	TrackBar	MenuStrip
ComboBox	Panel	TextBox	RichTextBox

Fonte: Elaborado pelos autores

No Apêndice A encontra-se descrição de cada componente citado, incluindo as informações textuais que são sintetizadas pela ferramenta. Além disso, são apresentadas as ações do usuário que disparam os eventos interceptados. Determinados eventos disparados por estes componentes indicam que a informação textual associada a estes componentes deve ser sintetizada, como por exemplo, quando o usuário tentar sintetizar um texto sem seleção da voz nos idiomas disponíveis.

3.2 Procedimento de implementação

Nesta seção são apresentadas informações sobre as técnicas e ferramentas utilizadas para a implementação da ferramenta, a interpretação de interfaces e dos eventos no processamento de textos e da conversão de texto em áudio e a operacionalidade da ferramenta.

3.2.1 Técnicas e ferramentas utilizadas

Após o processo de especificação, realizou-se o desenvolvimento do sistema utilizando a ferramenta *Microsoft Visual Studio* 2010 *Professional*. Inicialmente, tornou-se necessário a instalação da biblioteca *System.Speech*. Essa biblioteca possibilitou a transição de texto em fala, reconhecimento da gramática, manipulação da entrada de áudio, manipulação da saída de áudio, conversão de texto em áudio e fragmentação da entrada de áudio.

Após os procedimentos apresentados no parágrafo anterior, realizou-se a instalação dos pacotes de voz que seriam interpretados pelo sistema. Como padrão, o pacote de voz do SO *Windows* é no idioma inglês. Os três pacotes de voz dos SO mais utilizados são: *Microsoft Zira Desktop, Microsoft Mike Desktop, Microsoft Ana Desktop*. Dentro estes pacotes de voz, a que apresenta melhor inteligibilidade em termos de transcrição texto-fala é o pacote *Microsoft Zira Desktop*, diante disso adotou-se a voz no sistema.

Para realizar a conversão texto-fala com uma voz voltada para o português fez-se necessária à aquisição de uma voz própria no idioma português. Dessa forma foi utilizada o pacote de voz no idioma português brasileiro desenvolvida pela empresa *NextUp Technologies* (NEXTUP TECHNOLOGIES, 2013). Para sintetize do texto no idioma espanhol, utilizou-se o pacote de voz *Microsoft Helena Desktop* adquirida também da referida empresa. Todos os pacotes de voz mencionados foram adquiridos gratuitamente da empresa.

Cada voz é reconhecida pelo sistema operacional como *software* que depois de instalado, torna-se disponível como pacote das vozes do sistema, sendo compatíveis com a biblioteca *System.Speech*, utilizada para fazer a síntese de voz. Entretanto, para o reconhecimento das vozes no sistema operacional *Windows* é necessário à realização das alterações no registro do sistema. Para isso, no editor de registro do referido sistema operacional, é necessário localizar a voz instada e exportá-la salvando em extensão.*reg*, sendo que essa extensão é padrão no editor de registro do SO. Após esse processo, constitui imprescindível mudar a localização onde a voz foi instalada tornando-a parte integrante do arquivo onde são armazenados todos os pacotes de voz do mesmo.

Tal procedimento é realizado quando é instalado qualquer pacote de voz. No Quadro 2 é exemplificado o procedimento elucidado no processo de reconhecimento da voz *Nuance Raquel Brasilian Female.*

Quadro 2 – Exportação do pacote de voz Raquel

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\TTS_MS_pt-
BR_Raquel_10.0]
@="Microsoft Server Speech Text to Speech Voice (pt-BR, Raquel)"
"416"="Microsoft Server Speech Text to Speech Voice (pt-BR, Raquel)"
"CLSID"="{a12bdfa1-c3a1-48ea-8e3f-27945e16cf7e}"
"LangDataPath"="C:\\Program Files (x86)\\Common Files\\Microsoft
Shared\\Speech\\Tokens\\TTS_MS_pt-BR_Raquel_10.0\\MSTTSLocptBR.dat"
"VoicePath"="C:\\Program Files (x86)\\Common Files\\Microsoft
Shared\\Speech\\Tokens\\TTS_MS_pt-BR_Raquel_10.0\\HRaquelT"
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\
Tokens\TTS_MS_pt-BR_Raquel_10.0\Attributes]
@=""
"Age"="Adult"
"Gender"="Female"
"Language"="416"
"Name"="Microsoft Server Speech Text to Speech Voice (pt-BR, Raquel)"
"Vendor"="Microsoft"
"Version"="10.0"
```

Fonte: Elaborado pelos autores

3.2.2 Principais procedimentos do algoritmo

Para o processo de desenvolvimento da ferramenta de conversão de texto-fala adicionou-se a referência *System.Speech* no projeto. Dentro do *namespace* responsável pelo sintetizador de voz, a classe fundamental para síntese é a *Speech. Synthesizer*. Dessa forma, instanciou-se a classe, utilizando seu método *Speak*. Tais procedimentos são responsáveis pela sintetize do texto em fala.

Para geração da lista de voz, utilizou o método *"GetInstalledVoices"*. Para esse processo, torna-se necessário, os *namespaces using System.Speech.Synthesis* e *using System.Speech.AudioFormat*. Ao instanciar um objeto do tipo *"ComboBox"*, via parâmetro, para este método, resultou-se em caixa de seleção, onde o usuário seleciona a voz desejada. O procedimento responsável pela geração de voz é descrito no Quadro 3.

Quadro 3 – Método GetInstalledVoices

```
speaker.SpeakCompleted += new EventHandler<SpeakCompletedEventArgs>
(speaker_SpeakCompleted);
        Pause.Enabled = false;
        foreach (InstalledVoice voice in speaker.GetInstalledVoices())
        {
           VoiceInfo info = voice.VoiceInfo;
           string detalhesVoz = "Nome : "
           + info.Name +
           "- Idioma:" + info.Culture + " -
            Idade: "+ info.Age +
           "Gênero:" + info.Gender +
            "- Descrição:" +
           info.Description;
           cmbVoz.Items.Add(voice.VoiceInfo.Name);
        }
```

Fonte: Elaborada pelos autores

Dentro da aplicação é possível a importação de arquivo em formato .txt. Para isso, é preciso a utilização do componente *OpenFileDialog*, importando em todos os formatos .txt existentes. Esse procedimento é descrito no Quadro 4.

Quadro 4 – Importando arquivo em formato .txt

```
private void button1_Click(object sender, EventArgs e)
 {
  label3.Text = "Importando arquivo";
  try {
        OpenFileDialog ofd = new OpenFileDialog()
        ofd.CheckFileExists = true; ofd.CheckPathExists = true;
        ofd.DefaultExt = "txt"; ofd.DereferenceLinks = true;
        ofd.Filter = "Text files (*.txt)|*.txt|" + "RTF files (*.rtf)|*.rtf|" +
       "Works 6 and 7 (*.wps)|*.wps|" + "Windows Write (*.wri)|*.wri|" +
       "WordPerfect document (*.wpd)|*.wpd";
        ofd.Multiselect = false; ofd.RestoreDirectory = true;
        ofd.ShowHelp = true; ofd.ShowReadOnly = false;
        ofd.Title = "select a file"; ofd.ValidateNames = true;
        if (ofd.ShowDialog() == DialogResult.OK) {
        StreamReader sr = new StreamReader(ofd.OpenFile());
        richTextBox1.Text = sr.ReadToEnd();
    }
  }
  catch (Exception )
  {
       MessageBox.Show("can not open the file", "Text to Speak", MessageBoxButtons.OK,
       MessageBoxIcon.Error);
  }
}
```

Fonte: Elaborada pelos autores

A etapa do processamento do texto aplica uma série de transformações em um texto de entrada recebido, visando melhorar a pronúncia das palavras feita pela etapa da síntese. O resultado desta transformação consiste em um texto padronizado, no qual são substituídas determinadas construções linguísticas por suas formas simplificadas que podem ser lidas como palavras comuns.

Para processamento de texto realizam-se dois métodos distintos. Cria-se uma classe chamada *play*, onde se inicialmente instancia-se o método (*speaker.SelectVoice(cmbVoz.Tex*t) para seleção de voz, e posteriormente através dos eventos trackBar, configura-se a velocidade e volume da voz. Esse procedimento é descrito no Quadro 5.

Quadro 5 – Método *void play*

```
private void Play(string words)
{
        speaker.SelectVoice(cmbVoz.Text);
        speaker.Volume = trackBar1.Value;
        speaker.Rate = trackBar2.Value;
        speaker.SpeakAsync(words);
    }
```

Fonte: Elaborada por autores

Após a criação da classe "*Play"*, a mesma foi instanciada na classe responsável para conversão do texto em fala. Para tal procedimento, tornou-se necessária inicialmente a verificação da existência de vozes para conversão texto-fala. Logo após o método "*Play*", é instanciado. Caso exista uma voz configurada para o idioma, é utilizada outra sobrecarga do método "Falar", que também utiliza um objeto de *SpeechSyntesizer* selecionando uma voz específica, processando o texto e chamando o método "Speak". Esse procedimento é descrito no Quadro 6.

Quadro 6– Método para conversão texto em fala

```
 private void button2_Click(object sender, EventArgs e)
{ if (cmbVoz.SelectedIndex >= 0){
   label3.Text = "Falando"; Speak.Enabled = false; Pause.Enabled = true;
   Play(richTextBox1.Text);
}
Else {
        MessageBox.Show("Por favor, selecione uma voz", "Text to Speech",
        MessageBoxButtons.OK, MessageBoxIcon.Warning); cmbVoz.Focus(); }
}
```

Fonte: Elaborada pelos autores

A conversão de texto em áudio é o processo da ferramenta responsável por receber um texto de entrada e pronunciar estas palavras de acordo com o idioma selecionado. Para realizar esta pronúncia, é utilizada a voz configurada para aquele idioma na ferramenta. As vozes disponíveis para serem utilizadas pela ferramenta são todas as vozes instaladas no SO.

Como as vozes instaladas podem variar de um sistema para outro, a ferramenta não assume nenhuma voz como padrão para determinado idioma. Caso o usuário não configure uma voz para o idioma selecionado, a ferramenta utilizará a voz que, na própria configuração do SO, estiver configurada como voz padrão sendo essas vozes em português, inglês e espanhol.

O método "speak" permitir ouvir o texto no mesmo instante ou gerar um arquivo de áudio. Ao setar a "speaker.SetOutputToWaveStream", informa-se ao sintetizador que ele deve salvar a narração em um arquivo .wav.

Não há necessidade de o texto ser processado antes de ser passado ao sintetizador de áudio. A própria classe responsável por comandar a síntese força o processamento do texto. Desta forma, existe a garantia que um texto não será pronunciado sem passar pelo processo que busca melhorar a legibilidade do mesmo. Esse procedimento é descrito no Quadro 7.

Quadro 7 – Método classe para geração de arquivo em áudio

```
private void button5_Click(object sender, EventArgs e)
    {
       label3.Text = "Exportando arquivo";
       SaveFileDialog sfd = new SaveFileDialog();
       sfd.Filter = "All files (*.*)|*.*|wav files (*.wav)|*.wav";
       sfd.Title = "Save to a wave file";
       sfd.FilterIndex = 2;
       fd.RestoreDirectory = true;
       if (sfd.ShowDialog() == DialogResult.OK)
       {
              FileStream fs = new speaker.SetOutputToWaveStream(fs);
              FileStream(sfd.FileName,FileMode.Create,FileAccess.Write);
              speaker.Speak(richTextBox1.Text); fs.Close();
       }
  }
```

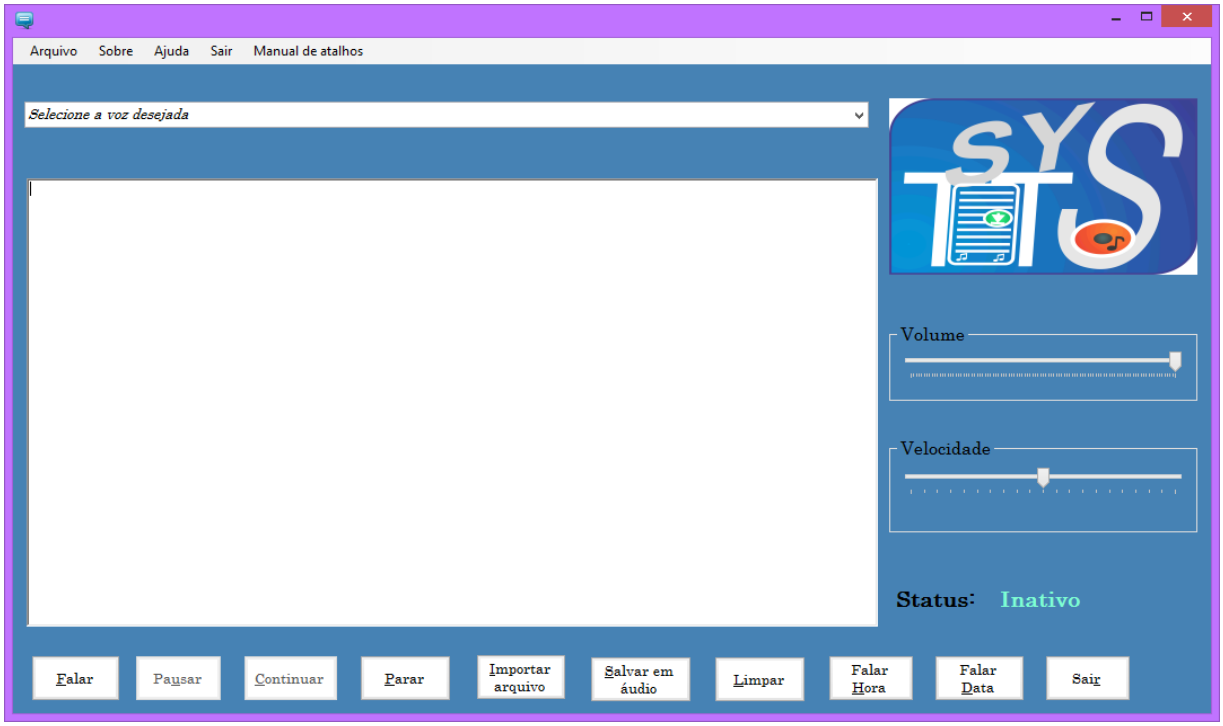
Fonte: Elaborada pelos autores

O sistema também permite o processamento de data. As datas encontradas são transformadas em suas formas extensas. É respeitado o idioma selecionado, fazendo a transformação de forma diferente para datas em português, inglês ou espanhol. O sistema permite também o processamento de hora em idioma selecionado pelo usuário.

3.3 Operacionalidade da interface

Dentro do sistema, o usuário deverá inserir ou exportar o texto que deseja realizar a conversão de texto em fala. Após a digitação do texto, o usuário acionará o botão de leitura e, com isso, o texto é capturado pelo sistema fonético em português, inglês, e espanhol. Nesse processamento, respeitam-se acentuações, espaços, vírgulas e a norma culta da língua portuguesa. Na figura 5 abaixo é ilustrado a tela principal do sistema.

Figura 5 – Interface do sistema



Fonte: Própria do sistema

Programou-se um botão para interromper o processo de fala, caso o usuário julgue necessário. Também é possível que os textos convertidos em fala sejam gerados em arquivo de áudio com as mesmas propriedades do texto lido.

O sistema também contempla uma opção onde o usuário poderá realizar as configurações desejadas, como por exemplo: controle de velocidade e volume da aplicação. O sistema ainda possui uma opção de “Ajuda” onde o usuário poderá esclarecer algumas de suas dúvidas sobre o mesmo. O sistema possui teclas de atalhos para melhor eficiência no processamento textual na utilização de *software* leitores de tela.

4 RESULTADOS E DISCUSSÕES

A ferramenta desenvolvida conseguiu obter informações textuais e sintetizar estas informações conforme proposto. O fato da ferramenta utilizar-se de tecnologias de síntese de voz desenvolvidas por terceiros possibilitou a utilização de tecnologias mais apuradas, que tornaram a pronúncia mais natural, tanto no idioma português como no idioma inglês e espanhol.

Para transmissão da fala, instalaram-se inúmeros pacotes de vozes empregadas apenas para testes de segmentação. Porém essas não apresentaram bons resultados quando a inteligibilidade, coerência e pronúncia para transcrição dos textos no idioma português brasileiro. Diante disso optou-se pela voz *Nuance Raquel Brasilian Female*. A maioria das dificuldades enfrentada nesse procedimento foi fazer com que SO reconhecesse a voz instalada.

O processamento textual aplicado pela ferramenta mostrou-se capaz de melhorar o resultado final da leitura. Nesse processo, o sistema apresenta diferenciação na entoação da pontuação, ou seja, há uma diferenciação na sintetize de um ponto de interrogação e de ponto final.

Entretanto, a ferramenta não apresentou uma entoação dos textos de acordo com todos os procedimentos gramaticais como substituição das abreviações, as siglas e os símbolos especiais em sua forma extensa, identificando a sílaba tônica de cada palavra, ritmo de fala presente em palavras e orações.

Porém tais características não são presente na maioria dos sistemas TTS devido à complexidade da realização de tais procedimentos, o que implica que essas tecnologias requerem avanços para que aproxime cada vez na inteligibilidade de interfaces máquina-homem. Os avanços apreciáveis apresentadas na ferramenta seria a aproximação de uma linguagem que se aproxima consideravelmente linguagem humana.

Quando a ferramenta recebe um texto mais extenso para ser lido da aplicação secundária, este texto será lido até o fim. Isso significa que, mesmo que o usuário interaja com outros componentes da interface enquanto o texto é lido, as informações textuais destes outros componentes não serão lidas, pois a ferramenta estará em status “ocupada” terminando a primeira leitura. Essa característica se deve ao fato do processamento textual em sistemas TTS processar um texto irrestrito e depois realizar sua conversão, sendo impossível interceptar um processamento textual que já está sendo processado.

O trabalho de Gomes (1998) foi o único entre os correlatados que se utilizou de técnicas de IA para fazer a conversão texto-fala e o único a desenvolver um sintetizador de áudio próprio. Já o trabalho de Chbane (1994) propôs uma modelagem com todos os procedimentos essenciais para conversão de texto em fala, utilizando a modelagem prosódica, porém não realizou a implementação do sistema. Todos os trabalhos apresentados são passíveis de extensão, e não permite a interceptação de uma leitura de um texto para que outro seja lido, apenas para a narração do texto.

A ferramenta desenvolvida realiza o processamento textual semelhante a essas ferramentas correlatadas. Porém semelhante a essas ferramentas, a interação como *software* leitores de tela não apresenta resultados expressivos. Além de atender a um maior número destas características, a ferramenta desenvolvida destaca-se pela qualidade da voz sintetizada e pela sua capacidade de gerar os arquivos processados em áudio.

5. CONCLUSÕES

Os resultados obtidos ao término do trabalho são satisfatórios. A ferramenta foi capaz de realizar a síntese dos textos em fala em tempo real nos idiomas português, inglês e espanhol, aumentando assim sua utilidade prática. O estudo dos sistemas de conversão texto-fala possibilitou um melhor entendimento de como estes sistemas devem funcionar e como limitar o escopo da ferramenta desenvolvida.

O desenvolvimento da ferramenta tornou-se possível através da utilização da biblioteca *System.Speech*. Esta biblioteca foi capaz de prover todos os recursos necessários para a síntese de voz.

Os objetivos propostos foram atendidos visto que a ferramenta é capaz de solicitar ao usuário qual voz ele deseja para realizar o processamento textual. É possível também obter informações textuais externos através da importação de arquivo .txt e armazenar os arquivos em áudio para que os mesmos possam ser acessados posteriormente.

A experiência adquirida nesse trabalho representa o aprendizado de uma técnica de programação eficiente. O estudo para desenvolvimento da ferramenta ressaltou a importância da responsabilidade em desenvolver um sistema TTS com naturalidade e inteligibilidade para que o mesmo possa ser aceito pelo usuário. Merece destaque a acessibilidade que esse sistema pode trazer para os deficientes visuais e vocais.

Em relação às técnicas de programação, é sugerido que o sistema seja rigorosamente analisado, conforme uma técnica específica UML (*Unified Model Language*), permitindo melhor documentação, modularização, atualização, manutenção e outras vantagens que um sistema bem documentado e estruturado permite.

Diversos estudos podem ser realizados futuramente como estudo mais aprofundado a respeito da modelagem da entonação e implementação de um modelo de geração automática da curva da frequência fundamental. Para isso também será necessário realizar a implementação de um analisador morfológico e sintático para enfatizar ou não grupos de palavras e demarcar as fronteiras prosódicas. Também será necessário um estudo da extração da semântica do texto, dado que a entonação é uma característica que acompanha a intenção do autor ao falar a frase, ou de algum sentimento (entusiasmo, decepção entre outros) que ele queira passar juntamente com a informação.

Após estudos sobre sistemas TTS, conclui-se que a ferramenta desenvolvida pode ser aprimorada envolvendo modelagem prosódica, maior naturalidade da fala e inserção de outros

idiomas. Alguns pontos da ferramenta podem ser melhorados e outras funcionalidades ainda podem ser implementadas. Como sugestão para trabalhos futuros, citam-se os seguintes itens:

a) Implementar a interpretação dos componentes que a ferramenta atualmente não é capaz de interpretar, através da utilização de alguma outra tecnologia capaz de obter informações textuais;

b) utilizar a biblioteca de conversão de texto desenvolvida para a ferramenta de forma independente em outras aplicações;

c) desenvolver um módulo de síntese de áudio próprio para que a ferramenta funcione de forma independente de tecnologias desenvolvidas por terceiros; e

d) desenvolver um módulo para permitir que o aplicativo receba comandos de voz, facilitando assim sua usabilidade.

REFERÊNCIAS

ACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. **Unified Software Development Process.** Addison-Wesley Pub Co, 2000. p. 77-82.

AZUIRSON, Gabriel A. V. **Investigação da modelagem linguistica e prosódica em sistemas de sintese de voz** . 2009. 64 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia da Computação) - Centro de Informática, Universidade Federal de Pernambuco, Recife. Disponível em: <www.cin.ufpe.br/~tg/2009-2/gava.doc>. Acesso em: 27 maio 2013.

BORGES, José A. **DOSVOX - Um Novo Acesso dos Cegos à Cultura e ao Trabalho**. 1994 – Universidade Federal do Rio de Janeiro. Disponível em: < http://www.nce.ufrj.br/aau/dosvox >. Acesso em 19 de outubro de 2013.

BRAGA, Daniela; DIAS, Miguel. **Sistemas de Conversão Texto-Fala: estado da arte, aplicações, arquitetura e desafios.** 2009. Ecola de Verão, Faculdade de Letras da Universidade do Porto. Disponivel em: < http://web.letras.up.pt/bhsmaia/EDV/apresentacoes/Braga_Texto_Fala.pdf>. Acesso em 11 de Outubro de 2013.

CAIRO, Hélio S. **Modelamento prosódico para conversão texto-fala do português falado no Brasil**. 1995. UNICAMP. Disponível em <http://www.ufjf.br/revistagatilho/files/2010/06/pereira.pdf>. Acesso em 07 de Outubro de 2013.

CHBANE, Dimas T. **Desenvolvimento de um sistema para conversão de textos em fonemas no idioma português**. 1994. 125 f. Dissertação (Mestrado em Engenharia) - Escola Politécnica, Universidade de São Paulo, São Paulo. Disponível em: <http://www.linodecampos.net/textos/disdimas.pdf>. Acesso em 15 de Janeiro de 2013.

COSTA, Evandro; GUERRERO, Dalton; MENDONÇA, Andréa. **Elicitação de Requisitos - Evidências de uma Problemática na Formação dos Estudantes de Computação**. 2008. Disponível em: <http://fees.inf.puc-rio.br/FEESArtigos/artigos/artigos_FEES08/mendonca.pdf>. Acesso em 31 de Julho de 2013.

CROCHIERE, Ronald E.; FLANAGAN, James L. **Speech Processing: an Evolving Technology.AT & T Technical Journal,** v. 65, n. 5, p. 2-11, Sept/Oct. 1990. . Disponível em: < http://onlinelibrary.wiley.com/doi/10.1002/j.1538-7305.1986.tb00374.x/abstract > Acesso em 06 de Maio de 2013.

DUARTE, Mauricio; UZAI, Gustavo. **Sistema de Reconhecimento de Voz – Aplicabilidade.** 2008. Artigo (Tecnologia em Informática para a Gestão de Negócios) - Faculdade de Tecnologia de Garça (Fatec), Garça-SP. Disponível em: <http://www.fatecgarca.edu.br/revista/Volume1/2.pdf>. Acesso em 01 de Agosto de 2013.

DUTOIT, Thierry. **A Short Introduction to Text-to-Speech Synthesis.** 1997. Artigo (Tecnologia em Informática para a Gestão de Negócios) - Faculdade de Tecnologia de Garça (Fatec), Garça-SP. Disponível em: < http://tcts.fpms.ac.be/synthesis/introtts_old.html >. Acesso em 01 de Agosto de 2013.

ESQUIVEL,A.S. **Um Sistema de Síntese de Voz. In: Congresso Nacional de Informática**, 18. São Paulo, 1985. Anais. São Paulo, Sucesu, 1985. p. 776-82.

INFANTE ,Ulisses; CIPRO, Neto. **Curso de gramática aplicada aos textos**. São Paulo: Scipione; 1999. P. 100. Disponível em <http://www.ebah.com.br/content/ABAAAAK7oAD/pasquale-c-1-neto-ulisses-infante-gramatica-lingua-portuguesa>Acesso em: 06 de Março de 2013

FERHARDT, Fernando. **Ferramenta para conversão texto-fala da interface de aplicações windows forms.** 2012. Trabalho de Conclusão de Curso (Bacharelado em Ciências da computação) submetido à Universidade Regional de Blumenau, Universidade Regional de Blumenau Centro de Ciências Exatas e Naturais. Disponível em < www.bc.furb.br/docs/MO/2012/350330_1_1.pdf >. Acesso em 21 de outubro de 2013.

FERNANDES, Cristian. **Delegates e eventos com C# .Net** . Disponível em: <http://www.devmedia.com.br/delegates-e-eventos-com-c-csharp-net/7050>. Acesso em 22 Outubro de 2013

GOMES, Leandro C. T. **Sistema de conversão texto-fala para a língua portuguesa utilizando a abordagem de síntese por regras**. 1998. 107 f. Dissertação (Mestrado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas. Disponível em < http://www.bibliotecadigital.unicamp.br/document//?code=vtls000132102&fd=y > Acesso em16 de Março de 2013.

GRUPO VIRTUOUS, **Democratizando a Educação,** 2012. Grupo Virtous Tecnologia Educacional. Disponível em < feed.snap.do/?publisher=SnapdoSoftonicYB&dpid=SnapdoSoftonicYB&co=BR&userid=665af26e-5bf9-425d-afc1-937c2de03be2&searchtype=ds&q=br&installDate=05/05/2013> . Acesso em 06 de Maio de 2013.

MAEBE, Jonas; RONSSE, Michiel; BOSSCHERE, Koen . **Precise detection of memory leaks .** (2004). Disponível em: <http://www.cs.virginia.edu/woda2004/papers/maebe.pdf>. Acesso em 08 de Outubro de 2010.

MACORATTI, José C. **VB .NET : usando os recursos do Microsoft .NET Speech.** [S.l.], 2010. Disponível em: <http://imasters.com.br/artigo/17394/dotnet/vb-net-usando-os-recursos-do-microsoft-net-speech>. Acesso em: 11 set. 2013.

LATSCH, Vagner L. **Um sistema de conversão de texto-fala para Windows.** 2002. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Escolas Engenharia, Departamento de eletrônica e de computação, Universidade Federal do Rio de Janeiro. Disponível em: <www02. Lps.ufrj.br/~sergioln/theses/dsc05vagnerlatsch.pdf>. Acesso em 20 de outubro de 2012.

KLATT, Dennis H. Software for a cascade/parallel formant synthesizer**. Journal of the Acoustical Society of America**, Melville, n. 67, p. 971-995, Mar. 1980.

MARCONI, Marina A; LAKATOS, Eva M. **Fundamentos de metodologia científica**. 7. Ed -São Paulo: Atlas, 2010. P. 35.

MAYA, Alcides. **Linguagem de Programação I (linguagem C#).** 2000. Escola Alcides Maya-Segundo módulo. Disponível em <http://www.alcidesmaya.com.br/apostilas/linguagem_c.pdf>. Acesso em 30 de Julho de 2013.

MICROSOFT CORPORATION. **PropertyDescriptor class** . Disponível em: <http://msdn.microsoft.com/pt-br/library/system.componentmodel.propertydescriptor.aspx>. Acesso em 22 outubro de 2013.

PRESSMAN, Roger S. **Engenharia de Software.** São Paulo: Pearson Makron Books, 1995. P. 786-788.

SHINYASHIKI, Roberto. **Pensador Uol Roberto Shinyashiki.** Disponível em : < http://pensador.uol.com.br/autor/roberto_shinyashiki/>. Acesso em 08 de Outubro de 2013.

OECHSLER, Thiago M. **Processamento de texto escrito em linguagem natural para um sistema conversor texto-fala** . 2009. 73 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível <http://www.inf.furb.br/tcc/index. php?cd=11&tcc=1212>. Acesso em: 20 de Março de 2012.

SILVA, Solimar S. **Um Estudo de Modelos Básicos de Prosódia para o Português Brasileiro**. Dissertação de Mestrado. 2004. 73. Disponível em: <http://www.pee.ufrj.br/teses/textocompleto/2004052801.pdf> Acesso em 07 de Outubro de 2013.

STRINGARI, Adriano F. **Processamento de rotas por um sistema conversor texto-fala**. 2010. 71 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível em: <http://www.bc.furb.br/docs/MO/2011/346532_1_1.pdf>. Acesso em: 6 de maio 2012.

VAGNES, Luis S. **Desenvolvimento de um sistema de conversão texto-fala com modelagem de prosódia.** 2006. 82 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível em: <www.ct.ufrj.br/biblioteca/teses/?p=2495> Acesso em 20 de maio de 2012.

TIOBE SOFTWARE BV. **Tiobe programming community index for July 2012. [S.l.],** 2012. Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acesso em: 11 jul. 2012.

WULFE, Benjamin. **ManagedSpy: deliver the power of Spy++ to Windows Forms with o ur new tool. [S.1.]**. 2006. Disponível em: <http://msdn.microsoft.com/en-us/magazine/cc163617.aspx>. Acesso em: 03 set. 2011

YOUNG, Sand J.; FALLSIDE, F. **Speech Synthesis from Concept: A Method for Speech Output from Information Systems.** Journal of Acoustical Society of America, v. 66, n. 3, p. 685-95, Sept. 1989.

Teixeira, J.P, Freitas, D.R.S et al. (1998). **MULTIVOX – Um conversor texto-fala para Português**. Disponível em : <http://www02.lps.ufrj.br/~sergioln/theses/msc06solimarsilva.pdf>. Acesso em 07 de Outubro de 2013.

APÊNDICE A - Relação dos componentes de interface interpretados pela ferramenta

O Quadro 10 traz o nome de cada componente interpretado pela ferramenta, incluindouma breve descrição destes e as informações textuais de cada um que são sintetizadas. Alémdisso, são apresentadas as ações do usuário que disparam os eventos interceptados pela ferramenta.

Quadro 8 – 1º Relação completa dos componentes

	Descrição, informação textual lida e eventos interceptados
GroupBox	**Descrição:** é uma caixa com uma legenda na parte superior que agrupa outros componentes dentro de si.
	Informação lida: legenda.
	Ação que ativa a leitura: Passar a seta do *mouse* sobre o (*evento MouseEnter*).
	Ação que interrompe a leitura: Tirar a seta do *mouse* do componente (*evento MouseLeave*).
Label	**Descrição:** é um rótulo textual na tela.
	Informação lida: texto.
	Ação que ativa a leitura: Passar a seta do *mouse* sobre o componente (evento MouseEnter).
	Ação que interrompe a leitura: Tirar a seta do mouse do componente (evento *MouseLeave*).
Panel	**Descrição:** É um componente que agrupa outros componentes dentro de si, sem nenhum tipo de organização automática destes componentes filhos.
	Informação lida: Este componente não possui informação textual para ser sintetizado.
	Verificações feitas pela ferramenta para este componente: A ferramenta verifica periodicamente se um novo componente filho foi adicionado ou removido em tempo de execução neste componente. Quando um novo componente filho é adicionado, a ferramenta precisa carregar suas configurações. Também é verificada a necessidade de recarregar as configurações de seus componentes filhos quando a seta do mouse é passada sobre o componente (evento MouseEnter) ou quando é movida sobre o componente (evento MouseMove).

Fonte: Elaborada pelos autores

Continuação do Quadro 8 – Relação completa dos componentes

Descrição, informação textual lida e eventos interceptados	
ProgressBar	**Descrição:** É uma barra de progresso que normalmente é associada à execução de algum processo mais demorado.
	Informação lida: O componente possui três valores associados: mínimo, máximo e atual. Quando o valor atual é igual ao valor mínimo, a barra de progresso está vazia, quando é igual ao valor máximo a barra de progresso está cheia.
	Ação que ativa a leitura: Passar a seta do *mouse* sobre o componente.
	Ação que interrompe a leitura: tirar a seta do *mouse* do componente.
RadioButton	**Descrição:** É um botão circular que pode estar marcado ou desmarcado e tem uma legenda associada.
	Informação lida: a legenda associada.
	Ação que ativa a leitura: passar a seta do *mouse* sobre o componente (evento MouseEnter).
	Ação que interrompe a leitura: tirar a seta do *mouse* do componente (evento MouseLeave).
TextBox	**Descrição:** É um caixa de texto simples.
	Informação lida: O texto contido.
	Ação que ativa a leitura: Passar a seta do mouse sobre o componente (evento MouseEnter).
	Ação que interrompe a leitura: Tirar a seta do mouse do componente (evento MouseLeave).
TrackBar	**Descrição:** É uma barra horizontal à qual é atribuído um valor mínimo e um valor máximo. Possui uma seta que pode ser movida pela barra, para representar um valor entre este mínimo e este máximo.
	Informação lida: o valor atualmente selecionado.
	Ações que ativam a leitura: É iniciada a leitura do valor selecionado.
	Ações que interrompem a leitura: − tirar a seta do mouse do componente (evento MouseLeave); − mudar o valor selecionado (evento ValueChanged). É interrompida a leitura do último valor selecionado.

Fonte: Elaborada pelos autores

APÊNDICE B – Código fonte da principal interface principal

O Quadro 9 são apresentados o código da principal da interface do sistema. Esse algoritmo trata somente do forumulário princiapal do sistema.

Quadro 9 – Código fonte da interface principal

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Speech.Synthesis;
namespace Text_To_Speak
{
   public partial class Form1 : Form {
      SpeechSynthesizer speaker = new SpeechSynthesizer();
      SpeechSynthesizer voice = new SpeechSynthesizer();
      SpeechSynthesizer reader;
      bool flag = false;
      public Form1() {
     InitializeComponent();
     speaker.SpeakCompleted += new EventHandler<SpeakCompletedEventArgs>
    (speaker_SpeakCompleted);
     Pause.Enabled = false;
     foreach (InstalledVoice voice in speaker.GetInstalledVoices())
     {
          cmbVoz.Items.Add(voice.VoiceInfo.Name);
     }
```

Fonte: Elaborada pelos autores

Continuação do quadro 9 – Código fonte da interface principal

```
void speaker_SpeakCompleted(object sender, SpeakCompletedEventArgs e)
    {
       Speak.Enabled = true;
       Pause.Enabled = false;
    }
    private void Form1_Load(object sender, EventArgs e)
    {
       richTextBox1.Text = "";
       comboBox1.Text = "NotSet";
       Pause.Enabled = false;
       Resume.Enabled = false;
    }
    private void Play(string words)
    {
       speaker.SelectVoice(cmbVoz.Text);
       speaker.Volume = trackBar1.Value;
       speaker.Rate = trackBar2.Value;
       speaker.SpeakAsync(words);
    }
    private void button1_Click(object sender, EventArgs e)
    {
     try
     {
          label3.Text = "Importando arquivo";
          OpenFileDialog ofd = new OpenFileDialog();
          ofd.CheckFileExists = true;
          ofd.CheckPathExists = true;
           ofd.DefaultExt = "txt";
          ofd.DereferenceLinks = true;
          ofd.Filter = "Text files (*.txt)|*.txt|" +
          "RTF files (*.rtf)|*.rtf|" +
           Works 6 and 7 (*.wps)|*.wps|" +
```

Fonte: Elaborada pelos autores

Continuação do quadro 9 – Código fonte da interface principal

```
            Windows Write (*.wri)|*.wri|" +
           "WordPerfect document (*.wpd)|*.wpd";
            ofd.Multiselect = false;
            ofd.RestoreDirectory = true;
            ofd.ShowHelp = true;
            ofd.ShowReadOnly = false;
             ofd.Title = "select a file";
             ofd.ValidateNames = true;
            if (ofd.ShowDialog() == DialogResult.OK)
           {
              StreamReader sr = new StreamReader(ofd.OpenFile());
              richTextBox1.Text = sr.ReadToEnd();
              byte[] bytes = Encoding.Default.GetBytes(teste);
              = Encoding.UTF8.GetString(bytes);
              richTextBox1.Text = teste;
              StreamReader sr = new StreamReader(ofd.OpenFile());
              string unicodeString = sr.ReadToEnd();
              Encoding iso = Encoding.GetEncoding("ANSI");
              Encoding utf8 = Encoding.UTF8;
              string msg = iso.GetString(utf8.GetBytes(unicodeString));
              richTextBox1.Text = msg;
              byte[] myUTF8Bytes =                -
ASCIIEncoding.Convert(Encoding.ASCII,Encoding.UTF8, myASCIIBytes);
              Console.WriteLine(BitConverter.ToString(myUTF8Bytes));
           }
        }
        catch (Exception )
        {
                MessageBox.Show("can not open the file", "Text to Speak",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
```

Fonte: Elaborada pelos autores

Continuação do quadro 9 – Código fonte da interface principal

```
}
private void button2_Click(object sender, EventArgs e)
{
  if (cmbVoz.SelectedIndex >= 0)
   {
     label3.Text = "Falando";
     Speak.Enabled = false;
     Pause.Enabled = true;
     Play(richTextBox1.Text);
   }
   else
   {
      MessageBox.Show("Por favor, selecione uma voz", "Text to Speech",
     MessageBoxButtons.OK, MessageBoxIcon.Warning);
      cmbVoz.Focus();
   }
}
private void button3_Click(object sender, EventArgs e)
{
   speaker.Pause();
   label3.Text = "Pausando fala";
   Resume.Enabled = true;
}
private void button4_Click(object sender, EventArgs e)
{
   speaker.Resume();
   label3.Text = "Continuando fala";
}
private void button5_Click(object sender, EventArgs e)
{
   label3.Text = "Exportando arquivo";
```

Fonte: Elaborada pelos autores

Continuação do quadro 9 – Código fonte da interface principal

```
    SaveFileDialog sfd = new SaveFileDialog();
    sfd.Filter = "All files (*.*)|*.*|wav files (*.wav)|*.wav";
    sfd.Title = "Save to a wave file";
    sfd.FilterIndex = 2;
    sfd.RestoreDirectory = true;
    if (sfd.ShowDialog() == DialogResult.OK)
  {
     FileStream fs = new FileStream(sfd.FileName,FileMode.Create,FileAccess.Write);
            speaker.SetOutputToWaveStream(fs);
            speaker.Speak(richTextBox1.Text);
            fs.Close();
        }
   }
private void button6_Click(object sender, EventArgs e)
{
    label3.Text = "Limpando tela";
    richTextBox1.Text = "";
}
private void button7_Click(object sender, EventArgs e)
{
    Application.Exit();
}
private void button8_Click(object sender, EventArgs e)
{
    label3.Text = "Parando a fala";
    speaker.SpeakAsyncCancelAll();
}
private void sobreToolStripMenuItem_Click(object sender, EventArgs e)
{
   Form2 frmSobre = new Form2();
   frmSobre.ShowDialog();
}
```

Fonte: Elaborada pelos autores

Continuação do quadro 9 – Código fonte da interface principal

```
private void ajudaToolStripMenuItem_Click(object sender, EventArgs e)
{
   Form3 frmAjuda = new Form3();
   frmAjuda.ShowDialog();
}
private void sairToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.Exit();
}
private void falarToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (cmbVoz.SelectedIndex >= 0)
   {
      Speak.Enabled = false;
      Pause.Enabled = true;
      Play(richTextBox1.Text);
   }
   else
  {
      MessageBox.Show("Por favor, selecione uma voz", "Text to Speech",
      MessageBoxButtons.OK, MessageBoxIcon.Warning);
      cmbVoz.Focus();
   }
}
private void importarToolStripMenuItem_Click(object sender, EventArgs e)
{
   try
   {
    OpenFileDialog ofd = new OpenFileDialog();
      ofd.CheckFileExists = true;
      ofd.CheckPathExists = true;
      ofd.DefaultExt = "txt";
```

Fonte: Elaborada pelos autores

Continuação do quadro 9 – Código fonte da interface principal

```
ofd.DereferenceLinks = true;
            ofd.Filter = "Text files (*.txt)|*.txt|" + "RTF files (*.rtf)|*.rtf|"
          + "Works 6 and 7 (*.wps)|*.wps|" +
            "Windows Write (*.wri)|*.wri|" +
            "WordPerfect document (*.wpd)|*.wpd";
            ofd.Multiselect = false;
            ofd.RestoreDirectory = true;
            ofd.ShowHelp = true;
            ofd.ShowReadOnly = false;
            ofd.Title = "select a file";
            ofd.ValidateNames = true;
            if (ofd.ShowDialog() == DialogResult.OK)
            {
               StreamReader sr = new StreamReader(ofd.OpenFile());
               richTextBox1.Text = sr.ReadToEnd();
            }
         }
         catch (Exception)
         {
            MessageBox.Show("can not open the file", "Text to Speak",
MessageBoxButtons.OK, MessageBoxIcon.Error);
         }
      }
      private void exportarToolStripMenuItem_Click(object sender, EventArgs e)
      {
         SaveFileDialog sfd = new SaveFileDialog();
         sfd.Filter = "All files (*.*)|*.*|wav files (*.wav)|*.wav";
         sfd.Title = "Save to a wave file";
         sfd.FilterIndex = 2;
         sfd.RestoreDirectory = true;
         if (sfd.ShowDialog() == DialogResult.OK)
```

Fonte: Elaborada pelos autores

Continuação do quadro 9 – Código fonte da interface principal

```
{
            FileStream fs = new FileStream(sfd.FileName, FileMode.Create,
FileAccess.Write);
            speaker.SetOutputToWaveStream(fs);
            speaker.Speak(richTextBox1.Text);
            fs.Close();
        }
    }

    private void button9_Click(object sender, EventArgs e)
    {
      label3.Text = "Informando Hora";
      Play("A hora é " + DateTime.Now.ToShortTimeString());
    }

    private void button1_Click_1(object sender, EventArgs e)
    {
       label3.Text = "Informando Data";
       Play("Hoje é " + DateTime.Now.ToShortDateString());
    }
    private void manualDeAtalhosToolStripMenuItem_Click(object sender, EventArgs e)
    {
       Form4 Atalhos= new Form4();
       Atalhos.ShowDialog();
    }
  }
}
```

Fonte: Elaborada pelos autores