Calebe Augusto dos Santos
Gino Dornelles

Text-To-Speech: Sintetizador de Voz para Documentos como Ferramenta de Apoio a Deficientes Visuais

Florianópolis – SC
2008

Calebe Augusto dos Santos
Gino Dornelles

Text-To-Speech: Sintetizador de Voz para Documentos como Ferramenta de Apoio a Deficientes Visuais

Orientador
Ricardo Pereira e Silva

Universidade Federal de Santa Catarina
Departamento de Informática e Estatística

Florianópolis – SC
2008

Trabalho de conclusão de curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para obtenção do grau de bacharel em Sistemas de Informação.

Profº. Dr. Ricardo Pereira e Silva
Orientador

Profº. Dr. Frank Augusto Siqueira
Universidade Federal de Santa Catarina

Profº. Dr. Mário Antônio Ribeiro Dantas
Universidade Federal de Santa Catarina

Dedicamos a Deus (Pai Criador), a nossas famílias (que nos criaram), e a nossos amigos (com os quais criamos laços).

Agradecimentos

Agradecemos a Deus pela saúde e por estarmos vivos e plenos de energia.

A todos os professores que surgiram em nosso caminho durante esta passagem pela Universidade e com os quais compartilhamos grandes momentos.

Aos nossos familiares que foram o amparo e a esperança nos momentos mais conturbados e nos quais nos espelhamos para vencer.

Ao professor Ricardo pela excelência e competência na guia de nosso projeto.

Aos professores membros da banca examinadora.

“As palavras são tecidas a partir de uma multidão de fios ideológicos e servem de trama a todas as relações sociais em todos os domínios.”

Mikhail Bakhtin

Resumo

O presente trabalho visa oferecer um suporte destinado a um público específico, mas também a usuários diversos, tendo em seu fim o desenvolvimento de uma ferramenta que auxilie tanto pessoas comuns como na inclusão social de uma parcela da população brasileira considerada como portadora de deficiência visual.

Os sintetizadores de voz, conhecidos como TTS (Text-to-Speech) são responsáveis por executar a leitura, neste contexto chamada por sintetização, de textos (palavras, frases, orações). Visamos obter melhores resultados no processo de leitura aumentando o nível de inteligibilidade por parte do usuário em relação ao texto lido.

Sabemos que existem diversas ferramentas na área, porém nem todas se preocupam com este quesito, criando softwares que lêem de forma homogênea um texto inteiro, acarretando pouca legibilidade e naturalidade.

Aplicamos tecnologias com o intuito de portar o TTS com características específicas, como por exemplo, tratar com diferenças de leitura deterrninados trechos dentro de um texto (parágrafos, fim de página, cabeçalho, etc.) a fim de aumentar o significado do processo de sintetização.

Palavras-chave: TTS (Text-To-Speech), .NET Framework, SSML (Speech Synthesis Markup Language), SAPI (Speech Application Programming Interface).

Abstract

This current paper aims to provide a support for a specific audience, but also the various users, taking its order to develop a tool to help both people and social inclusion of a portion of the Brazilian population regarded as being disabled vision.

The voice synthesizer, known as TTS (Text-to-Speech) are responsible for implementing the reading, in this context called for synthesis of texts (words, phrases, sentences). Aim to achieve better results in the process of increasing the level of reading comprehension by the user on the text read.

We know that there are many tools in the area, but not all are concerned about this question, creating software that reads text on a uniform a whole, causing little plainness and readability.

Applied technologies with the aim of carrying the TTS with specific characteristics, such as dealing with differences in reading determined excerpts within a text (paragraphs, end-page headline, etc.). In order to increase the significance of the process of synthesizing .

Keywords: TTS (Text-To-Speech), .NET Framwork, SSML (Speech Synthesis Markup Language), SAPI (Speech Application Programming Interface).

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Listagens

1 Introdução

Se existe uma característica que nos define e separa dos outros animais é a linguagem, nossa capacidade de comunicação através da fala, de produzir sons dando origem a diversas línguas e dialetos, corroborando com a assertiva o fato de precisarmos de intérpretes em alguns casos.

Interagir com máquinas através da fala não é novidade atualmente, porém, a tecnologia nesta área vem sofrendo refinamentos e isso contribui para que haja uma reprodução de voz mais próxima da nossa linguagem, além de contarmos com bibliotecas e frameworks para reconhecimento vocal.

Dados de pesquisas realizadas pelo IBGE mostram que o Brasil apresenta uma parcela maior do que se supunha de deficientes visuais (IBGE, 2007), são pessoas que apresentam algum grau de cegueira e por isso precisam de apoio para ingressar na sociedade de forma plena. Embora o software não contemple a língua portuguesa inicialmente, nada impede que futuras modificações sejam inseridas de forma a abranger este quesito.

Sendo assim, este trabalho apresenta caráter social, além de abrir caminho para a utilização de ferramentas que permitam a confecção do software, como é o caso dos frameworks, que são arquiteturas desenvolvidas com o objetivo de se obter o máximo de reutilização, representando um conjunto de classes abstratas e concretas com um máximo de especialização (MATTSSON, 1996). Podemos dizer também que conforme (JOHNSON & FOOTE, 1988) framework é um conjunto de classes destinadas a uma família de problemas associados.

O tema central aborda TTS (Text-To-Speech), também conhecido como Speech Synthesis ou apenas sintetizador de voz, muito empregado na área da educação e visto aqui como uma ferramenta de apoio a deficientes visuais. Como a sintetização trata da leitura de textos e envolve sua correta compreensão e sabendo que nossa língga portuguesa é conhecida pela ampla variedade de palavras e também de regionalismos referentes ao vasto território nacional, comtemplamos alguns casos referentes a fonemas e sua pronúncia para elucidar a estrutura interna de um TTS.

Especificamente neste projeto, o TTS trabalhará na conversão de documentos em formato de texto HTML (HyperText Markup Language) para o formato de áudio, respeitando as etapas de desenvolvimento de software vistas em (SILVA, 2004) e agregando um nível de entendimento satisfatório por parte dos usuários finais.

A ferramenta pode ser visualizada através da figura 1.1, onde temos a aplicação fazendo uso do .NET Framework, utilizando uma biblioteca com classes para conversão em formato

mp3 chamada YETI e a vocalização a cargo da SAPI, seja apenas a nível desta interface ou utilizando a linguagem SSML e seus elementos para a sintetização de voz.

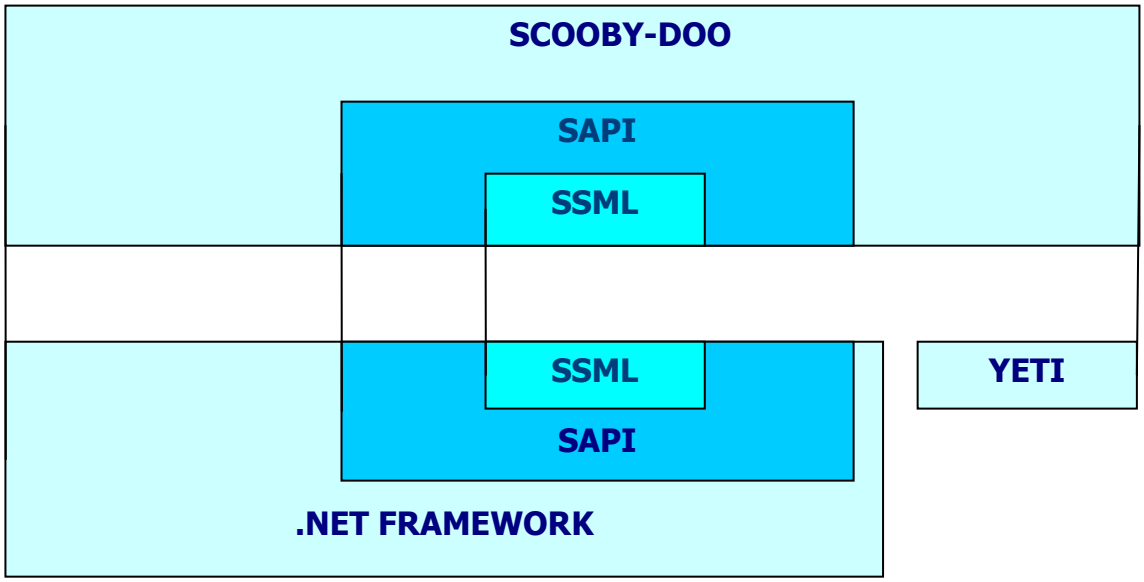


Figura 1.1 Visão geral da aplicação

1.1 Justificativa

Na confecção de software, uma das perguntas que devemos fazer é: quem vai se utilizar das funcionalidades do sistema que se pretende modelar? Qual o público-alvo?

Neste projeto temos um público específico, que terá a possibilidade de utilizar a ferramenta sem dificuldades, mas esta também terá alcance geral, pois qualquer usuário que possa digitar em um teclado usufrui de seus benefícios. É difícil imaginar para quem tem visão normal sentir o mundo sem o sentido da visão, tendo que confiar nos outros sentidos para se orientar.

Através de uma parceria da UFSC com a Microsoft, foram disponibilizados alguns softwares, entre eles, o Microsoft Visual Studio 2008, além do .Net Framework 3.5, todos extremamente úteis para o trabalho com sintetização de voz.

Existem muitas soluções hoje no mercado em se tratando de sistemas TTS, embora nem todas se preocupem em ampliar seu domínio para outros formatos de documento para vocalização (HTML, PDF, DOC). A grande maioria executa vocalização através de entrada de texto, assim como atende a requisitos como velocidade de leitura e utilização de vozes diferentes.

Temos um hiato entre a utilização de tecnologias e a acessibilidade que esta tecnologia deveria prover para seus usuários. Esta lacuna é um dos principais percalços pelos quais muitas

aplicações da área de síntese de voz esbarram ao tentar criar um elo entre o desenvolvedor e o usuário.

Embora uma pesquisa na literatura da área de projetos envolvendo o desenvolvimento de ferramentas, novas tecnologias e suporte visando à inclusão digital e posteriormente, a inclusão social no Brasil nos revele uma gama cada vez maior de utensílios para esta finalidade. Ainda assim existe uma carência na parte de entendimento do texto e na forma como ele é lido, carência realçada no motivo da voz ainda ser considerada muito robotizada e não ser dada a devida ênfase a partes do texto que a merecem, incluindo a própria língua portuguesa e suas nuances.

Este desenvolvimento da área no Brasil através da pesquisa efetuada em vários institutos nos fornece o conhecimento, mas a motivação nasce da possibilidade de melhorar e popularizar este processo, fazendo com que seja mais natural para um usuário portador de deficiência visual poder efetuar a leitura de seus documentos.

1.2 Objetivos

Podemos classificar os objetivos neste trabalho em gerais e específicos.

1.2.1 Objetivo Geral

Ao escolher este tema - TTS - como título de nosso trabalho de conclusão de curso, pretendemos primeiramente estudar a literatura na área do desenvolvimento de software de apoio a deficientes visuais, com o objetivo de criar uma ferramenta que atenda às expectativas do público alvo e que traga também mais qualidade no processo de leitura e posterior compreensão do texto falado.

1.2.2 Objetivos Específicos

Dentre os principais objetivos específicos deste trabalho, destacamos:

- Estudar o estado da arte das ferramentas de apoio a deficientes visuais;
- Estudar as características e o funcionamento dos ambientes de TTS;
- Estudar as características e o funcionamento da API SAPI;
- Estudar as características da conversão de documentos entre o formato HTML e SSML;
- Implementar uma aplicação no Visual Studio 2008 utilizando a linguagem C#.

1.3 Metodologia

Sabendo que este trabalho apresenta cunho tanto prático quanto teórico, a metodologia empregada também se divide.

Primeiramente realizamos um estudo geral na área de TTS para descobrir tendências e tecnologias já existentes, bem como carências e insuficiências no processo de leitura de textos, para melhor orientar e traçar nossos objetivos.

Na metodologia de desenvolvimento buscamos respeitar os ciclos de vida de software vistos na disciplina de Análise e Projeto de Sistemas, utilizamos uma linguagem que pudesse servir como apoio à criação da aplicação, adotamos o paradigma orientado a objeto, figurando como a escolha pela própria experiência acadêmica e de mercado, além de uma vasta obra disponível para pesquisa. Neste contexto, empregamos o ambiente Visual Studio 2008 disponibilizado pela UFSC como já mencionado.

Utilizamos a linguagem C# (CSharp) por ser utilizada em nosso ambiente de trabalho e dispor de mecanismos que tornem o desenvolvimento mais rápido e robusto além da similaridade com o Java visto no decorrer do período acadêmico.

O framework .NET também foi utilizado, assim como APIs para vocalização e XML para tratamento de dados lidos em textos, bem como outros trabalhos tendo o mesmo foco.

1.4 Estrutura do documento

Este trabalho está dividido em 9 capítulos. O capítulo 2 apresenta um panorama referente à população com deficiência visual no Brasil, dando importância aos números oriundos do censo do IBGE, além de apresentar algumas ferramentas desenvolvidas em nosso país, como o DOSVOX e a rede SACI além de um breviário sobre acessibilidade.

O capítulo 3 abrange TTS, sua definição bem como as chamadas Speech Tecnologies: Speech Recognition (SR) e Speech Synthesis (TTS), terminado com alguns desafios a serem superados na área.

O capítulo 4 dista termos da lingüística (Fonética e Fonologia) empregados no processo de leitura, interpretação e compreensão de textos. Neste capítulo objetivamos apresentar exemplos em nossa língua para incutir o desafio que é construir um TTS em português.

No capítulo 5 temos a apresentação dos sistemas operacionais com síntese de voz e da API SAPI em todas as suas versões até o presente momento e algumas distinções entre estas.

O capítulo 6 vislumbra o SSML, interessante e vital neste trabalho, expomos alguns de seus atributos para vocalização; o Voice XML e a SRGS.

O capítulo 7 traz a plataforma .Net e considerações sobre frameworks e bibliotecas.

Por fim, o desenvolvimento da aplicação e a apresentação das tecnologias chegam no capítulo 8, trazendo incluso as etapas do ciclo de vida do software.

A conclusão do trabalho e as considerações finais encontram-se no capítulo 9.

Ao fim do presente trabalho elaboramos um apêndice com a codificação do mesmo, além de uma tabela de atributos envolvendo padrões de vocalização e as classes da API para sintetização utilizadas, além de um modelo inicial usado para tratar o HTML.

2 *Deficientes Visuais*

A Organização Mundial da Saúde (OMS) considera deficiente visual a pessoa que é privada, em parte, segundo critérios pré-estabelecidos ou totalmente, da capacidade de ver. Baixa visão ou visão subnormal é o comprometimento do funcionamento visual em ambos os olhos, mesmo após correção de erros de refração comuns com uso de óculos, lentes de contato ou cirurgias oftalmológicas (EFRON, 1980).

A definição é técnica e quantitativa. Podemos dizer que baixa visão é para quem tem uma acuidade visual menor que 0,3 (Snellen). A Escala de Snellen, também conhecida como Escala Optométrica de Snellen é utilizada para fazer pré-diagnóstico da condição visual de pessoas em todo o mundo.até a percepção de luz ou, um campo visual menor que 10 graus do ponto de fixação (EFRON, 1980).

A OMS estima que a população de deficientes seja o equivalente a apenas 1% da população, fato discrepante com a realidade atual, o IBGE (Instituto Brasileiro de Geografia e Estatística) apenas recentemente se interessou por incluir no senso dados para coletar a real população de PPDs (pessoas portadoras de deficiência).

De acordo com o censo realizado em 2000, há 11.8 milhões de brasileiros com deficiência visual e desse montante, 160 mil apresentam total incapacidade para enxergar.

O avanço social da maior parte dos deficientes no Brasil é limitado pela baixa renda, acesso precário à cultura formal e políticas públicas inadequadas ao desenvolvimento social.

Esse imenso contingente encontra-se, em sua maioria, alijado do mercado de trabalho por falta de capacitação profissional (MATTAR, 2003). Um dos fatores que levam a este cenário é a ausência de uma política de incentivo a programas de capacitação de deficientes por parte das Instituições Públicas e Privadas.

A questão da responsabilidade social empresarial é um tema atual de grande importância em todo mundo, e em especial no Brasil (NERI, 2003).

Dados totais do Censo de 2000 encontram-se na tabela abaixo.

Tabela 2.1 Tipos de deficiência em valores absolutos

Tipo de deficiência	Valores absolutos	Porcentagem
Deficiência visual	16.573.937	9.8%
Incapaz de enxergar	159.824	0.1%
Grande dificuldade permanente de enxergar	2.398.472	1.4%
Alguma dificuldade permanente de enxergar	14.015.641	8.3%

Ainda sobre o censo de 2000, temos os seguintes dados vinculados nas próximas tabelas referentes ao tipo de deficiência quanto ao gênero e à distribuição em áreas urbanas e rurais.

Tabela 2.2 Tipos de deficiência por gênero

Tipo de deficiência	Total	Homens	Mulheres
Deficiência Visual	16 573 937	7 204 046	9 369 891
Incapacidade de enxergar	159 824	70 861	88 963
Grande dificuldade permanente de enxergar	2 398 472	1 027 477	1 370 995
Alguma dificuldade permanente de enxergar	14 015 641	6 105 708	7 909 932

Tabela 2.3 Tipo de deficiência em área urbana

Área Urbana

Tipo de deficiência	Total	Homens	Mulheres
Deficiência Visual	13 225 198	5 578 226	7 646 972
Incapacidade de enxergar	131 390	57 739	73 651
Grande dificuldade permanente de enxergar	1 959 617	814 122	1 145 495
Alguma dificuldade permanente de enxergar	11 134 190	4 706 364	6 427 826

Tabela 2.4 Tipo de deficiência em área rural

Área Rural

Tipo de deficiência	Total	Homens	Mulheres
Deficiência Visual	3 348 739	1 625 820	1 722 919
Incapacidade de enxergar	28 434	13 121	15 312
Grande dificuldade permanente de enxergar	438 854	213 354	225 500
Alguma dificuldade permanente de enxergar	2 881 451	1 399 344	1 482 107

Segundo o Decreto Federal n.º 914/93, um PPD é "*aquela pessoa que apresenta, em caráter permanente, perdas ou anomalias de sua estrutura ou função psicológica, fisiológica ou anatômica, que gerem incapacidade para o desempenho de atividades, dentro do padrão considerado normal para o ser humano*".

Com os novos dados, o resultado do Censo de 2000 mostrou que os números eram superiores ao que se acreditava até então, revelando que o Brasil é um país com um contingente alto de PPDs, esses números também serviram para fazer com que se respeitasse mais esse público e que houvesse um maior interesse, afinal de contas, representam um número elevado de usuários e deve-se prover mecanismos de inserção desses usuários na sociedade, seja através de sites provendo total acessibilidade, seja através de softwares especiais, pois a deficiência, se existe, é no meio, e não na pessoa.

Hoje temos vários sites com recursos para facilitar a vida dos usuários PPDs visuais, incluindo a própria página do IBGE, sites do governo e vários projetos em universidades vinculados à inserção dos PPDs, promovendo uma inclusão social através da inclusão digital.

2.1 Apoio à deficiência visual no Brasil

Em 1990 a Rede de Informações Integradas sobre Deficiências (REINTEGRA) iniciou os trabalhos na troca de informações sobre a questão da deficiência, era um trabalho envolvendo várias entidades, incluindo a USP, era também um trabalho de integração atendendo aproximadamente 30.000 usuários em 21 estados e contribuindo para criar uma base de dados sobre os usuários da rede (BORGES, 2008).

Em 1992 a Rede Nacional de Pesquisa *(*RNP), um projeto do Ministério da Ciência e Tecnologia, tinha como objetivo levar o acesso à internet para além das universidades, nascendo como resultado, em 1994, a RENDE (Rede Nacional de Comunicação entre Portadores de Deficiência). Com a RENDE, os PPDs tiveram acesso a serviços como correio eletrônico e jornais on-line.

Entre 1993 e 1994, o NCE-UFRJ desenvolveu o **DOSVOX**, um sistema sintetizador de voz que viabiliza a operação do computador por deficientes visuais, sendo conhecido no meio até hoje e passando por várias versões. Veremos mais sobre o DOSVOX no fim deste capítulo.

Em 1997, surgiu um novo projeto desenvolvido pela RNP e o NCE-UFRJ: o **INTERVOX**, uma iniciativa pioneira que permitiu acesso amplo à Internet aos deficientes visuais de todo o Brasil. O INTERVOX oferecia como serviços de aplicação, acesso a jornais, correio eletrônico, FTP e WWW em forma de texto. Atendeu aproximadamente 500 portadores de deficiência visual na época (BORGES, 2008).

Em 1999 surgia a rede **SACI**, uma parceria entre a USP/CECAE, o NCE-UFRJ, a RNP e outros colaboradores. Além de contar com um portal com notícias e links, este projeto apoiava o desenvolvimento e distribuía software gratuito voltado para a facilitação do uso de computadores por deficientes físicos e motores.

O Instituto Benjamin Constant (IBC) é um parceiro do projeto DOSVOX. A parceria com o IBC propiciou o desenvolvimento de grande parte da tecnologia de impressão Braille usada hoje no Brasil (Braille Fácil).

A Associação Brasileira de Educadores de Deficientes Visuais (ABEDEV) reúne profissionais que atuam na área da educação, habilitação, reabilitação e apoio pedagógico às pessoas portadoras de deficiência visual.

O MEC promove a inclusão dos PPDs através dos CAPs (Centros de Apoio Pedagógico), scentros para geração de material didático e impressão Braille orientados por região.

O CNPq associado ao grupo DOSVOX orginou o **INTERVOX II**, um projeto destinado a desenvolver uma série de ações visando dar novas alternativas de acesso à internet para os usuários deficientes visuais. Os parceiros deste projeto foram o NCE/UFRJ e o Instituto Benjamin Constant. Como fruto desse projeto, surgiu um navegador chamado **WEBVOX**.

Os próximos dois tópicos deste capítulo dizem respeito ao DOSVOX citado anteriormente e a rede SACI, muito importante como ferramenta de apoio à deficiência no Brasil.

2.2 DOSVOX

Dentre os vários projetos na área de TTS, o DOSVOX foi utilizado em uma disciplina do curso de Sistemas de Informação (Informática e Sociedade) acerca da inclusão digital, além de ter sido tema de um seminário na disciplina de Engenharia de Usabilidade frisando caracterísicas de assessibilidade. Desta forma tomamos conhecimento de que havia no Brasil uma preocupação em portar o deficiente visual de mecanismos de inserção. Começávamos assim a definir a idéia de nosso TCC.

Utilizamos esta seção para mostrar algumas características da ferramenta.

O DOSVOX é um sistema operacional totalmente desenvolvido no Brasil para microcomputadores do tipo PC para cegos. Sua origem dista de um aluno cego que em 1993 cursava a Universidade Federal do Rio de Janeiro (UFRJ) e sua dificuldade para lidar com questões diárias da vida acadêmica.

A primeira versão do DOSVOX rodava no ambiente DOS, surgindo o nome que batizou o sistema operacional: uma mistura de DOS, por causa do ambiente e VOX, em referência ao sintetizador de voz do software.

O sistema é um ambiente desenvolvido especialmente para cegos e que permite uma interatividade constante entre o computador e o deficiente visual. Dentre os recursos do DOSVOX está um sistema operacional, que contém todos os elementos de interface com o usuário, um sistema de síntese de fala para língua portuguesa, um editor, um leitor e um impressor/formatador de textos e outro para Braille.

Figura 2.1 Tela inicial do DOSVOX

2.3 SACI

A rede SACI nasceu em agosto de 1.999 e teve seu nome vinculado ao famoso personagem do folclore brasileiro. Pegando carona no meio de transporte utilizado pelo levado Saci Pererê e, como ele, que viaja num redemoinho, alterando a paisagem por onde passava, a SACI viaja por uma rede, a Internet e, com isso, pretende ajudar a mudar o cenário da deficiência.

Suas principais ferramentas de trabalho são a Internet e os Centros de Informação e Convivência (CICs). Através da Internet, disponibiliza os seguintes serviços aos seus usuários: endereço eletrônico, suporte técnico, softwares adaptados para deficientes, além de bases de dados, listas de discussão, agenda de eventos, entre outros.

Os Centros de Informação e Convivência são locais com computadores, softwares adaptados e monitores especializados em ministrar cursos de informática para portadores de deficiência.

O site foi construído segundo critérios de acessibilidade virtual, apresentando atualmente: oito listas de discussão (5 delas monitoradas por usuários portadores de deficiência), dois softwares especiais que podem ser utilizados gratuitamente (o Kit SACI 1, voltado para pessoas com deficiência visual, e o Kit SACI 2, para pessoas com dificuldades motoras), quatro bases de dados próprias e também acesso a informações armazenadas por outras instituições (REDE SACI, 2007).

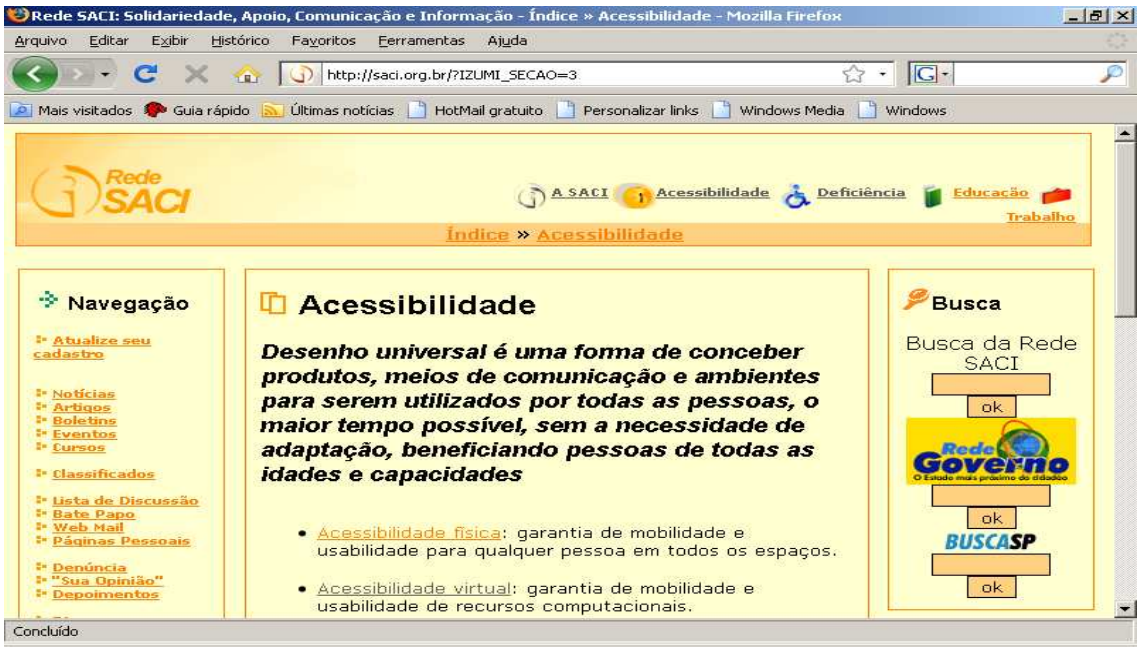


Figura 2.2 Página da Rede SACI com dicas de acessibilidade

2.4 Acessibilidade

Observar a acessibilidade de um produto consiste em considerar a diversidade de seus possíveis usuários e as peculiaridades da interação dessas pessoas com o produto, o que pode se manifestar tanto nas preferências do usuário (o que prefere ler a ouvir), quanto nas restrições à qualidade do equipamento utilizado (um usuário cuja impressora só trabalha com preto e branco). Pode ocorrer preferência na existência de necessidades educativas especiais que não podem ser ignoradas pelos desenvolvedores do produto (entre os usuários pode haver alguns que não ouçam os sons), conseqüentemente, mensagens sonoras são inadequadas para eles (TORRES, 2004).

Embora não tenhamos a intenção de tornar o software com total acessibilidade, pois para isso ele teria que ser muito mais interativo, lendo botões e menus, por exemplo, para facilitar sua utilização por deficientes, temos em mente que a simples leitura de textos será de grande valor para estes usuários em especial.

A acessibilidade é uma qualidade que se comprova a partir da satisfação de determinados requisitos, os quais estão especificados pela W3C. Selos de qualidade têm sido criados por algumas entidades para assinalar a qualidade do produto que está sendo adquirido ou utilizado (TORRES, 2004).

2.5 Considerações Finais

Embora exista uma parcela alta de PPDs no Brasil e haja muitas entidades dispostas a apoiar estas pessoas de alguma forma, seja criando centros de convivência ou até um sistema operacional para cegos, ainda há muito o que realizar, seja no próprio processo de sintetização,

onde a leitura de textos nem sempre é compreendida plenamente, seja na falta de acessibilidade promovida por alguns softwares, que se preocupam apenas em ler o que é digitado diretamente do teclado, não cuidando de fornecer mecanismos para o cego poder acompanhar a leitura em outros formatos, como no caso de um estudante, por exemplo, que fica impossibilitado de ler os capítulos das aulas pois estas figuram em formatos diversificados.

Realizando a leitura de documentos em HTML pretendemos também fazer com que este estudante possa se utilizar da aplicação para dar prosseguimento aos seus estudos, garantindo assim acesso à informação.

3 Speech Tecnologies

Existem basicamente duas tecnologias para leitura e interpretação (Speech Tecnologies): Speech Recognition (SR) e Speech Synthesis. Esta última é comumente chamada de Text-To-Speech ou, popularmente, TTS, uma vez que o discurso normalmente é sintetizado a partir de dados em textos (DUTOIT, 2007).

A informação transmitida pelos sintetizadores pode vir de muitas maneiras.

Cientistas da área geralmente aplicam alguns níveis para a descrição da leitura: acústica, fonética, fonologia, morfologia, sintaxe, semântica e o chamado pragmático ou discursivo (DUTOIT, 2007). Todos estes aspectos estão relacionados, embora dois outros elementos ocupem o centro das atenções quando pretendemos criar um TTS, que são: aquilo que será lido, ou seja, o **texto**, e como este se processa, sua **leitura**.

Assim sendo, a palavra Speech refere-se a processos associados à produção e à percepção dos sons utilizados na linguagem falada. Uma série de disciplinas acadêmicas estudo a fala e os sons, incluindo acústica, psicologia, patologia, lingüística, a ciência cognitiva e a informática.

3.1 Speech Synthesis

A síntese de voz há muito tempo é vital para as ferramentas de tecnologias assistivas e nessa área é significativa e difundida. Ela permite que barreiras do ambiente sejam removidas para as pessoas com uma grande faixa de limitações. As aplicações há mais tempo em uso são os leitores de tela, que são utilizados por pessoas com limitações visuais.

Hoje em dia, as Speech Synthesis ou TTSs são comumente utilizados por pessoas com dislexia e outras dificultadas de leitura bem como por crianças ainda não alfabetizadas.

São ainda empregadas como ajuda a pessoas com graves dificuldades de fala, usualmente dedicado a vocalização de voz.

A mais importante qualidade dos sistemas de síntese de voz é a **naturalidade** e a **inteligibilidade**. A naturalidade descreve o quão próximo o som da síntese é da voz humana, enquanto a inteligibilidade é o quanto o som é compreensível. O ideal dos sistemas TTS é que maximizem estas duas características.

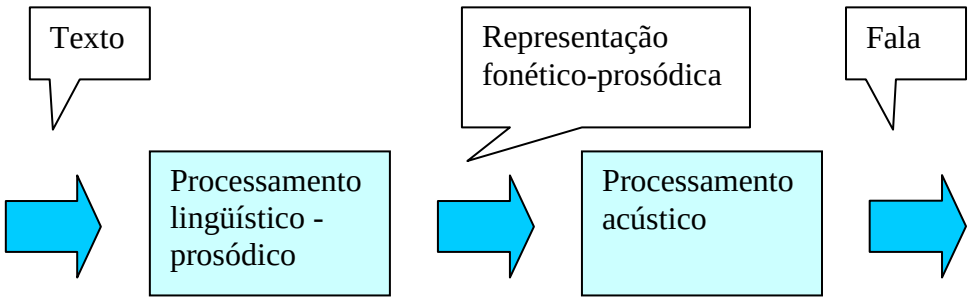


Figura 3.1 Tratamento do texto dentro do TTS

Na figura 3.1 temos a divisão do TTS em dois grandes blocos de processamento, o bloco de processamento linguistco-prosódico e o bloco de processamento acústico do sinal de fala. O primeiro tratando a forma como o texto será pronunciado e o segundo da forma como se dará esta pronúncia e como a pessoa ouvirá a mensagem já processada.

Speech Synthesis é a produção artificial da fala humana. Um sistema utilizado para este fim é chamado de sintetizador, e pode ser implementado em software ou hardware, podendo ser criados por concatenação de trechos de discursos gravados que estão armazenados em um banco de dados, por exemplo, ou simulando a produção de voz humana, como na formação sintética.

Para domínios específicos, o armazenamento de todas as palavras ou frases permite uma produção de alta qualidade, embora possa implicar em mais gastos com memória e certamente em tempo de resposta.

A qualidade de um Speech Synthesis é verificada por sua semelhança com a voz humana e por sua capacidade de entendimento. A partir dos anos 80 alguns sistemas operacionais já traziam bibliotecas para sintetização de voz, o que fazia com que seus softwares apresentassem mais qualidade. Além dos sistemas operacionais com essa característica, que veremos mais adiante, existem também muitos exemplos de aplicações que podem usar voz, dentre as quais:

- E-learning;

- Suporte para aplicativos móveis;

- Segurança;

- Call Center;

- Integração com PDAs;

- Reconhecimento de comandos de voz;

- Suporte a disléxicos;

- Operações cirúrgicas.

A Figura 3.2 mostra a arquitetura típica de um TTS.

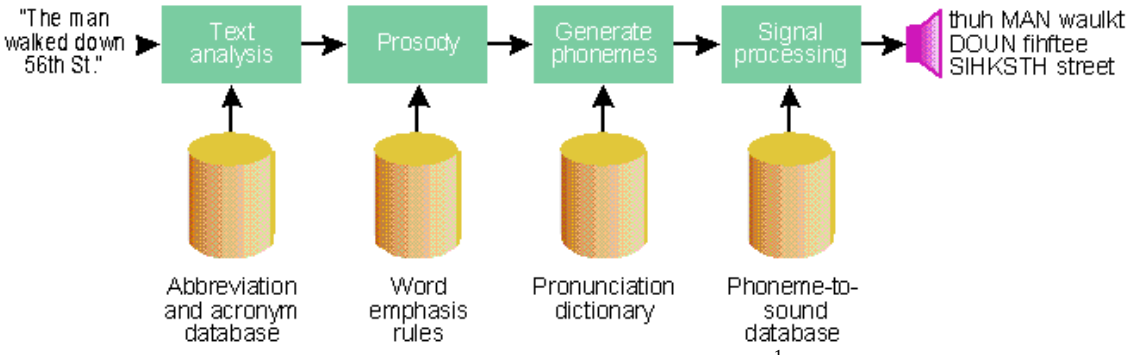


Figura 3.2 Text-to-Speech Engine [1]

O processo começa com uma seqüência de texto, o text analysis converte números em palavras, identifica pontuação como vírgula, dois pontos etc., converte abreviaturas em palavras, números e até mesmo a forma como pronunciar as siglas.

O trabalho do Text Analysis é bastante complexo, pois a linguagem escrita pode ser bastante ambígua. Um humano não teria problemas para pronunciar "St. John St." como "Saint John Street", mas um computador precisa ser treinado para tal.

Após a etapa de análise, temos a Prosody, onde, para tentar acoplar maior naturalidade ao texto, aspectos prosódicos são realçados, como a ênfase com que se deva ler determinada palavra.

Em seguida, o text-to-speech engine determina como as palavras são pronunciadas, quer procurando-as em um dicionário ou por algum algoritmo que supõe a pronúncia, cabendo aqui a referência ao contexto da palavra. O resultado desta análise no exemplo da figura 3.4 é a frase original expressa com seus fonemas correspondentes.

Por fim, são analisados os fonemas e suas pronúncias recuperadas a partir de um banco de dados. Na prática, cada fonema é ligeiramente modificado por seus vizinhos, de modo que a tabela tem muitas vezes 1.600 ou mais entradas. Utilizando técnicas de processamento de sinal, os sinais de áudio digital são enviados para um dispositivo de saída como um PC com placa de som e alto falantes.

[1] Figura extraída do site www.microsoft.com

As duas tecnologias primárias para gerar síntese de voz são: a concatenação sintética e a formação sintética. Cada uma dessas tecnologias possui pontos fortes e fracos e a utilização é que irá determinar qual a tecnologia mais apropriada.

3.1.1 Concatenação Sintética

A Concatenação Sintética é baseada na concatenação de segmentos de voz gravada. Geralmente, concatenando essas gravações, se produz uma vocalização mais natural. Entretanto, diferenças naturais entre a fala e as técnicas de concatenar segmentos de voz, resultam em áudio de baixa qualidade.

Existem três subtipos principais de concatenação sintética, que são:

- **Unidade de seleção de síntese**
  Grande BD onde são armazenadas diversas falas. Essas falas são separadas em fonemas, sílabas, morfemas, palavras, frases e sentenças.

- **Síntese de fonema**
  Gravação dos fonemas da linguagem. Sua complexidade está vinculada à língua. Exemplo: Espanhol – 800 fonemas, alemão – 2.500 fonemas.

- **Síntese para domínio específico**
  Vozes gravadas para aplicações específicas, como call centers, relógios, calculadoras. Possui grande uso há bastante tempo.

3.1.2 Formação Sintética

A formação sintética não utiliza a voz humana como exemplo para execução.
Ela utiliza modelos acústicos; alguns parâmetros para essa formação podem ser obtidos como a freqüência fundamental, voz e níveis de ruídos. Esses são alguns parâmetros utilizados para criar a forma da onda artificial da voz.

Muitos sistemas baseados em formação sintética geram artificialmente voz "robotizada" que nunca será confundida com a voz humana. Entretanto, o máximo de naturalidade nem sempre é o objetivo dos sistemas de síntese de voz. Além disso, os sistemas de formação sintética têm a vantagem sobre os de concatenação sintética no que se refere à reusabilidade, além de serem mais rápidos. Essa rápida síntese de voz é utilizada, por exemplo, em computadores na navegação utilizando leitores de tela.

Sintetizadores de voz por formação, usualmente são programas menores do que os programas por concatenação, pois não necessitam de banco de dados com exemplos de fala. Eles podem ainda ser utilizados em sistemas embarcados onde memória e processamento são limitados. Devido aos sistemas baseados em formação sintética terem o completo controle de

todos os aspectos da fala, uma grande variedade de prosódia e entonação pode ser criada, convertendo não apenas sentenças, mas uma variedade de emoções e tons da voz.

3.1.3 Síntese Articulatória

A síntese articulatória é baseada nos modelos de trato vocal e no processo de articulação envolvido. A forma do trato vocal pode ser controlada pelo número de formas das articulações, como: língua, mandíbula e lábios. A fala é criada digitalmente pela simulação do fluxo de ar através dessa representação do trato vocal.

3.2 Speech Recognition

O reconhecimento de fala é o processo pelo qual um computador (ou outro tipo de máquina) identifica palavras faladas. Basicamente, significa falar com a máquina e ela reconhecer corretamente aquilo que está sendo dito. As seguintes definições são as informações básicas necessárias para se entender tecnologias de reconhecimento de fala (COOK, 2002).

Os sistemas de reconhecimento de fala (ASR) podem ser separados em diferentes classes, descrevendo os tipos de afirmações que têm a habilidade de reconhecer. Essas classes são baseadas no fato de que uma das dificuldades do ASR é a capacidade de determinar quando um orador começa e termina uma dicção. A maioria dos pacotes não se encaixa em mais do que uma classe, dependendo do modo como eles estão sendo usados.

Em (COOK, 2002) temos identificados cinco tipos de reconhecedores de voz:

- Palavras isoladas

Reconhecedores de palavras isoladas normalmente exigem de cada elocução uma parada (falta de um sinal de áudio). Isso não significa que ele só aceita palavras, mas que exige uma dicção em uma única vez. Frequentemente, estes sistemas têm a chamada "Escuta / Não Escuta", em que é requerido que o orador tenha que esperar entre as pausas (normalmente fazendo tratamento durante estas).

- Palavras conectadas

Sistemas de palavras conectadas são semelhantes às palavras isoladas, mas permitem separar afirmações para serem rodadas em conjunto, com uma pausa mínima entre elas.

- Discurso contínuo

Reconhecedores com capacidades de fala contínua são alguns dos tipos de ASR mais difíceis de implementar, uma vez que devem se utilizar de métodos especiais para determinar fronteiras de dicção. Reconhecedores de fala contínua permitem que os usuários falem quase que naturalmente, enquanto o computador determina o conteúdo.

Basicamente, é um ditado feito para o computador.

- Fala espontânea

Parece haver uma variedade de definições para o que realmente se entende por fala espontânea. Em um nível básico, ela pode ser pensada como um discurso que ocorre naturalmente. Um sistema com ASR de fala espontânea deverá ter a capacidade de lidar com uma variedade de recursos naturais como palavras sendo executadas em conjunto, tais como "ums" e "ahs", e mesmo ligeiras engasgadas por parte do locutor.

- Verificação / identificação de voz

Alguns sistemas de ASR têm a capacidade para identificar usuários em particular, visando assim sistemas mais específicos que tenham em vista a verificação de pessoas como sistemas de segurança, por exemplo.

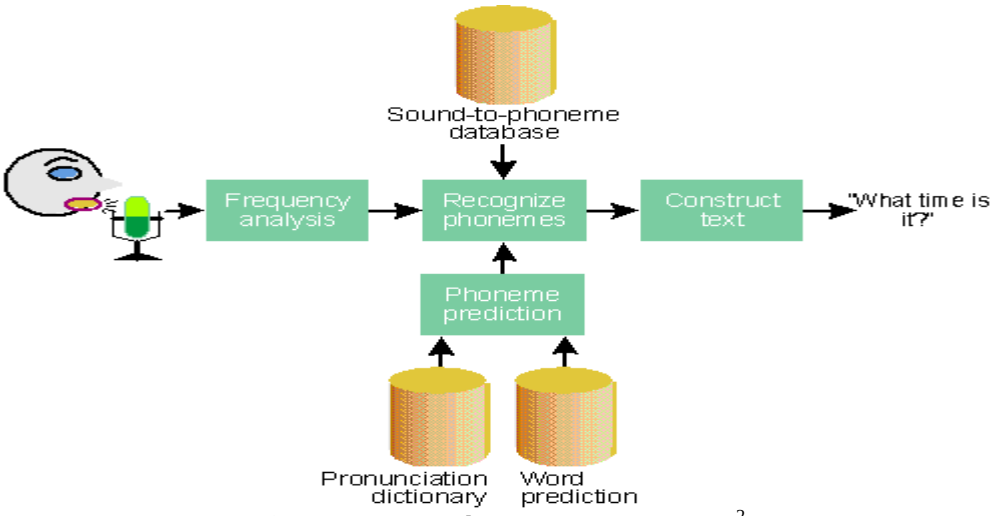


Figura 3.3 Speech Recognition Engine [2]

A figura 3.3 apresenta o funcionamento da speech recognition.

Nesta, vemos que o áudio é quebrado em fonemas por um módulo de reconhecimento de fonemas. Este módulo procura um som para o fonema na base de dados. Cada entrada de dados contém um modelo que descreve o que soa como um fonema. Tal como acontece com o text-to-speech, a tabela normalmente tem alguns milhares de entradas.

Comparar os dados de áudio contra os milhares de fonemas da base de dados levaria um longo tempo, existe para isso um mecanismo de reconhecimento de fala que contém um

[2] Figura extraída do site www.microsoft.com

módulo de previsão que reduz o número de candidatos por predizer os fonemas que são prováveis de ocorrer em um contexto particular. Por exemplo, alguns fonemas raramente ocorrem no início de uma palavra, como os "ft" no final da palavra "raft". Outros fonemas nunca ocorrem aos pares. Mas mesmo com estas otimizações, o reconhecimento de fala ainda fica demasiadamente moroso.

Uma base de dados World Prediction é usada para reduzir ainda mais os fonemas candidatos da lista, eliminando fonemas que não produzem palavras válidas. Após a audiência, o "y eh", o reconhecedor irá ouvir por "s" e "n", uma vez que "yes" e "yen" são palavras válidas. Também irá ouvir por "m" no caso da pronúncia de "Iêmen", por exemplo.

Se a aplicação só quer saber se o usuário disse "yes" ou "no", o fonema reconhecedor não precisa ouvir por "n" após o "y eh," apesar de "yen" ser uma palavra.

Depois que os fonemas são reconhecidos, eles são analisados em palavras, convertidos em strings de texto e passados para a aplicação.

3.3 Desafios da Normalização de Textos

A normalização de textos em um TTS é o processo pelo qual o texto é tratado inicialmente, assim que o mesmo é digitado ou captado em algum documento para ser preparado para vocalização, ela inicia-se com a etapa de Text Analysis, onde se analisam todos os integrantes daquele texto escrito (expressões, siglas, nomes próprios, etc.).

O processo de normalização de texto raramente é direto. Os textos normalmente possuem diversos heterônimos, números e abreviações e todos eles requerem expansão em uma representação fonética. Em português, há muitas palavras que são pronunciadas de diferentes modos dependendo do contexto. Por exemplo: "Ele usou uma colher para colher a horta". Esta frase contém duas pronúncias diferentes para "colher" ou ainda
"Eu jogo o jogo".

A maioria dos sistemas TTS não gera representações semânticas para os textos de entrada resultando em uma fala não confiável e nem compreensível. Diferentes técnicas heurísticas são utilizadas para tentar resolver esse problema e desfazer a ambigüidade homográfica, como por exemplo, efetuando estatísticas de ocorrência de palavras próximas.

Decidir como converter números é outro problema de sistemas TTS. Um exemplo é converter um número como "1234". Ele pode ser convertido em "mil duzentos e trinta e quatro" se for um ano ou valor, como também pode ser traduzido em "doze trinta e quatro" se for matrícula ou ainda em "um dois três quarto" se for um número de cartão de crédito. Esse problema tem uma solução parecida com as palavras homográficas que é a análise das palavras próximas.

3.3.1 Desafios de avaliação

É muito difícil avaliar sistemas de voz consistentemente, pois diferentes sistemas geralmente utilizam dados de voz diferentes. A qualidade da síntese de voz depende principalmente da qualidade da gravação da voz existente. Como conseqüência, avaliar a síntese de voz é o mesmo que avaliar a qualidade da gravação.

Recentemente pesquisadores começaram a avalizar sistemas de voz baseados em um dataset comum. Com isso, pode-se avaliar os sistemas focando nas diferenças de tecnologia e não nas diferenças de gravações.

3.4 Histórico

Embora a partir da década de oitenta a vocalização ficou em destaque com sua utilização por alguns sistemas operacionais, seu emprego teve início no fim dos anos trinta. Hoje temos vários exemplos de softwarwe que desempenham esta função.

3.4.1 VODER

Na história da vocalização, temos o primeiro sintetizador vocal totalmente funcional, o Homer Dudley's VODER (Voice Operating Demonstrador), sendo demonstrado em 1939 no World's Fair. O VODER foi baseado no Vocoder (Voice Operated Recorder), ambos foram desenvolvidos por Homer Dudley, numa investigação nos Laboratórios da Bell, em New Jersey; foi constituído de um dispositivo composto de um analisador e uma voz artificial.

O analisador detectava níveis de energia sonoras sucessivas nas amostras medidas ao longo de todo o espectro de áudio com freqüência através de uma série de filtros. Os resultados poderiam ser visualizados graficamente.

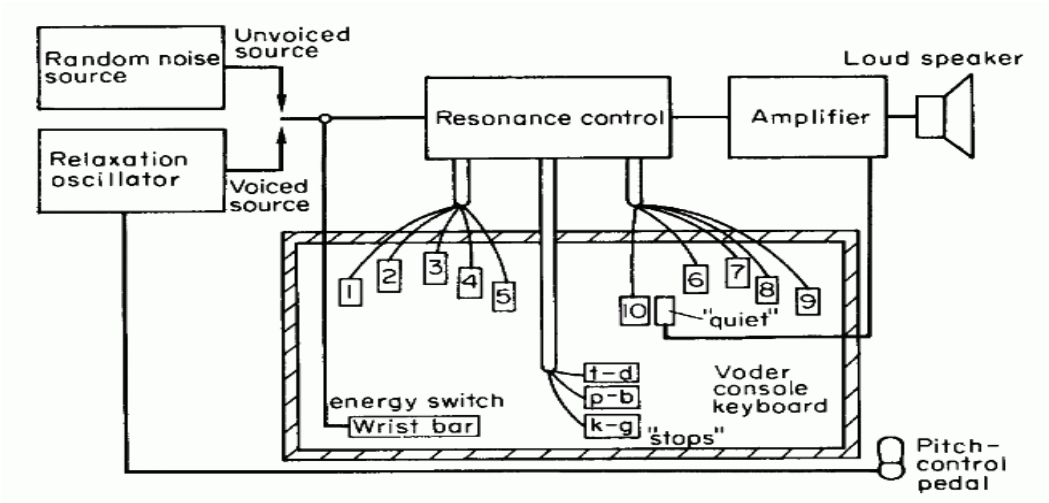


Figura 3.4 Diagrama do VODER [3]

A figura 3.4 apresenta o VODER com sua vocalização da fonte e possíveis ruídos aaleatórios sendo captados, tratados e amplificados na saída por um alto falante.

3.4.2 FESTIVAL

O Festival é um sistema de síntese de voz desenvolvido pelo Speech Technology Research (CSTR) da Universidade de Edimburgo. Disponibiliza um sistema de síntese completo com várias APIs, assim como um ambiente para desenvolvimento e pesquisa de técnicas de síntese de voz.

Uma descrição do Festival encontra-se em (BLACK, 1999), tendo sua criação creditada a Alan W Black, Richard Caley e Paul Taylor.

O Festival oferece um framework para a construção de sistemas de speech synthesis, possui uma API própria e um conjunto de bibliotecas escritas em C++ e atualmente suporta os idiomas inglês, espanhol e galês, tendo o inglês como sua implementação mais avançada, além de possuir um interpretador de comando para controle (BLACK, 1999).

Figura 3.5 Página da versão on-line do Festival

Na página do Festival temos opções para escolha de voz (masculina ou feminina) além da nacionalidade, restringindo-se a inglês da Grã-Bretanha ou dos EUA e um campo de texto limitado para digitação. Clicando ao lado deste podemos salvar a leitura do texto em formato wav para ser ouvida por um software de áudio.

[3] Figura extraída do site www.stanford.edu

4 Fonética e Fonologia

Sabendo que o TTS envolve uma área de conhecimento como a prosódia e os fonemas das palavras, apresentamos um estudo breve sobre alguns termos relevantes e que denotam conhecimento quando se pretende tratar de reconhecimento vocal, embora este não seja nosso objetivo inicial.

A língua portuguesa é uma língua neolatina, formada da mistura latim vulgar mais a influência árabe e das tribos que viviam na região. Sua origem está altamente conectada a outra língua (o galego), mas o português é uma língua própria e independente (BRANCO,2008).

Apesar da influência dos tempos tê-la alterado, adicionando vocábulos franceses, ingleses, espanhóis, ela ainda tem sua identidade única, sem a força que tinha no seu ápice, quando era quase tão difundida como agora é o inglês (BRANCO, 2008).

Estudar a fonologia que ocorre na língua portuguesa para uma correta compreensão dos sinais sonoros nos leva de encontro a duas formas muito similares para averiguar a qualidade daquilo que se fala e ouve: a prosódia e a ortoépia.

A **prosódia**, segundo o Aurélio, é "a variação na altura, intensidade, tom, duração e ritmo da fala", sendo então relacionada com a correta acentuação e entonação dos fonemas.

A **ortoépia**, segundo o mesmo dicionário, é "a pronúncia normal e correta", cuidando da correta articulação e pronúncia dos grupos fônicos.

Temos um dos principais entraves no desenvolvimento de qualquer sistema que pretenda trabalhar com a transformação de texto em som, a correta passagem de um para outro, guardando seu contexto de uso, variações de humor, de tempo, variações regionais (sotaque) e outras formas de expressão verbal que caracterizam nossa espécie e nossa rica e às vezes sorrateira linguagem.

4.1 Fonoestilística

A Fonoestilística segundo (ANDRADE, 2007) trata dos valores expressivos de natureza sonora observáveis nas palavras e nos enunciados. Fonemas e prosodemas (acento, entonação, altura e ritmo) constituem um complexo sonoro de extraordinária importância na função emotiva e poética.

Além de permitir a oposição de duas palavras (função distintiva), a matéria fônica desempenha uma função expressiva que se deve a particularidades da articulação dos fonemas, às suas qualidades de timbre, altura, duração, intensidade. Os sons da língua podem provocar-nos uma sensação de agrado ou desagrado e ainda sugerir idéias, impressões.

Existem (ANDRADE, 2007) encontros de fonemas que são mais relevantes em textos escritos, produzindo efeitos desagradáveis para quem ouve, devido a pausas e entonações que podem dificultar o processo de entendimento do ouvinte, abaixo seguem alguns exemplos.

- **Cacofonia**: ocorre quando as sílabas finais de uma palavra se encontram com as sílabas iniciais de outra palavra, produzindo um efeito diferente e que nubla o perfeito entendimento fazendo com que as palavras ganhem outro sentido; segundo o Aurélio "é um encontro ou repetição de sons que desagrada ao ouvido".
  Ex: Herói da nação.

- **Hiatos**: seqüência de vogais pertencendo a sílabas diferentes, que produz um efeito confuso para os ouvintes.
  Ex: ...e aí o outro é...

- **Ecos**: representação de sentenças com finais idênticos e em pequenos intervalos.
  Ex: ... tem um outro aqui...eu tenho...deixe eu ver aqui... ... ...eu tenho dois aqui....tenho do que você me deu agora... né? ahn:: ahn::e tenho...deixa eu ver se tá no outro aqui...

4.2 Alterações observadas nas falas

Podem ocorrer alterações nas falas das pessoas por dois motivos: acrescentamento e supressão.

Por Acrescentamento:

- **Prótese**: acréscimo de algum fonema no início da palavra. Adjunção de segmento fonético no início de uma palavra, sem que se lhe altere o sentido.
  Ex: ruído por arruído; lagoa por alagoa.

- **Epêntese**: desenvolvimento de fonema(s) no meio de uma palavra.
  Ex: barata (anima*l*) por brata; advogado por adevogado.

- **Paragoge**: adição de letra ou sílaba no final de uma palavra.
  Ex: ante por antes; quite por quites.

Por Supressão:

- **Aférese**: eliminação de fonema ou grupo de fonemas no início da palavra.
  Ex: José por Zé.

- **Síncope**: eliminação de fonema ou sílaba no interior da palavra.
  Ex: maior por mor.

- **Apócope**: eliminação de fonema ou sílaba no final da palavra.
  Ex: Santo por São; muito por mui.

- **Metátese**: transposição de fonemas dentro de um mesmo vocábulo; hipértese, comutação.
  Ex: semper por sempre; desvariar por desvairar.

4.3 Prosódia

Seu objetivo é a identificação correta da sílaba tônica. Existe uma diferença significativa entre a língua culta e a língua utilizada no dia-a-dia, sabendo que a prosódia preocupa-se em adequar-se à língua culta.

Na língua portuguesa temos a vogal como elemento fundamental da sílaba, a falta de acentuação de muitas palavras pode induzir as pessoas ao erro e fazer com que as palavras troquem de classe, como em cateter (oxítona) e catéter (paroxítona), um erro de prosódia. A prosódia estuda e fixa a posição da sílaba tônica, como em rubrica (paroxítona) ao invés de rúbrica (proparoxítona), ocorrendo o que se denominou de
silabada (troca da sílaba tônica).

4.4 Ortoépia

Origina-se do grego orthós (reto, direito) mais épos (palavra), cuida da correta articulação e pronúncia dos grupos fônicos.

Interessante notar aqui é que quando um erro de ortoépia, este pode passar a fazer parte da linguagem de um grupo ou população, altera-se a grafia para poder incluí-lo, originando desta forma uma nova língua falada ou dialeto.

Erros ortoépicos são considerados erros de linguagem assim como os erros ortográficos, sendo que estes aparecem na escrita, e os primeiros têm a escrita correta, mas pecam na pronúncia.

Erros de ortoépia denominam-se cacoépia, e como a linguagem falada é mais dominante que a escrita, os erros de cacoépia formam um oceano de expressões falhas.

Uma aproximação da prosódia com a ortoépia ocorre nas salas de bate-papo, onde a

linguagem escrita tenta assimilar a fluição da linguagem falada, obviamente com resultados, em vários casos, desastrosos.

Itens que ocorrem na ortoépia:

- Perfeita emissão das vogais e grupos vocálicos.
  Ex: leite e não leiti; queijo e não quêjo;

- Articulação correta dos fonemas consonantais.
  Ex: mulher e não mulhé; companhia e não compania;

- Adequada e correta ligação das palavras nas frases.
  Ex: A mulher estava sozinha, porém, havia um vulto a espreitar o leite que ela derramava.

5 SAPI

Antes de entrarmos na seção a cerca da API SAPI, iniciamos com alguns sistemas operacionais que se destacam pela utilização de sintetizadores de voz.

5.1 Apple

O primeiro sistema de síntese de voz integrado a um sistema operacional foi o Mac in Talk em 1984. No início da década de 90, a Apple expandiu sua capacidade oferecendo um grande suporte a sistemas de TTS. Com o início dos computadores baseados no Power PC, incluíram exemplos de voz de grande qualidade. A Apple também se utilizou do reconhecimento de voz em seus sistemas para a execução de comandos.

5.2 Amiga OS

O segundo sistema operacional com possibilidade de síntese de voz foi o Amiga OS em 1985. Seu sistema incluía a diferenciação entre vozes masculinas e femininas além de um indicador do nível de estresse possibilitados pelos avanços feitos no hardware do Amiga OS e no chipset de áudio.

5.3 Microsoft Windows

O Windows utiliza a **SAPI** (Speech Application Programming Interface) e o **SRE** (Speech Recognition Engine).

O Windows XP disponibilizou um programa chamado "Narrator" que utilizada a SAPI, a figura 5.1 apresenta a sua interface com a voz a ser utilizada, velocidade, volume e taxa.

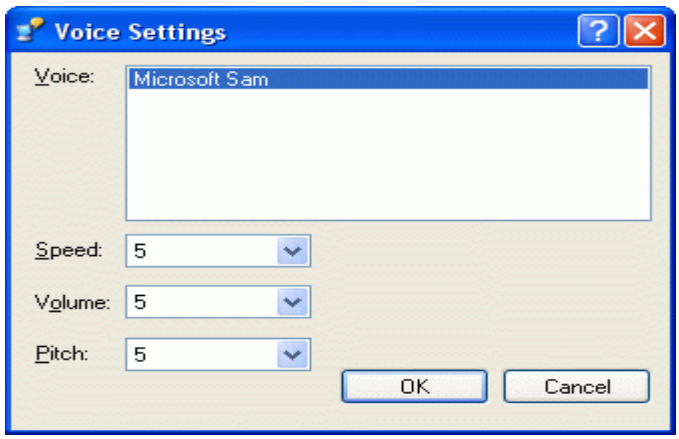


Figura 5.1 Configuração do Narrator no Windows XP

Todos os programas compatíveis com o Windows podem fazer o uso da SAPI para desenvolvimento de aplicações TTS. A Microsoft possui ainda o Microsoft Speech Server que é um completo pacote de síntese e reconhecimento de voz para aplicações comerciais como Call Centers.

O Windows Vista inclui diversas inovações relacionadas à Síntese e Reconhecimento de voz:

- Nova versão do Speech API: SAPI 5.3;
- Atualização do SRE (versão 8);
- Novo motor de voz para síntese e a nova voz chamada "Microsoft Anna";
- O reconhecedor de voz suporta oito linguagens até o presente momento. São elas: inglês dos Estados Unidos, inglês do Reino Unido, chinês tradicional, chinês simplificado, japonês, alemão, francês e espanhol.

5.3.1 SAPI

A Speech Application Programming Interface ou **SAPI** é uma API desenvolvida pela Microsoft para permitir o uso do reconhecimento e síntese de voz. Algumas aplicações que utilizam a SAPI são o Microsoft Office, Microsoft Agent e o Microsoft Speech Server. Além disso, as empresas podem desenvolver sua própria biblioteca de reconhecimento e síntese de voz, e integrá-la à SAPI.

A SAPI é um componente de livre distribuição e pode ser embarcada em qualquer aplicação.

A API de Voz pode ser vista como uma interface, ou parte de um middleware que está entre as aplicações e os motores de voz (reconhecimento e síntese). Entre as versões 1 e 4 da SAPI, as aplicações podiam se comunicar diretamente com os motores.

A SAPI inclui a abstração da **Interface de Definição** onde as aplicações e os motores estão em conformidade. As aplicações podem ainda fazer uso de objetos de alto nível simplificados antes de fazerem acesso direto aos métodos dos motores.

Estes motores , tanto de síntese como de reconhecimento, são os responsáveis por todo o trabalho referente à criação da voz e/ou reconhecimento. Os motores são o núcleo do TTS e da SR. No caso da TTS eles se encarregam de captar o texto lido e faze-lo percorrer as etapas dentro do Speech Synthesis e devolvê-lo na forma de onda sonora.

Na SAPI 5, entretanto, aplicações e motores não tem comunicação direta entre si. Ao invés disso, eles conversam em tempo de execução com um componente (sapi.dll). Esse componente é a implementação da API, que a aplicação fará uso e a interface para o motor.

Tipicamente, aplicações SAPI 5 fazem chamadas através da API (por exemplo, para carregar uma gramática de reconhecimento, iniciar um reconhecimento ou indicar um texto para síntese). O componente **sapi.dll** em tempo de execução interpreta esses comandos e os processa, quando necessário chama o motor através da **Interface do Motor** (por exemplo, o carregamento de uma gramática para um arquivo é feito em tempo de execução, porém quando os dados da gramática são passados para o motor de reconhecimento então ele usa o reconhecedor). Os motores de reconhecimento e de síntese também geram eventos enquanto são processados (por exemplo, para indicar a entonação reconhecida ou para indicar palavras limites na síntese de voz).

A API atualmente está na versão 5, constituindo uma nova família.

A primeira versão da SAPI foi lançada em 1995 e foi utilizada no Windows 95 e no Windows NT 3.51. Essa versão inclui um baixo nível de reconhecimento direto de voz, trazia uma versão simplificada de alto nível da API para comando de voz e voz falada.

A versão 2 da SAPI foi lançada em 1996 e não trouxe grandes mudanças.

A SAPI 3 aumentou o limite suportado para reconhecimento de voz ditada (voz discreta e não continua*)* e aumentou os exemplos de fontes de áudio para as aplicações.

5.3.1.1 SAPI 4

A SAPI 4 foi lançada em 1998. Esta versão inclui na API, suporte a COM, um pacote de classes C++ para facilitar a programação e controles ActiveX para permitir o desenvolvimento de aplicações em Visual Basic. Ele foi embarcado como parte de um SDK que inclui o motor de reconhecimento e síntese de voz. Ele também foi embarcado no Windows 2000.

Os principais componentes da API SAPI 4 (que também são disponíveis em C++, COM e ActiveX) são:

- Comando de Voz: objetos de alto nível para comando e controle por reconhecimento de voz;
- Ditado de Voz: objetos de alto nível para reconhecimento de voz ditada continuamente;
- Voz Falada: objetos de alto nível para síntese de voz ;
- Voz de Telefonia: objetos de alto nível para aplicações de fala em telefones;
- Reconhecimento Direto de Voz: objetos para controle direto do motor de reconhecimento;
- Texto Para Voz Direto: objetos para controle direto do motor de síntese de voz;
- Objetos de Áudio: para leitura e escrita em dispositivos ou arquivos.

5.3.1.2 SAPI 5

A Speech SDK Versão 5.0 incorporava a SAPI 5.0 lançada em 2000. Ela foi completamente reformulada em relação às versões anteriores e nenhum controle nem aplicações usadas nas versões anteriores podem ser usadas nesta nova sem consideráveis modificações.

O formato da nova API inclui o conceito de real separação entre aplicação e o motor, assim todas as chamadas são direcionadas através da sapi.dll em tempo de execução. Esta mudança foi feita para fazer a API mais independente do motor, prevenindo as aplicações de dependência inadvertida de modificações feitas em um motor específico. Além disso, essa modificação foi pensada para facilitar a incorporação da tecnologia de voz dentro de aplicações, através de alguns códigos de inicialização na aplicação.

A nova API foi feita inicialmente pura em API COM e pode ser usada facilmente em C/C++. Suporte para Visual Basic e linguagens de script foram incluídos posteriormente. Sistemas operacionais Windows acima do Windows 98 e NT 4.0 a suportam. As maiores inovações que a API inclui são:

- Shared Recognizer: permite o reconhecimento de voz para aplicações no desktop;
- In-proc recognizer: utilizado por aplicações que precisam explicitamente do controle do motor de reconhecimento de voz;
- Grammar objects: utilizado para especificar as palavras que o reconhecedor irá ouvir;

- Voice object: converte para áudio um arquivo de texto. Uma linguagem de marcação (similar ao XML, mas não estritamente XML) pode ser utilizada para controlar o processo de síntese;

- Audio interfaces: interface para controlar os dispositivos de áudio, tanto o microfone quanto os alto-falantes;

- User lexicon object: permite que a customização de palavras e pronúncias sejam adicionadas pelo próprio usuário. Essas palavras serão inseridas no reconhecimento de voz e na síntese como objetos léxicos;

- Object tokens: este conceito permite que reconhecedores, ferramentas TTS, objetos de áudio, léxicos e outras categorias de objetos, sejam registrados, enumerados e instanciados de uma forma comum.

Sapi 5.0

Versão embarcada do final de 2000 como parte da Speech SDK versão 5.0, junta com a versão 5.0 do reconhecedor de voz e o motor de síntese. O motor do reconhecedor de voz suporta ditado contínuo; foram lançados em inglês dos Estados Unidos, japonês e chinês simplificado. No sistema em inglês, são disponibilizados modelos acústicos especiais com voz de criança ao telefone.

O motor de síntese foi disponibilizado em inglês e chinês. A versão da API com motor de reconhecimento foi embarcada no Microsoft Office XP em 2001.

Sapi 5.1

Essa versão da API e do motor TTS foi embarcado no Windows XP.

Sapi 5.2

Essa foi uma versão especial da API, apenas para usuários do Microsoft Speech Server, em 2004. Ela inclui suporte para linguagens de marcação como SRGS e SSML bem como melhorias adicionais na performance. O Speech Server foi ainda embarcado como a versão 6 do cliente do reconhecedor e com a versão 7 do servidor do motor de reconhecimento.

Sapi 5.3

Versão da API utilizada no Windows Vista juntamente com o novo reconhecedor e motor de reconhecimento. Nomeado como Windows Speech Recognition, é agora integrado ao sistema operacional. A Speech SDK e as APIs são parte do Windows SDK.

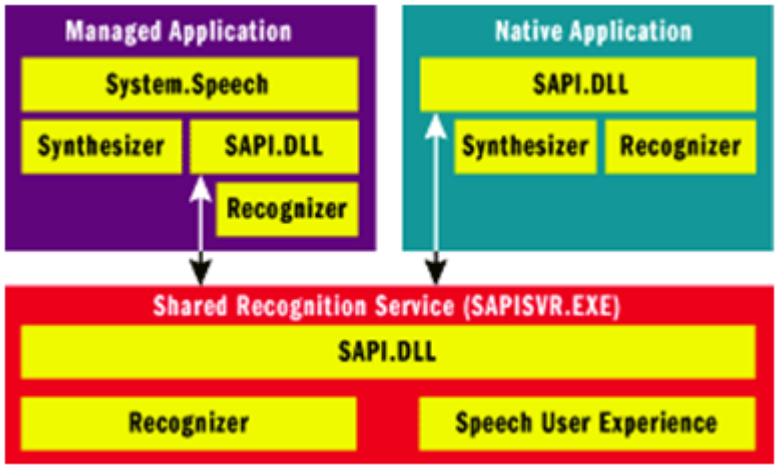


Figura 5.2 SAPI 5.3 [4]

Na figura 5.2 temos a versão 2 da familia 5 da SAPI utilizada no projeto, esta versão permite o uso da SSML a exemplo da versão anterior e apresenta a separação entre os chamandos motores de síntese e de reconhecimento das aplicações, ganhando desta forma em performance. A sapi.dll é a interface utilizada pelo motor na comunicação com as aplicações.

A SAPI permite a utilização das APIs de speech tanto para síntese como para reconhecimento nas duas formas de aplicações, seja nativamente ou não.

[4] Figura extraída do site http://msdn.microsoft.com

6 SSML

Uma grande quantidade de linguagens de marcação foram desenvolvidas baseadas em XML para auxiliar no desenvolvimento dos sistemas de síntese de voz. A mais recente dessas linguagens, é a **SSML** (Speech Synthesis Markup Language - Linguagem de Marcação de Síntese de Voz), que surgiu como recomendação da W3C em 2004, é, portanto, uma norma da W3C para formatar o texto de forma que ele possa ser sintetizado.

Existem outras linguagens de marcação como, por exemplo, a JSML (Java Speech Markup Language - Linguagem de Marcação Java de Voz) e a SABLE. Apesar de cada uma dessas se propor a ser o padrão, nenhum delas foi largamente adotada.

Linguagens de marcação de síntese de voz são diferentes das linguagens de marcação de diálogo, antes de iniciarmos as ponderações sobre o SSML, frisamos duas tecnologias muito úteis no trato com sistemas de reconhecimento de voz, o Voice XML e a SRGS.

6.1 Voice XML

Segundo (WEINSTEIN, 2001) é uma linguagem para descrever interações homem-máquina utilizando reconhecimento de fala, linguagem natural e de síntese vocal. VoiceXML foi desenvolvida por um fórum constituído de várias empresas interessadas em texto falado (tais como AT&T, IBM, Lucent e Motorola) e mais tarde foi padronizada pelo World Wide Web Consortium.

VoiceXML visa proporcionar um meio de expressão racionalizada no desenvolvimento de aplicativos. No entanto, a filosofia da VoiceXML difere da de Speech Builder (construção de discurso) em várias áreas-chave (WEINSTEIN, 2001).

A VoiceXML inclui tags relacionadas ao reconhecimento de voz, gerenciamento de voz e intensidade do diálogo em adição às linguagens de marcação de TTS.

A navegação por voz possibilita que um usuário possa navegar em uma página, desenvolvida na linguagem VoiceXML especificamente para este tipo de navegação.

São disponibilizados menus de opções que o usuário pode escolher e de acordo com a opção desejada será oferecido um submenu ou as informações desejadas. Tudo é feito mediante síntese e reconhecimento de voz pelo sistema.

A Voice XML foi desenvolvida para a criação de diálogos de áudio que caracterizam os sintetizadores de voz, áudio digital, reconhecimento do orador, telefonia, etc. Contudo o seu principal objetivo é aproveitar as capacidades das aplicações baseadas na Web e entrega de

conteúdos para a criação de aplicações de voz interativas, sendo seu maior mercado atualmente nas Centrais de Atendimento.

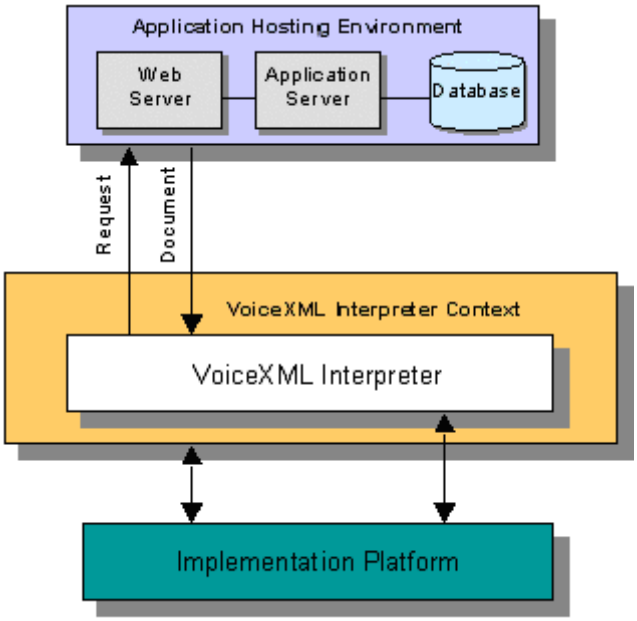


Figura 6.1 Voice XML [5]

A figura 6.1 mostra umas das vantagens de se utilizar a Voice XML, onde, em um call center, por exemplo, ao invés do cliente ficar escutando todas as mensagens de texto gravado, este fala qual é a sua opção e a VoiceXML Interpreter se encarrega de buscar a resposta no banco de dados, tornando o processo mais interativo e mais rápido.

6.2 SRGS

Por outro lado, as definições de regras de gramática foram padronizadas pelo W3C com o nome de **SRGS** (Speech Recognition Grammar Specification).

Combinações de regras de gramática permitem haver reconhecimento de voz em um mesmo diálogo (WEINSTEIN, 2001).

Uma gramática de reconhecimento de voz é um conjunto padrão de palavras e diz a um sistema de reconhecimento de fala o que se pode esperar que um ser humano fale, padronizando assim as possíveis respostas.

[5] Figura extraída do site www.voicexmlreview.org

Trata-se de outra recomendação, fundamental para o apoio da Voice XML para reconhecimento de fala, é usada pelos desenvolvedores para descrever os utilizadores finais dando respostas às solicitações faladas. Ele define uma sintaxe para representar gramáticas para uso no reconhecimento da fala, para que os desenvolvedores possam especificar as palavras e os padrões de palavras para serem ouvidos por um reconhecedor.

A sintaxe da gramática é apresentada em duas formas, uma ABNF (também conhecida como formalismo de Backus-Naur, forma normal de Backus, ou ainda forma de Panini-Backus, ou seja, uma notação para as gramáticas de linguagens de programação, conjuntos de instruções e protocolos de comunicação, e também como notação para representar partes de gramáticas de linguagens naturais e um formulário XML.

A especificação torna as duas representações mapeáveis para permitir transformações automáticas entre ambas.

A listagem 6.1 apresenta as duas formas de sintaxe.

Listagem 6.1: ABNF e XML

ABNF:

```
#ABNF 1.0 ISO-8859-1;

mode dtmf;

$digit = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9;
public $pin = $digit <4> "#" | "*" 9;
```

XML:

```
<?xml version="1.0"?>

<grammar mode="dtmf" version="1.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-
grammar/grammar.xsd"
         xmlns="http://www.w3.org/2001/06/grammar">

<rule id="digit">
 <one-of>
   <item> 0 </item>
   <item> 1 </item>
   <item> 2 </item>
   <item> 3 </item>
   <item> 4 </item>
```

```
    <item> 5 </item>
    <item> 6 </item>
    <item> 7 </item>
    <item> 8 </item>
    <item> 9 </item>
  </one-of>
</rule>

<rule id="pin" scope="public">
  <one-of>
    <item>
      <item repeat="4"><ruleref uri="#digit"/></item>
      #
    </item>
    <item>
      * 9
    </item>
  </one-of>
</rule>

</grammar>
```

Neste exemplo temos uma gramática simples onde uma variável pin recebendo quatro dígitos mais um cerquilha “#” ou também a seqüência de “*” seguido de “9” (por exemplo, ao receber uma mensagem de ajuda).

Sendo assim, a SRGS define uma sintaxe para a representação de gramáticas com o reconhecimento de voz, de modo que os programadores possam especificar palavras e padrões de palavras que possam ser “ouvidas” por um reconhecedor de voz.

6.3 A SSML

A SSML é uma Linguagem de Marcação para Síntese de Voz, como já está subentendido em seu nome, uma linguagem de marcação, baseada no XML, porém aplicada a sistemas de síntese de voz, é também uma recomendação do grupo W3C.

A SSML é frequentemente embarcado em scripts Voice XML utilizados para os sistemas interativos de telefonia. Ainda assim, pode ser utilizado de forma individual, é baseado no JSML (Java Speech Markup Language), linguagem desenvolvida pela Sun.

Conforme (SILVA, 2001), a especificação da Speech Synthesis Markup Language é uma parte importante deste novo conjunto de regras para os browsers de voz e foi desenhada para fornecer uma linguagem de marcação mais rica, baseada em XML (eXtended Markup Language), para que seja possível a criação de aplicações com este novo tipo de Interface Web.

A regra essencial para a elaboração desta linguagem de marcação é estabelecer mecanismos que permitam aos autores de conteúdos “sintetizáveis” controlarem aspectos do diálogo como a pronúncia, o volume, a ênfase de determinadas palavras ou frases nas diferentes plataformas.

Ela oferece marcas para controlar as vozes, a taxa de voz, volume, sexo e tom.

Oferece ainda marcações para controlar a forma como as palavras são sintetizadas, como por exemplo, para soletrar as abreviaturas.

Exemplo de código SSML:

Listagem 6.2: Exemplo de SSML

```
<?xml version="1.0"?>
<speak
xmlns="http://www.w3.org/2001/10/synthesis"xmlns:dc="http://purl.org/dc/elements/1.1/
version="1.0">
<metadata>
        <dc:title xml:lang="en">Telephone Menu: Level 1</dc:title>
</metadata>
<p>
         <s xml:lang="en-US">
                <voice name="David" gender="male" age="25">
                        For English, press <emphasis>one</emphasis>.
                </voice>
        </s>
         <s xml:lang="es-MX">
                <voice name="Miguel" gender="male" age="25">
                        Para español, oprima el <emphasis>dos</emphasis>.
                </voice>
        </s>
</p>
</speak>
```

A tag <speak> é o elemento raiz de um documento SSML. Pode-se notar que a versão do SSML é dada como um valor no atributo "version" do elemento <speak>.

No nosso exemplo acima, temos um elemento <p> com duas sentenças dadas em <s>. Um elemento é usado para ênfase de stress para uma palavra que é falada.

Esta ênfase é que irá ressaltar uma diferenciação na leitura.

Pode haver casos em que desejamos usar uma voz feminina, nesse cenário utiliza-se o elemento <voice>, que tem um atributo sexo (gender). Define-se o atributo

de gênero <gender> = "female", por exemplo.

A voz tem um elemento "age" que pode ser usado para referir a idade do gênero que fala o texto. Por exemplo, para especificar <voice gender="female" age="6"> o texto delimitado por esse elemento será uma voz falada por uma criança do sexo feminino cuja idade é 6 anos.

A SSML é comumente utilizado em aplicações interativas de telefonia, especialmente on-line. Este tipo de linguagem permite uma excelente clareza no som, garantindo que o sinal de áudio seja produzido de forma concisa. Outra aplicação comum de SSML é a criação de livros de áudio. Usando a síntese deste tipo de software torna-se possível converter rapidamente a palavra impressa para a palavra falada, tornando-se possível o atendimento para as pessoas que são deficientes visuais ou beneficiar apenas a leitura de qualquer tipo de material, incluindo livros com formatação em braille.

A utilização de SSML vai além da simples criação de uma coleção de palavras que são pronunciadas corretamente. Funções no âmbito de aplicação do SSML também permitem a introdução de ritmo, variância do nível vocal, taxas de som e inflexão. Em alguns casos, os programas criados usando SSML como o núcleo da aplicação
podem permitir a inclusão dos sotaques regionais.

A SSML atualmente compreende 12 elementos, conforme a estrutura do documento, processamento e a pronúncia:

<**speak**>: elemento raiz.

atributo:

- lang : especifica a linguagem usada (lang é também definido por <paragraph> e <sentence>).

<**paragraph**> e <**sentence**>: representam as estruturas internas dos textos.

<**say-as**>: informações sobre o tipo de texto a fim de auxiliar a pronúncia correta, especificando que é um texto para ser interpretado, por exemplo, como moeda, data ou endereço.

atributo:

- type : engloba muitos tipos diferentes, como o tipo de pronúncia (“acronym”, “spell-out”) tipos numéricos (“number”, “ordinal”, “cardinal”, “digits”), tempo e tipos de medida (“date”, “time”, “currency”, “measure”).

<**phoneme**>: provê pronúncia fonética.

atributo:

- alphabet: especifica qual o alfabeto fonético utilizado (por exemplo, IPA - International Phonetic Alphabet).

<**sub**>: sintetiza os substitutos para o texto contido em um atributo “alias”.

atributo:

- alias: especifica o texto correto a ser pronunciado (e.g. _{WWW}).

Prosódia e Estilo

<**voice**>: especifica a voz utilizada.

atributos:

- lang (especificação opcional de língua);
- gender (gêneros: “male”, “female”, “neutral”);
- age (prefência de idade da voz para falar o texto – valor inteiro);
- name (específico da plataforma);
- variant (indica a variação de vozes a serem selecionadas).

<**emphasis**>: O texto a ser falado com ênfase.

atributo:

- level: nível de força do elemento emphasis (“none”, “reduced”, “moderate”, “strong”)

<**break**>: elemento para controlar a pausa entre as leituras.

atributos:

- size (“none”, “small”, “medium”, “large” – opcionais)
- time (duração da pausa em segundos ou milisegundos – opcional)

<**prosody**>: permitir controle sobre a prosódia utilizada.

atributos:

- pitch (baseado em Hertz, valores relativos ou parâmetros “default”, “low”, “medium”, “high”);
- contour (trabalha com intevalos de tempo em porcentagens e valores em Hertz ou porcentagens);

- range (baseado em Hertz, valores relativos ou parâmetros “default”, “low”, “medium”, “high”);

- rate (velocidade da leitura por minuto, valores relativos ou com os parâmetros “default”, “slow”, “medium”, “fast”);

- duration (tempo a ser utilizado para pronunciar um item em milésimos de segundo);

- volume (intervalos de 0.0 a 100.0, valores relativos ou parâmetros “default”, “silent”, “soft”, “medium”, “loud”).

Outros elementos

<**audio**>: incorpora um arquivo de áudio, que é repetido quando o elemento for alcançado.

atributo:

- src (especifica o nome do arquivo de áudio);

<**mark**>: marcador de texto para ser usado tanto para referência interna ou para ser usado externamente por outros documentos.

atributo:

- name (string do nome do evento quando o elemento mark é alcançado – atributo requerido).

Tendo em vista todos estes elementos que podem ser usados no processo de sintetização dentro da SAPI, fizemos uso desta linguagem de marcação para atribuir maior significado na leitura, diminuindo o ruído existente durante a mensagem enviada do emissor ao receptor, ou seja, do próprio TTS ao usuário ou usuários finais.

7 *Plataforma .NET*

Neste capítulo, passamos a abordar a parte prática envolvida em nosso projeto com a especificação do framework empregado. Começaremos com um breve conceito e a seguir uma amostra do que é um framework, sua diferença em relação às bibliotecas e do .NET Framework utilizado na fase de desenvolvimento da aplicação.

7.1 Frameworks e Bibliotecas

Com a complexidade vista nos sistemas atuais, a tarefa de programação fica cada vez mais abrangente, utilizam-se metodologias e técnicas que facilitam a implementação, uma destas técnicas é o reuso de software, onde economizamos esforço e tempo desprendido no desenvolvimento. Podemos pensar em reuso quando temos um baixo acoplamento entre classes, por exemplo.

Pensando em várias aplicações distintas, mas com uma grande intersecção entre estas é que temos a origem do conceito de framework. Trata-se de um conjunto de classes e interfaces que mostra como decompor uma família de problemas, as classes devem ser flexíveis e extensíveis para permitir a construção de várias aplicações com pouco esforço, especificando apenas as particularidades de cada uma (SAUVÉ, 2008).

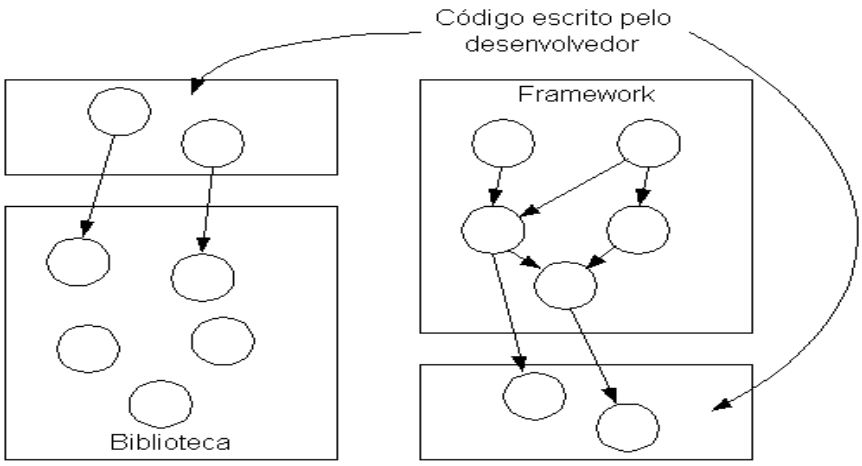


Figura 7.1 Framewors e bibliotecas [6]

[6] Figura extraída do site www.dsc.ufcg.edu.br

A figura 7.1 mostra a diferença entre bibliotecas e frameworks, onde nestes vemos o modelo de colaboração adotado, com os objetos se comunicando, ao contrário das bibliotecas.

Framework se diferencia de uma simples biblioteca (toolkit), pois esta se concentra apenas em oferecer implementação de funcionalidades, sem definir a reutilização de uma solução de arquitetura (design).

Os frameworks podem ser estudados em nível de código e apresentam algumas características (SAUVÉ, 2008):

- Um framework deve ser reusável.
  É o propósito final, para ser reusável, deve primeiro ser usável e bem documentado.

- Deve ser extensível.
  O framework contém funcionalidade abstrata (sem implementação) que deve ser complementada.

- Deve ser eficiente.
  Devido a seu uso em muitas situações, algumas das quais poderão necessitar de maior eficiência.

- Deve ser completo.
  Para endereçar o domínio do problema pretendido.

Entre as vantagens de se utilizar um framework em um projeto, temos a **Reusabilidade** que promove reuso de código e de projeto, com produtividade, qualidade, performance, robustez e interoperabilidade das aplacações.

A **Generalidade** é uma característica que permite a um framework. Ser utilizado utilizado para um domínio de aplicações.

Extensibilidade permite que o domínio do framework seja aumentado ou refinado mais facilmente.

A **Modularidade** facilita a manutenção e o entendimento do framework e das aplicações existentes, também diminui o impacto das mudanças no projeto ou na implementação.

A **Robustez** mantém as funcionalidades mesmo com mudanças na estrutura interna ou externa.

7.2 .NET Framework

O .NET Framework (ou DOTNET) é uma coleção de bibliotecas unificadas, unificação esta provida pelo chamado Ambiente de Execução Independente de Linguagem (Common Language Runtime), entre a grande quantidade de linguagens que fazem parte deste framework, constituindo a assim chamada plataforma .NET, aparece o C# , adotada em nosso projeto.

Na verdade, a plataforma .NET é mais abrangente, visa uma mudança de foco na informática, passando de um mundo de aplicativos, Web sites e dispositivos isolados para uma infinidade de computadores, dispositivos, transações e serviços que se conectam diretamente e trabalham em conjunto para fornecerem soluções mais amplas e ricas.

A plataforma .NET é um ambiente de desenvolvimento poderoso, que permite o desenvolvimento de aplicações DESKTOP (p/ Windows ou console), aplicações para aparelhos móveis (palm-tops, celulares, etc.) e desenvolvimento de aplicações WEB (através da tecnologia ASP.NET).

É um ambiente de desenvolvimento multi-plataforma, multi-linguagens, multi-dispositivos, e que dá suporte total a orientação a objetos, além de conter uma grande biblioteca de classes.

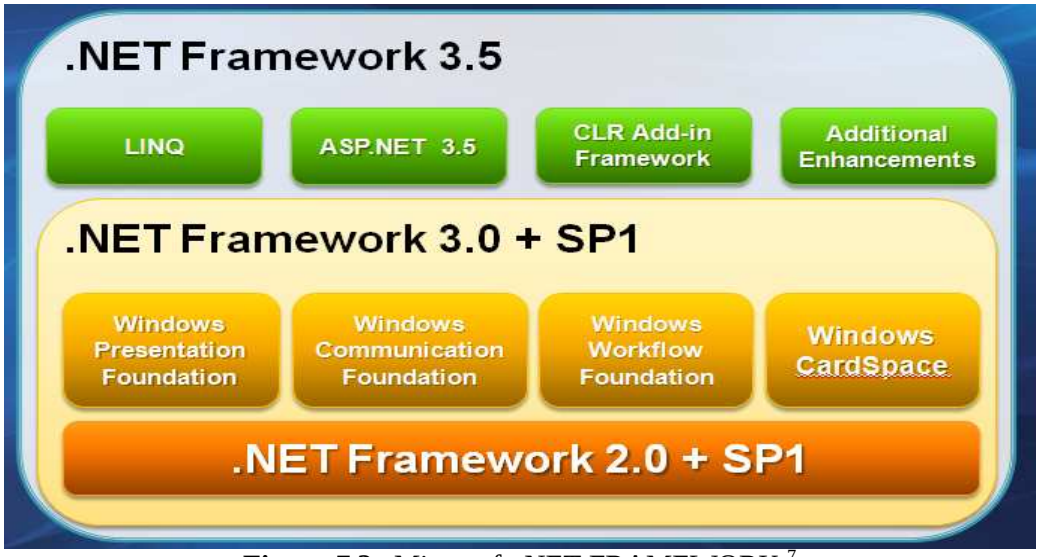


Figura 7.2 Microsoft .NET FRAMEWORK [7]

Na figura 7.2 temos uma visualização da plataforma da Microsoft NET 3.5 empregada no projeto.

[7] Figura extraída do site guy.dotnet-expertise.com

O Microsoft .NET Framework 3.5 combina o poder do .NET Framework 2.0 com novas tecnologias, como visto na figura 7.2, para construção de aplicativos, possibilitando novas experiências, comunicação integrada além de habilidades para vários processos corporativos.

Estas novas tecnologias são:

- **Windows Presentation Foundation**: responsável pela parte de apresentação gráfica, com classes responsáveis por animações, gráficos 2D e 3D, mídia, documentos e layout;

- **Windows Communication Foundation:** abriga tecnologias de comunicação da microsoft;

- **Windows Workflow Foundation:** modelo de programação para o desenvolvimento e execução de estados e fluxos dentro de aplicativos;

- **Windows CardSpace:** solução baseada em padrões para trabalhar e gerenciar várias identidades digitais.

O Microsoft .NET Framework 3.5 foi utilizado por fazer uso do namespace System.Speech que permite a utilização tanto de reconhecimento como de síntese de voz. Sua utilização também se deveu ao fato de acompanhar o Visual Studio, uma IDE para desenvolvimento de aplicações da Microsoft.

8 Desenvolvimento

Neste capítulo passamos à fase de confecção do projeto, onde delineamos nossa rota de ação visando a elaboração do TTS propriamente dito. Na metodologia de desenvolvimento de software, podemos observar em (SILVA, 2004) cinco etapas pertencentes ao chamado ciclo de vida de um software: Análise, Projeto, Implementação, Testes e Manutenção, sendo um processo iterativo.

Todas estas etapas serão apresentadas, embora algumas tenham recebido maior ênfase durante a consecução.

8.1 Análise

A etapa de análise trata de enfatizar o domínio do problema e da sua abstração.

Posto isto, temos o objetivo de criar um software que atenda às exigências para a leitura e compreensão de textos originalmente em língua inglesa, não se preocupando, a priori, com as extensões da API para outros idiomas. Inicialmente ele pretende atender aos requisitos compreendidos pelos outros TTSs existentes no mercado, mas com a meta de sofrer alterações em sua estrutura interna de forma a incorporar novas funcionalidades ao longo de seu ciclo de vida. Estas novas funcionalidades ficam melhor compreendidas quando expusermos o processo de integração de HTML com SSML ainda neste capítulo.

Foram analisadas ferramentas de TTS de domínio público, a exemplo do Festival, e outras integrando sistemas operacionais como o Windows XP e seu Narrator, partimos do princípio a construir um TTS que pudesse executar leituras de textos de modo satisfatório e que fosse possível executar melhorias em sua estrutura ao longo do processo.

Verificamos os seguintes requisitos para o software:

- A aplicação deve executar a leitura de páginas em formato HTML, de forma que o usuário possa compreender as palavras e também ter uma noção da estrutura do documento, noção esta oriunda da ênfase dada a certos elementos presentes numa página HTML (imagens, tabelas, links, quebra de linha, etc.);

- A aplicação deve fornecer suporte audível para elementos não textuais, tais como imagens, links e campos. Informacoes pertinentes no universo de origem HTML;

- Deve ser possível através de um arquivo .HTML armazenado, fazer a sua carga na aplicação;

- A aplicação deve salvar a leitura do documento nos formatos de áudio (mp3 e wave) para posteriormente ser ouvida;

- A aplicação deve oferecer ao usuário a possibilidade de controlar a velocidade da taxa de leitura e o volume;

- A aplicação deve oferecer ao usuário a escolha da leitura simples do texto ou ainda com o algoritmo de inferência de informação não textual.;

- A aplicação deve apresentar uma interface em conformidade com outras interfaces de aplicações vistas na área durante a pesquisa.

Inicialmente reunimos estes requisitos e montamos nosso diagrama de casos de uso para melhor entender as funcionalidades pretendidas. Alguns requisitos se encontram em muitos softwares de vocalização, outros representam requisitos que vão além do trivial, como a leitura de arquivos e configuração da leitura.

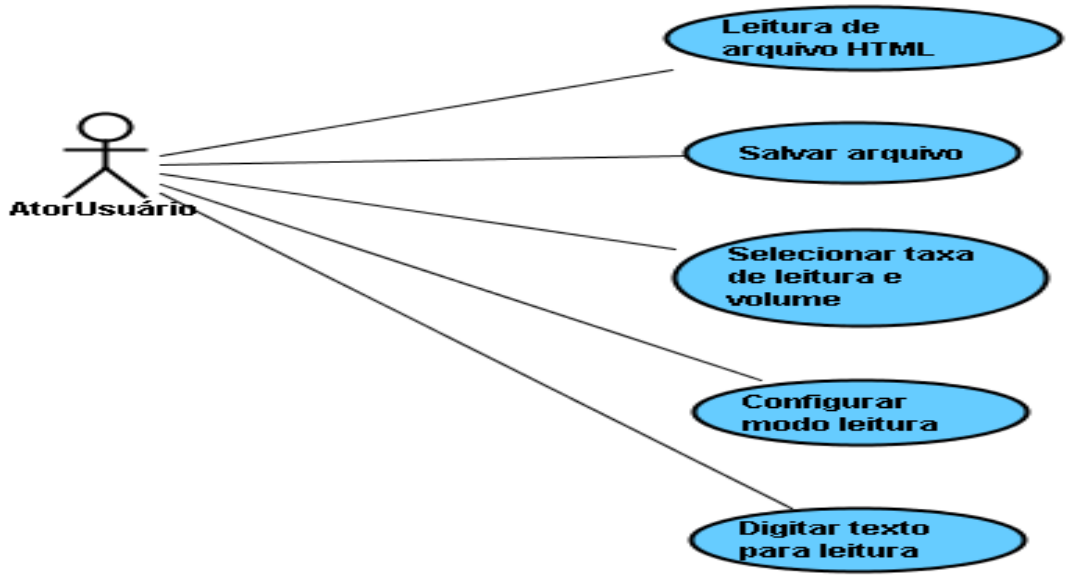


Figura 8.1 Diagrama de casos de uso

A ausência dos demais diagramas de UML nesta etapa fica justificada pelo fato de não termos foco na modelagem, usando os casos de uso apenas para definir melhor nossa visão do projeto.

8.2 Projeto

Esta etapa será abordada de forma sucinta, faremos apenas o detalhamento dos requisitos da etapa anterior, especificando assim a solução problema.

A aplicação permitirá ao usuário abrir uma caixa de pesquisa onde ele escolherá o arquivo em HTML para o processo de conversão. A conversão se dará para o formato destino SSML, já visto no capítulo 6, permitindo com isso, acrescentar maior inteligibilidade no processo de leitura.

Antes de utilizar o SSML, foi preciso definir a estrutura do HTML do qual faríamos a conversão partimos para um estudo da especificação desta linguagem de marcação onde tínhamos uma miríade de tags para tratar dentro de cada documento.

Optamos por tratar das principais tags encontradas e criamos um template para o método de limpeza de HTML a fim de filtrar apenas o texto.

Embora pareça trivial, a solução devia tratar casos particulares de tags onde o texto poderia aparecer de formas diferentes, que precisavam ser reconhecidas para ocorrer a leitura. As chamamos de tags com restrições, como por exemplo é tag referente ao link (< a href >), que não podia ser excluída diretamente pelo fato de perder-se a referência. Outras tags pediam leitura com ênfase, como nos casos de cabeçalho e de negrito, por exemplo, e em outros casos devíamos respeitar uma parada na leitura, como no caso de tabelas.

A vocalização ocorre de duas formas: a primeira é de forma direta, quando o usuário digita um texto qualquer (lembrando que este processo é quase que um padrão nos TTSs); outra forma é abrir um arquivo para a vocalização, requisito este demandando maior atenção nesta etapa.

A escolha de trabalhar com HTML se deveu pela quantidade de documentos neste formato e pela maciça utilização da internet como fonte de pesquisa e entretenimento. A aplicação devia então permitir o carregamento de documentos HTML e uma forma para preparar estes documentos para a vocalização. Nesta etapa ficou claro que devíamos trabalhar em duas etapas distintas:

- Limpeza do HTML;
- Conversão para SSML.

No apêndice D exibimos nosso modelo inicial para tratar a limpeza do HTML, que nada mais é do que um mapeamento feito com o intuito de demarcar as áreas no documento visando o processo de leitura.

A outra etapa é a conversão para o SSML, através da qual visamos aplicar inteligibilidade aliada à leitura. Esta característica foi adquirida através de um estudo feito com este padrão e a utilização de seus atributos junto ao texto das páginas.

Alguns TTSs com esta característica dispõem de opções diferentes quanto ao formato do arquivo para leitura, embora muitos se restrinjam a apenas um formato específico, figurando entre HTML, PDF e alguns outros.

A opção de salvar o documento oferece ao usuário a possibilidade de executar sua leitura em outro momento ou refazê-la para melhor entendimento; a leitura é salva em um formato mp3 e wave.

Os outros requisitos são padrões nos diversos sintetizadores de texto que existem no mercado, oferecendo possibilidade de digitar texto e tratar a leitura com aspectos referentes a volume, taxa e alguns com a opção de trocar as vozes, aspecto este não contemplado em nosso projeto.

Ao fim da etapa de projeto, já tínhamos delineado como faríamos para chegar às funcionalidades do software, além de definir as ferramentas a serem utilizadas, como a API para speech da Microsoft, onde poderíamos obter bons resultados na vocalização em função da história e do histórico de melhorias envolvendo a SAPI vista no capítulo 5.

8.3 Implementação

Utilizando as tecnologias estudadas mais propícias para a aplicação no projeto, a exemplo da SAPI, SSML e do .NET Framework, a implementação do projeto visa confeccionar o software que atenda os requisitos vistos na análise para um sistema de TTS.

Dividimos esta etapa para mostrar uma seqüência lógica do processo de concepção do projeto, as fases seguem a ordem abaixo:

- Tecnologias utilizadas;
- Integração;
- Sintetização.

8.3.1 Tecnologias utilizadas

Com o intuito de oferecer uma breve descrição das tecnologias utilizadas, começamos falando da IDE de desenvolvimento. A figura 8.2 mostra a IDE utilizada para a codificação.

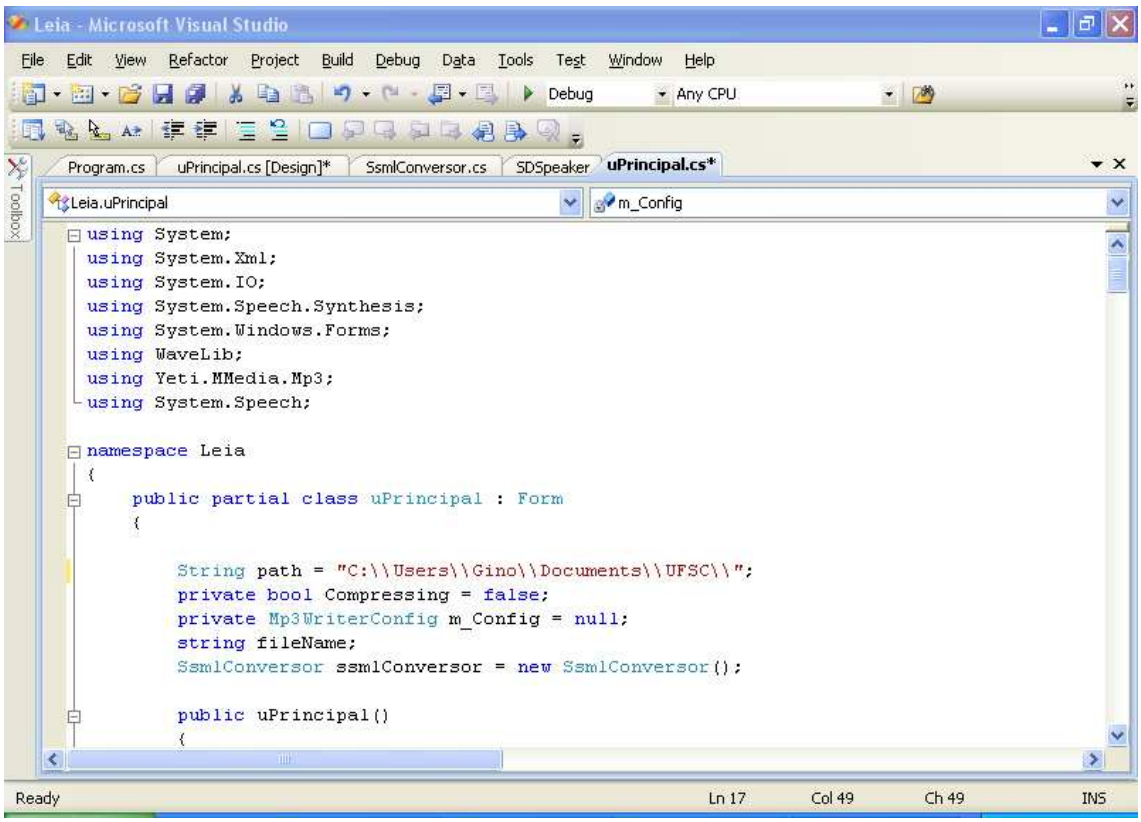


Figura 8.2 Microsoft Visual Studio 2008

O **Visual Studio** é um conjunto de ferramentas integradas da Microsoft para o desenvolvimento de software, dedicado ao .NET framework e às linguagens Visual Basic (VB), C , C++ (C Plus Plus), C# (C Sharp) e J# (Jey Sharp). Também é um grande produto de desenvolvimento na área web, usando a plataforma do ASP.NET.

A linguagem **C #**, usada com o Visual Studio, é uma linguagem de programação orientada a objetos criada pela Microsoft e que faz parte da plataforma .NET. Ela se baseia na estrutura das linguagens C++ e Java.

A API da Microsoft recebeu um capítulo exclusivo de forma a suprí-la deste item. Para mais referências a respeito da SAPI, consultar o capítulo 5.

8.3.2 Integração

Com a utilização da API da Microsoft para vocalização, utilizamos alguns métodos que nos permitiram criar uma aplicação simples, porém já executando a leitura de textos digitados em um campo de edição.

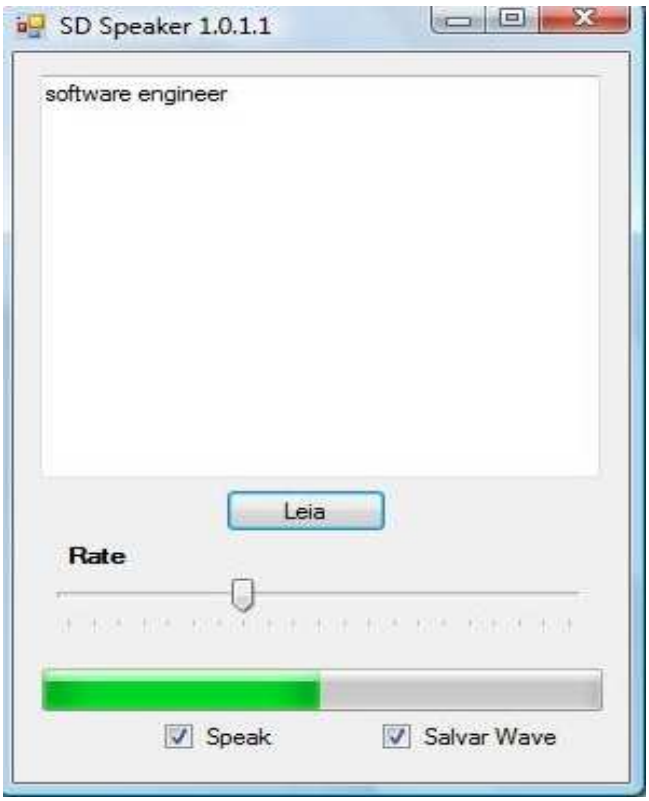


Figura 8.3 Interface inicial

A primeira fase do desenvolvimento abrangia apenas o reconhecimento de textos digitados e a vocalização sem nenhum tratamento.

Havia a necessidade de expandir o domínio, permitindo que outros tipos de documentos pudessem ser lidos, foi então que acoplamos a função de buscar por documentos em HTML, formato amplamento utilizado em paginas acessadas na Internet.

Páginas HTML em código fonte apresentam além dos textos que nos são essenciais na leitura, várias tags de marcação, o que não poderíamos passar para o usuário, houve então a necessidade de fazer um filtro para retirá-las do documento e garantir que apenas o que chegava ao usuário final fosse o texto puro para leitura. Na criação deste filtro, tivemos que estudar a especificação da linguagem HTML a fim de separar as tags que tinham uma relação direta com textos, bem como outras que pudessem ter textos associados, casos em que o texto escrito faz parte de uma tag em especial, como ocorre com links, tabelas, imagens, marcações de tamanho de fonte e várias outras que denotaram esforço no reconhecimento. Esta etapa resultou num documento com as principais tags (inicialmente não pretendemos alcançar todas as páginas HTML, apenas as mais semelhantes à especificação original), e como deveríamos tratá-las em relação à leitura, algumas seriam excluídas diretamente pelo fato de não haver restrições quanto ao texto, já outras precisavam ser tratadas visando verificar o que vinha na seqüência.

Ao final desta etapa, tínhamos apenas o texto, precisávamos aplicar a norma para sintetização (SSML) vista em mais detalhes no capítulo 6.

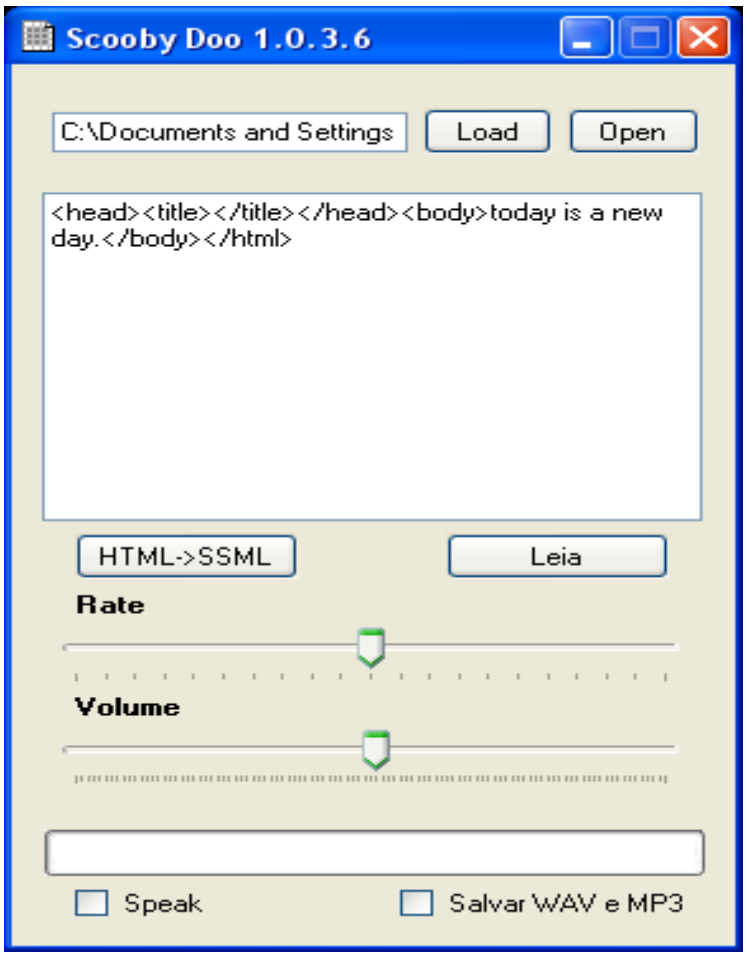


Figura 8.4 Interface carregando arquivo HTML

A figura 8.4 apresenta a interface com modificações. Foram incorporadas as operações de carregar e abrir arquivo, fazer a conversão para SSML, assim como alterar a taxa e o volume da leitura e duas checkbox para executar a leitura e salvar.

Pode-se verificar que todo o HTML é posto na tela sem qualquer tratamento inicial.

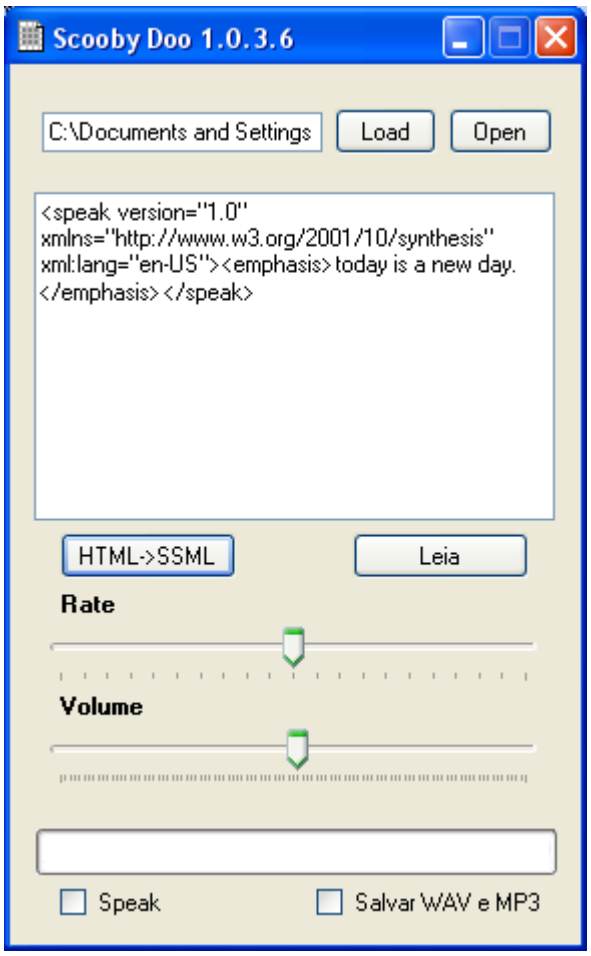


Figura 8.5 Nova interface usando a conversão HTML > SSML

A conversão em SSML vai montar um documento neste formato, de forma que de todo o texto mostrado acima, apenas o que foi lido é o que está entre o atributo de emphasis, ou seja, “today is a new day”.

Para adotar o padrão SSML foi preciso estudar as formas para se tratar o texto e para acrescentar legibilidade e fazer um casamento destes atributos com o tipo de texto que se está lendo, tendo em vista que podemos ter palavras em negrito, itálico, parênteses, parágrafos, aspas e uma série de símbolos e situações dentro de um documento que denotam tratamento adequado para a leitura.

8.3.4 Sintetização

A sintetização ficou a cargo da Microsoft Speech Server (MSS), um servidor para rodar aplicações com suporte a recursos de voz e o Speech SDK 5.1, um kit de desenvolvimento de software para mecanismos de fala e aplicativos para Windows, portanto, ele trabalha junto com a SAPI, esta fornecendo a interface e o outro as ferramentas.

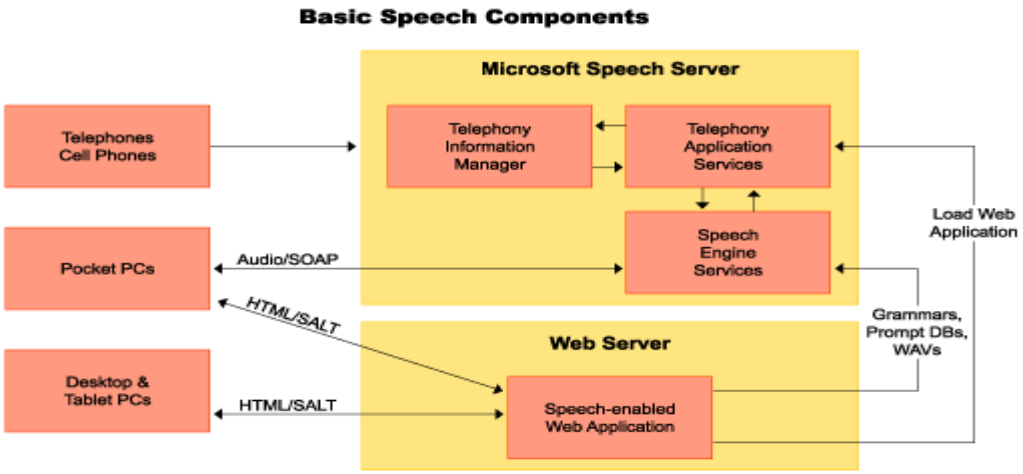


Figura 8.6 Principais componentes para soluções em speech [8]

Especificamente, a Microsoft Speech Server inclui reconhecimento de fala, síntese vocal, conectividade com interfaces, gerenciamento de interfaces para telefones, telefones celulares, computadores e dispositivos Pocket PC.

O Speech Server fornece uma plataforma aberta e baseada apenas em software que suporta as linguagens de programação .NET e normas de voz da Web como o Voice XML, tirando partido do ambiente de programação Visual Studio. É uma plataforma interativa de reconhecimento vocal, que também apresenta plataformas de telefonia, como visto na figura 8.6.

A plataforma contém todos os componentes do servidor para implantar telefonia (voz-somente) e multimodal (voz/visual) para aplicativos. A MSS combina tecnologias Web, serviços de processamento de fala, e capacidades de telefonia em um único sistema, executando assim o reconhecimento de fala e de síntese vocal para aplicações que podem ser acessadas por telefone, celular, Pocket PC, Tablet PC e outros dispositivos.

8.4 Testes

O processo de leitura não pode ser homogeneizado, temos tipos diferentes de leitura e compreensão de textos, o que propomos foi então acrescentar ênfases em pontos onde precisávamos ter conotações diferentes.

A etapa de testes visou desde o início testar estas modificações feitas no SSML e seu resultado no projeto, até que obtivéssemos o esperado na sintetização.

[8] Figura extraída do site www.microsoft.com

No que se refere ao código produzido, corrigimos alguns erros quanto à orientação a objeto; os requisitos iniciais foram satisfatórios, empregamos páginas simples e alcançamos o resultado esperado.

Realizamos alguns testes envolvendo páginas simples, porém com algumas tags a serem verificadas quanto ao desempenho e legibilidade, a seguir exibimos estas páginas e os comentários pertinentes às mesmas.

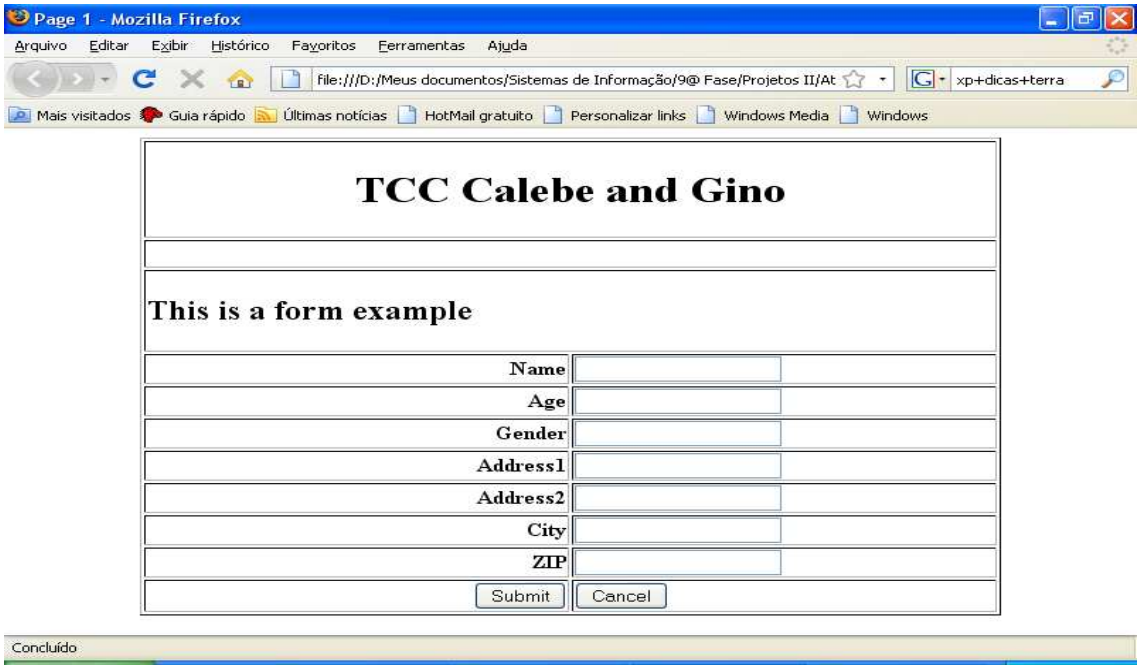


Figura 8.7 Primeira página de teste, com um formulário

Esta página apresentava um formulário com alguns campos. A leitura começava com o título da página e seguia com o título do documento. Entre os campos do formulário incluímos uma parada na leitura para facilitar melhor a compreensão, acrescentando mais sutileza para perceber que se trata de um formulário.

O segundo teste tratava da utilização de links, elementos quase que onipresentes em qualquer documento HTML.

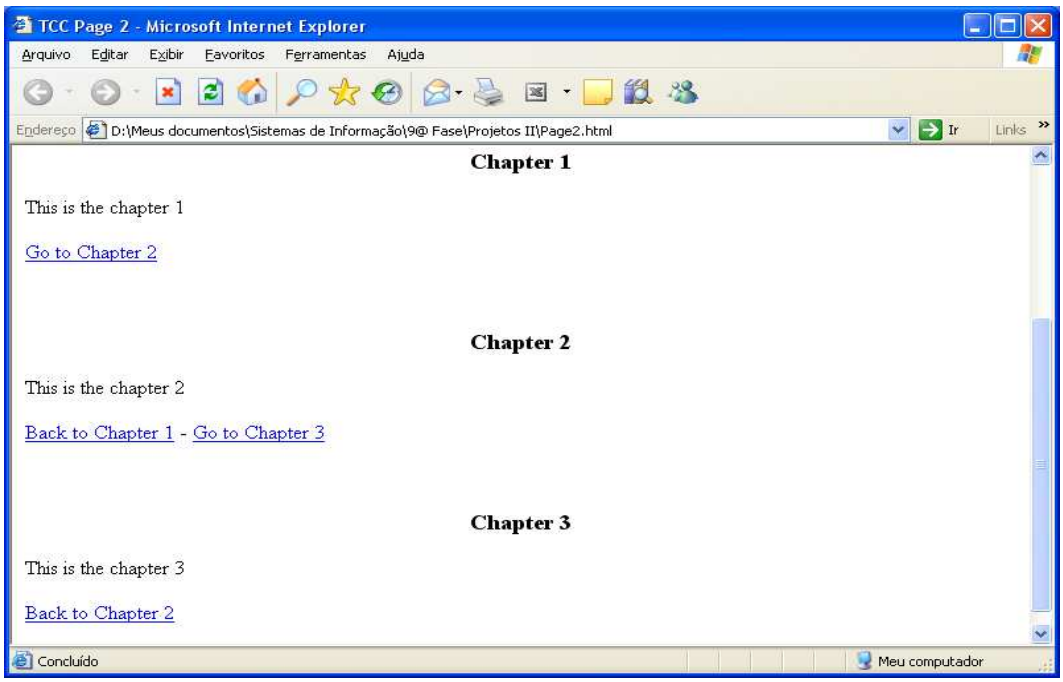


Figura 8.8 Segunda página de teste, utilizando links

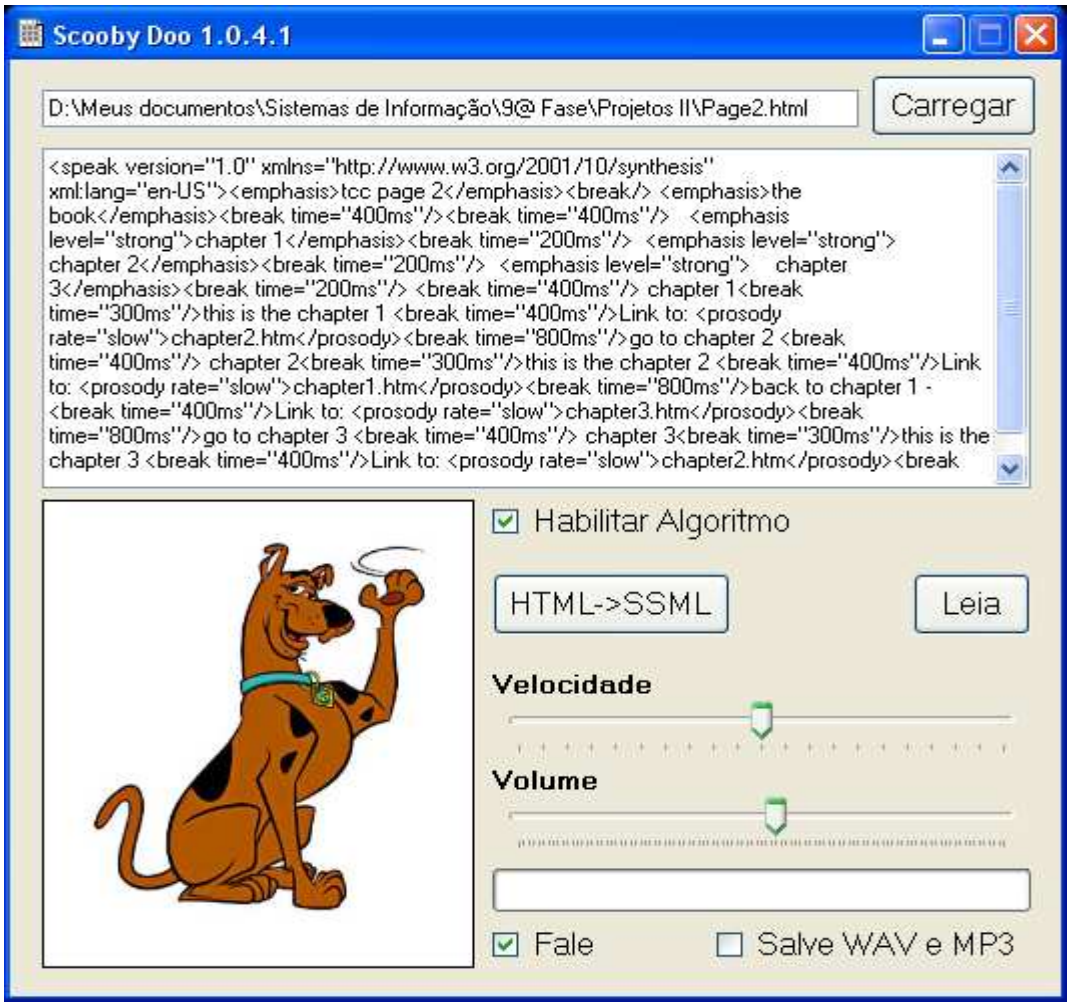


Figura 8.9 Resultado da conversão para SSML do teste 2 na aplicação

A figura 8.9 mostra a execução da página de teste 2 no software, a opção Habilitar Algoritmo está marcada e a leitura segue utilizando a SSML, como podemos ver na caixa de edição, trabalhando com o elemento break entre os textos e nos links utilizando a prosody rate = “slow”, sendo entendida como um ritmo de leitura mais lenta. O resultado final foi interessante, principalmente por haver muita diferença entre leituras utilizando o algoritmo em relação às que não o utilizavam.

A respeito do checkbox Habilitar Algoritmo, ele se refere à inferência de informações extraídas no texto, para melhorar a leitura. Ele pode, por exemplo, transformar negrito do HTML para emphasis no SSML. O algoritmo ainda se preocupa com elementos presentes em páginas HTML, que não são textos (formulários, imagens e links). Para estes casos, o algoritmo se preocupa em informar ao usuário quem em um determinado trecho da página, possui uma imagem ou campo de formulário com um nome que o programador atribuiu. Sobre a leitura do link, ele faz a leitura completa do link para a qual esta página faz referência.

O terceiro teste incluía três figuras em uma página, mídias estáticas muito freqüentes e que também precisavam de verificação, além de palavras em negrito.

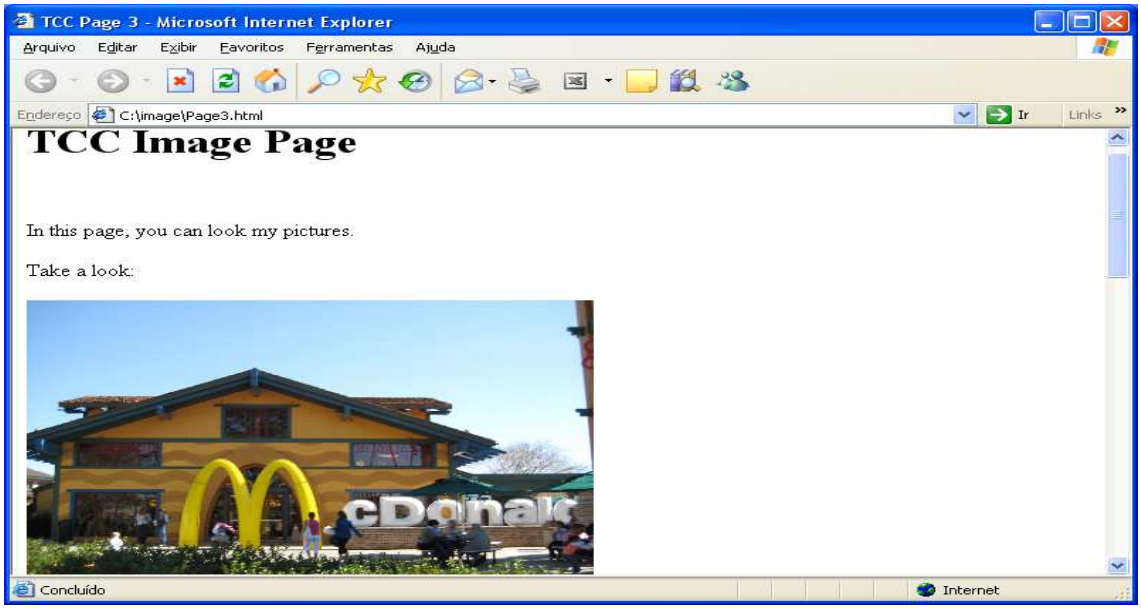


Figura 8.10 Página de teste 3 utilizando figuras

Na aplicação o resultado é mostrado a seguir. Os nomes das figuras são lidos de forma mais lenta, frisando que temos elementos diferentes no texto.

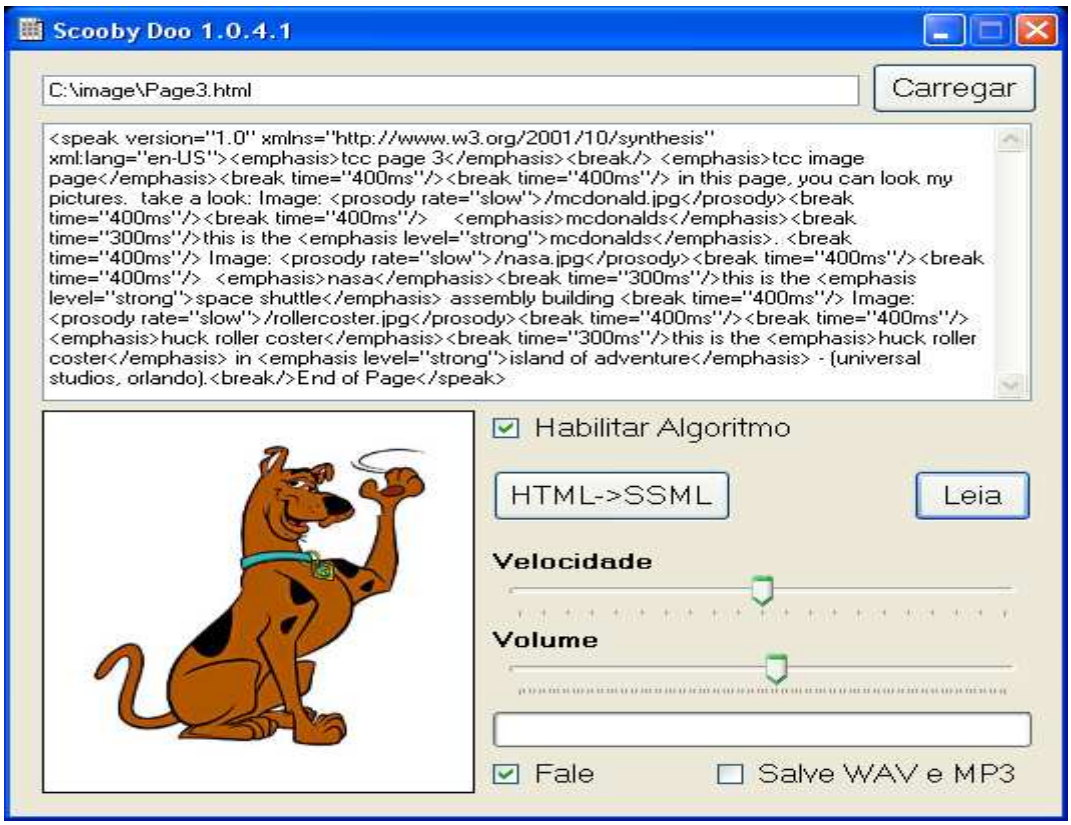


Figura 8.11 Resultado da conversão do teste 3 na aplicação

Pode-se perceber pelo código na figura 8.11 que é utilizado um nível de ênfase strong, ou seja, mais forte para a palavra em negrito **Island of Adventure,** realçando-a.

Os resultados mostraram que é possível obter bons resultados na legibilidade, mesmo sem se utilizar de muitos atributos diferentes, sabendo que a SSML dispõem de outras formas de definir a forma de vocalização.

Nos testes também chegamos à interface final resultante das etapas anteriores.

Aqui fizemos algumas modificações, uma delas diz respeito ao tamanho dos botões, que sofreram aumento, também acrescentamos uma opção de leitura com ou sem algoritmo, que nada mais é do que o usuário optar por ler com as ênfases do SSML ou normalmente, de forma corrida. Também mudamos a forma de carregar o arquivo que estava um pouco confusa, agora se utiliza o mesmo botão tanto para carregar como para abrir o arquivo.

8.5 Manutenção

Modificações futuras pedem a utilização de mais vozes e também a possibilidade de tratar melhor a leitura em português com a incorporação de bibliotecas para tal. Uma maior flexibilidade para o usuário executar a leitura também abrange o escopo.

Conjunto de melhorias futuras:

- Menos classes utilizadas para conversão em formato de áudio;
- Possibilidade de abrir arquivos em PDF;
- Inclusão de vozes;
- Adequação ao português do Brasil;
- Opção para ajuda;
- Tratar o reconhecimento de fala, visando conversar com a máquina;
- Independência de framework para executar a aplicação.

8.6 Visão do usuário

Na confecção de software é preciso saber dosar os diferentes tipos de visões, a visão do desenvolvedor, na qual temos garantidas nossas expectativas, e a visão do usuário, que nem sempre consegue conceber o programa da mesma forma. Ressalta esta última sentença, o fato de termos usuários especiais que pedem certas características extras, como tamanho dos botões, por exemplo.

Em (SILVA, 2004) temos durante o ciclo de vida um exemplo do que diferentes visões de um mesmo software pode influir no seu resultado final, desde a especificação de requisitos até o que o usuário realmente queria.

Este projeto não tem como objetivo ser totalmente voltado para portadores de deficiência, ele será sim uma ferramenta de apoio, onde o portador de deficiências visuais possa obter uma leitura com a qualidade que ele não teria caso fosse tentar ler um texto comum usando o Narrator do Windows XP. Partimos do princípio que uma pessoa totalmente cega teria que ter a ajuda de alguém para utilizar o TTS, nos preocupando apenas com o resultado final, a leitura.

Como mostrado no início deste capítulo, a interface é simples. A seguir apresentamos os passos que um usuário comum utilizaria para obter sintetização de documentos HTML:

- Executar a aplicação, sabendo que esta roda em sistemas operacionais Microsoft Windows;
- Clicar no botão Carregar para escolher um arquivo HTML qualquer que ele deseja ler;
- Clicar no botão Abrir para abrir o documento na tela da aplicação;
- Clicar no botão HTML -> SSML para executar a conversão;

- Setar a checkbox Fale;

- Clicar em Leia para o software executar a leitura;

- Pode finalizar setando no checkbox Salve WAVE e MP3;

- Pode também durante o processo de leitura ou antes configurar o volume e a velocidade de leitura;

- Um usuário comum pode também digitar um texto, clicar em HTML -> SSML, selecionar Leia e clicar no botão Leia pra ler diretamente da tela;

- Um usuário também pode optar por executar a leitura sem o SSML, com o texto corrido.

Para um usuário totalmente cego, a relevância se dará na execução da leitura e apenas, fase esta executada durante a próxima etapa de testes, onde fomos trabalhando com as modificações e seus resultados a priori.

9 Conclusões e Trabalhos Futuros

Percorrendo o caminho inicial nesta jornada que foi a criação deste projeto, confessamos existir um paradoxo, por um lado já tínhamos em mente os percalços e empecilhos encontrados nesta estrada referentes à simples oração: “fazer o computador falar”, o que denotava algum esforço e despreendimento. Aliado ao fazer falar, também havia a preocupação de se fazer entender, em uma conversa onde somente uma pessoa fala, a outra fica encarregada de compreender a mensagem passada. O paradoxo diz respeito a, por um lado, termos muitos exemplos de ferramentas para síntese de voz, inclusive inclusas em sistemas operacionais, mesmo que muitos usuários nem sequer saibam deste fato, por outro lado, poucas conseguem obter resultados satisfatórios quanto à legibilidade alcançada, tornando a leitura cansativa e incompreensível.

Existem muitas entidades no Brasil abraçando a causa da inclusão digital, algumas apenas por questões filantrópias e outras identificando um público até então ignorado. Universidades também contribuem para o desenvolvimento de aplicações na área, a exemplo da Universidade Federal do Rio de Janeiro com o DOSVOX e a USP também colaborando com esta na rede SACI além de muitas outras, mostraram que é possível desenvolver software de qualidade no Brasil para pessoas com deficiência visual.

A Microsoft através da SAPI já mostrava resultados na área de vocalização com uma API criada em 1995 e que passou por várias versões até hoje, o que nos chamou a atenção para aplicá-la no projeto. A SAPI se destina a vários outros tipos de aplicações e tem seu uso cada vez mais acentuado hoje em dia em questões de reconhecimento de voz e em toda a parte de telefonia celular com a MSS.

Apesar de não fazer parte do escopo de nosso projeto, delineamos um espaço importante quanto ao processo de reconhecimento de voz, donde temos uma preocupação maior com a captação de voz e todas as suas nuances, como sotaque, tom e etc. É uma área também importante e que pelo menos em sua teoria se fez presente no trabalho.

A utilização de etapas no desenvolvimento de software foi vital para se chegar aos requisitos iniciais e como poderíamos executá-los no projeto, de forma a obter um resultado satisfatório, os testes foram úteis no sentido de verificar o que precisava ser corrigido, a exemplo da interface e alguns elementos do SSML.

O SSML foi um elemento valioso no que tange a uma maior legibilidade do processo de leitura, haja vista a quantidade de elementos destinados a esse fim que se encontra em sua especificação e do qual nos servimos em parte.

9.1 Resultados Obtidos

Os resultados obtidos forma enumerados a seguir, partindo de nossos objetivos iniciais e também dos requisitos da etapa de análise:

- Criação de um software leve e de fácil manuseio;
- Leitura simples de textos digitados;
- Carregamento e tratamento de arquivos HTML para leitura;
- Utilização da SSML na vocalização;
- Opção de salvar arquivo em formatos de áudio (mp3 e wave);
- Opção de leitura sem o SSML;
- Controle de volume e velocidade.

Além destas funcionalidades, merece destaque o resultado final de leitura comparando-se a utilização da norma e a sua exclusão, resultados muito diversos em alguns casos, principalmente quando temos elementos variados no HTML vistos na etapa de testes, aplicações de pausas, ênfases e prosódia fazem muita diferença no resultado final.

O grande diferencial e motivo deste trabalho foi criar um TTS de qualidade e que se utilizasse melhor de elementos para realizar a leitura. Não nos preocupamos em fazê-lo compatível com vários formatos de documento e tampouco com outros idiomas além do inglês, nossa preocupação residiu na pronúncia e acreditamos ter alcançado este intento.

9.2 Trabalhos Futuros

Visando trabalhos futuros temos um objetivo de definir um número maior de tags abrangendo assim um domínio mais vasto quanto à leitura de páginas, o que não foi nosso primeiro objetivo.

A dependência de plataforma figura como um mudança, não fica muito útil obrigar a utilização do .NET Framework para rodar a aplicação, pois diminui sua abrangência e alcance.

Reiterando o que mencionamos na etapa de manutenção, existem muitas possibilidades de melhoria se utilizarmos a API para vocalização. A utilização de vozes é um requisito em outros TTSs e dá mais poder de escolha para o usuário final. Novos idiomas também podem ser incorporados à API e posteriormente utilizados na leitura.

9.3 Considerações Finais

A principal motivação de ter realizado este trabalho residiu na possibilidade de obter um resultado de qualidade na leitura, área considerada por alguns como de resultados não muito satisfatórios. Porém não foi o que descobrimos, acreditamos que futuramente poderemos obter resultados ainda melhores no processo de vocalização.

Cremos que este trabalho encontra grande relevância pelo fato de oferecer uma pesquisa bastante extensa na área, encobrindo muitas ferramentas, tecnologias e pesquisadores envolvidos no tema, servindo assim como uma fonte para pesquisas posteriores. A aplicação também se destaca por oferecer um algo a mais em relação às outras, seja na utilização do algoritmo para vocalização, seja na sintentização do HTML.

Anexo A Código-fonte: Scooby-Doo Speaker

As listagens que seguem apresentam o código-fonte da aplicação batizada como Scooby-Doo Speaker com as classes julgadas de maior relevância pelos autores no contexto deste trabalho.

Listagem A.1: Classe Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace Leia
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Principal());
        }
    }
}
```

Listagem A.2: *Classe HtmlToSsmlConversor.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Leia
{
    class HtmlToSsmlConversor
    {
```

```
        public string InsertHeader(string pString)
        {
            string header = "";
            header += "<speak version=\"1.0\"
xmlns=\"http://www.w3.org/2001/10/synthesis\" xml:lang=\"en-US\">";
            pString = header + pString;

            return pString;
        }

        public string InsertFooter(string pString, bool pApply)
        {
            string footer = "";

            if (pApply)
            {
                footer += "<break/>End of Page";
            }

            footer += "</speak>";
            pString += footer;

            return pString;
        }

        public string RemoveString(string pString, String[] pRemove)
        {//recebe um array de strings para serem removidas, e apenas remove
do texto.
            foreach (string pUniqueRemove in pRemove)
            {
                pString = pString.Replace(pUniqueRemove, " ");
            }
            return pString.Trim();
        }

        public string RemoveBetween(string pString, string pStart, string
pEnd)
        {/* recebe o inicio e ira remover tudo ateh encontrar a PROXIMA
string igual ao fim
          * e.g. removeBetween(edtReadme.Text, "<title>", "</title>");
          * usado para remover tags com assinaturas unicas.
          */
            try
            {
                return pString.Remove(pString.IndexOf(pStart),
pString.IndexOf(pEnd) - pString.IndexOf(pStart) + pEnd.Length);
            }
            catch
            {
                return pString;
            }
        }

        public string RemoveBetween(string pString, String[][] pStartEnd)
        {//overload do metodo acima com a diferenca de receber em array
bidimensional
         /*e.g. removeBetween(edtReadme.Text, new String[][] {    new
String[] { "<title>", "</title>" },
          *                                                     new
String[] { "<title>", "</title>" }
```

```
            *                                                   });
            */
              foreach (string[] pUniqueStartEnd in pStartEnd)
              {
                  try
                  {
                      pString.Remove(pString.IndexOf(pUniqueStartEnd[0]),
pString.IndexOf(pUniqueStartEnd[1]) - pString.IndexOf(pUniqueStartEnd[0]) +
pUniqueStartEnd[1].Length);
                  }
                  catch
                  {

                  }
              }
              return pString;
          }

          public string RemoveUntil(string pString, String[][] pRemoveUntil)
          {  /* igual ao removeBetween porem com a preocupacao de ao remover o
marcador de final, encontrar o primeiro marcador
              * igual ao final, apos ter sido encontrado o inicio. Deve ser
usado por exemplo, para remover um inicio ateh encontrar ">"
              * e.g. removeUntil(edtReadme.Text, new String[][] {  new
String[] { "<head", ">" },
              *                                                   new
String[] { "<!--", "-->" },
              *                                                   new
String[] { "<EMBED", ">" },
              *                                                   new
String[] { "<BGSOUND", ">" }}
              */
              int searchAfter;
              bool change = true;
              string initialText;
              string finalText;

              foreach (string[] pUniqueUntil in pRemoveUntil)
              {
                  change = true;

                  while (change)
                  {
                      try
                      {
                          initialText = pString;
                          searchAfter = pString.IndexOf(pUniqueUntil[0]);
                          pString =
pString.Remove(pString.IndexOf(pUniqueUntil[0]),
pString.IndexOf(pUniqueUntil[1], searchAfter) -
pString.IndexOf(pUniqueUntil[0]) + pUniqueUntil[1].Length);
                          finalText = pString;

                          change = !(initialText.Equals(finalText));
                      }
                      catch
                      {
                          change = false;
                      }
                  }
```

```
            }
            return pString;
        }

        public string ReplaceString(string pString, String[][] pReplace)
        {   /* Usado para fazer simples troca de strings. Util para fazer
trocas de tags por elementos break de ssml.
             * e.g. replaceString(edtReadme.Text, new String[][] {    new
String[] { "</title>", "<break/>" },
             *                                                        new
String[] { "</body>", "<break/>" } }
             *                                                    );
             *
             */

            bool change = true;
            string initialText;
            string finalText;

            initialText = pString;

            foreach (string[] pUniqueReplace in pReplace)
            {
                change = true;

                while (change)
                {
                    initialText = pString;

                    pString = pString.Replace(pUniqueReplace[0],
pUniqueReplace[1]);

                    finalText = pString;

                    change = !(initialText.Equals(finalText));
                }
            }

            return pString;
        }

        public string RemoveLink(string pString)
        {
            bool change = true;
            string initialText;
            string finalText;

            initialText = pString;

            while (change)
            {
                try
                {
                    int startLink;
                    int startRef;
                    int endRef;
                    string pathLink;
                    startLink = pString.IndexOf("<a");
                    startRef = pString.IndexOf("href=", startLink) + 6;
```

```
                    endRef = pString.IndexOf("\"", startRef);

                    pathLink = pString.Substring(startRef, endRef -
startRef);

                    pString = pString.Remove(startLink, pString.IndexOf(">",
startLink) - startLink + 1);

                    int endLink;
                    endLink = pString.IndexOf("</a>", startLink);
                    pString = pString.Remove(endLink, 4);
                    pString = pString.Insert(startLink, "<break
time=\"400ms\"/>Link to: <prosody rate=\"slow\">" + pathLink +
"</prosody><break time=\"800ms\"/>");

                    finalText = pString;

                    change = !(initialText.Equals(finalText));
                }
                catch (Exception)
                {
                    change = false;
                }
            }
            return pString;
        }

        public string RemoveInput(string pString)
        {
            bool change = true;
            string initialText;
            string finalText;

            initialText = pString;

            while (change)
            {
                try
                {
                    int startInput;
                    int startName;
                    int endName;
                    string inputName;
                    startInput = pString.IndexOf("<input");
                    startName = pString.IndexOf("name=", startInput) + 6;
                    endName = pString.IndexOf("\"", startName);

                    inputName = pString.Substring(startName, endName -
startName);

                    pString = pString.Remove(startInput,
pString.IndexOf(">", startInput) - startInput + 1);
                    pString = pString.Insert(startInput, "Field: <prosody
rate=\"slow\">" + inputName + "</prosody><break time=\"400ms\"/>");


                    finalText = pString;

                    change = !(initialText.Equals(finalText));
                }
```

```
                catch (Exception)
                {
                    change = false;
                }
            }
            return pString;
        }

        public string RemoveImage(string pString)
        {
            bool change = true;
            string initialText;
            string finalText;

            initialText = pString;

            while (change)
            {
                try
                {
                    int startImage;
                    int startName;
                    int endName;
                    string imageName;

                    startImage = pString.IndexOf("<img");
                    startName = pString.IndexOf("src=", startImage) + 5;
                    endName = pString.IndexOf("\"", startName);

                    imageName = pString.Substring(startName, endName -
startName);

                    if (imageName.Contains("/"))
                    {
                        imageName =
imageName.Substring(imageName.LastIndexOf("/"), imageName.Length -
imageName.LastIndexOf("/"));
                    }

                    pString = pString.Remove(startImage,
pString.IndexOf(">", startImage) - startImage + 1);
                    pString = pString.Insert(startImage, "Image: <prosody
rate=\"slow\">" + imageName + "</prosody><break time=\"400ms\"/>");

                    finalText = pString;

                    change = !(initialText.Equals(finalText));
                }
                catch (Exception)
                {
                    change = false;
                }
            }
            return pString;
        }

        public string ConvertSsml(string pString, bool pApply)
        {
            if (!pApply)
            {
```

```
                pString = this.RemoveUntil(pString, new String[][] { new
String[] { "<", ">" } });

            }
            else
            {

                pString = this.RemoveUntil(pString, new String[][] {  new
String[] { "<head", ">" },
                                                                       new
String[] { "<!--", "-->" },
                                                                       new
String[] { "<embed", ">" },
                                                                       new
String[] { "<bgsound", ">" },
                                                                       new
String[] { "<form", ">" },
                                                                       new
String[] { "<hr", ">" },
                                                                       new
String[] { "<select", "</select>" },
                                                                       new
String[] { "<ol", ">" },
                                                                       new
String[] { "<marquee", ">" },
                                                                       new
String[] { "<meta", ">" },
                                                                       new
String[] { "<p", ">" },
                                                                       new
String[] { "<table", ">" },
                                                                       new
String[] { "<td", ">" },
                                                                       new
String[] { "<tr", ">" },
                                                                       new
String[] { "<div", ">" },
                                                                       new
String[] { "<th", ">" },
                                                                       new
String[] { "<ul", ">" },
                                                                       new
String[] { "<!doctype", ">" },
                                                                       new
String[] { "<html", ">" },
                                                                       new
String[] { "<h2", ">" },
                                                                       new
String[] { "<h3", ">" },
                                                                       new
String[] { "<h4", ">" },
                                                                       new
String[] { "<h5", ">" },
                                                                       new
String[] { "<h6", ">" },
                                                                       new
String[] { "<body", ">" }
                                                                       });
```

```
                pString = this.ReplaceString(pString, new String[][] { new String[] { "<b>", "<emphasis>" },
                                                                        new String[] { "</b>", "</emphasis>" },

                                                                        new String[] { "<big>", "<emphasis>" },
                                                                        new String[] { "</big>", "</emphasis>" },

                                                                        new String[] { "<em>", "<emphasis>" },
                                                                        new String[] { "</em>", "</emphasis>" },

                                                                        new String[] { "<h1>", "<emphasis>" },
                                                                        new String[] { "</h1>", "</emphasis><break time=\"400ms\"/>" },

                                                                        new String[] { "<i>", "<emphasis>" },
                                                                        new String[] { "</i>", "</emphasis>" },

                                                                        new String[] { "<small>", "<emphasis level=\"reduced\">" },
                                                                        new String[] { "</small>", "</emphasis>" },

                                                                        new String[] { "<strong>", "<emphasis level=\"strong\">" },
                                                                        new String[] { "</strong>", "</emphasis>" },

                                                                        new String[] { "<title>", "<emphasis>" },
                                                                        new String[] { "</title>", "</emphasis><break/>" },

                                                                        new String[] { "<br>", "<break time=\"400ms\"/>" },
                                                                        new String[] { "<br />", "<break time=\"400ms\"/>" },
                                                                        new String[] { "</tr>", "<break time=\"200ms\"/>" },
                                                                        new String[] { "</td>", "<break time=\"200ms\"/>" },
                                                                        new String[] { "</u>", "<break time=\"200ms\"/>" },
                                                                        new String[] { "calebe", "kalebe" },
                                                                        new String[] { "<li>", "<emphasis level=\"strong\">" },
                                                                        new String[] { "</li>", "</emphasis><break time=\"200ms\"/>" },

                                                                        new String[] { "</h2>", "<break time=\"300ms\"/>" },
```

```
                                                                    new
String[] { "</h3>", "<break time=\"300ms\"/>" },
                                                                    new
String[] { "</h4>", "<break time=\"300ms\"/>" },
                                                                    new
String[] { "</h5>", "<break time=\"300ms\"/>" },
                                                                    new
String[] { "</h6>", "<break time=\"300ms\"/>" },

                                                                    new
String[] { " ", "<break time=\"400ms\"/>" },

                                                                    });



                pString = this.RemoveString(pString, new String[] {
"<center>", "</center>", "<dl>", "<dt>", "<dd>",
                "</dl>", "</dt>", "</dd>", "</font>", "</form>",
                "<menu>", "</menu>", "</ol>", "</marquee>", "</table>",
"</th>", "<u>", "</ul>", "</html>", "<head>",
                "</head>", "<body>", "</body>", "<label>", "</label>",
"</div>", "<p>", "</p>"   });

                pString = this.RemoveLink(pString);
                pString = this.RemoveInput(pString);
                pString = this.RemoveImage(pString);


            }

            pString = this.InsertHeader(pString);
            pString = this.InsertFooter(pString, pApply);

            return pString;
        }

    }
}
```

Listagem A.3: *Classe Principal.cs*

```
using System;
using System.Xml;
using System.IO;
using System.Speech.Synthesis;
using System.Windows.Forms;
using WaveLib;
using Yeti.MMedia.Mp3;
using System.Speech;

namespace Leia
{
```

```
    public partial class Principal : Form
    {

        String path = "C:\\Users\\Calebe\\Documents\\UFSC\\";
        private bool Compressing = false;
        private Mp3WriterConfig m_Config = null;
        string fileName;
        string waveFile;
        HtmlToSsmlConversor ssmlConversor = new HtmlToSsmlConversor();

        public Principal()
        {
            InitializeComponent();
        }

        private int GetRate()
        {
            return tbRate.Value;
        }

        private int GetVolume()
        {
            return tbVolume.Value;
        }

        private void Compress(String pName)
        {
            if (File.Exists( waveFile.Replace(".wav", ".5mp3")/*path + pName
+ ".5mp3"*/) &&
                (MessageBox.Show(this, "Override the existing file?", "File
exists",
                                        MessageBoxButtons.YesNo,
MessageBoxIcon.Question) != DialogResult.Yes))
            {
                return;
            }

            WaveStream s = new WaveStream(waveFile/*path + pName +
".wav"*/);
            try
            {
                m_Config = new Mp3WriterConfig(s.Format);
            }
            finally
            {
                s.Close();
            }

            try
            {
                progressBar.Value = 0;

                Compressing = true;
                try
                {
                    WaveStream InStr = new WaveStream(waveFile/*path + pName
+ ".wav"*/);
                    try
                    {
```

```
                        Mp3Writer writer = new Mp3Writer(new
FileStream(waveFile.Replace(".wav",".mp3")/*path + pName + ".mp3"*/,
FileMode.Create), m_Config);
                        try
                        {
                            byte[] buff = new
byte[writer.OptimalBufferSize];
                            int read = 0;
                            int actual = 0;
                            long total = InStr.Length;
                            Cursor.Current = Cursors.WaitCursor;
                            try
                            {
                                while ((read = InStr.Read(buff, 0,
buff.Length)) > 0)
                                {
                                    Application.DoEvents();
                                    writer.Write(buff, 0, read);
                                    actual += read;
                                    progressBar.Value = (int)(((long)actual
* 100) / total);

                                    Application.DoEvents();
                                }
                                //this.Text = "Audio Compress - Done";
                            }
                            finally
                            {
                                Cursor.Current = Cursors.Default;
                            }
                        }
                        finally
                        {
                            writer.Close();
                        }
                    }
                    finally
                    {
                        InStr.Close();
                    }
                }
                finally
                {
                    Compressing = false;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(this, ex.Message, "An exception has ocurred
with the following message", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void btnLoad_Click(object sender, EventArgs e)
        {
            this.Cursor = Cursors.WaitCursor;

            OpenFileDialog openFile = new OpenFileDialog();
            openFile.InitialDirectory =
System.Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
```

```
            openFile.Title = "Selecione o arquivo...";
            openFile.Filter = "Htm files (*.htm*)|*.htm*|Html files
(*.html*)|*.htm*";
            openFile.RestoreDirectory = true;

            if (openFile.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    this.fileName = openFile.FileName;
                    this.edtFile.Text = fileName;

                    this.waveFile = fileName.Replace(".html", ".wav");

                    if (waveFile == fileName) {
                        this.waveFile = fileName.Replace(".htm", ".wav");
                    }
                    this.btnLoad.Visible = false;
                    this.btnOpen.Visible = true;
                }
                catch (IOException ioe)
                {
                    MessageBox.Show(this, ioe.Message.ToString(),
ioe.Source.ToString(), MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            this.Cursor = Cursors.Default;
        }

        private void ApplyAlgoritm(bool pApply)
        {

            edtReadMe.Text = this.ssmlConversor.RemoveUntil(edtReadMe.Text,
new String[][] {  new String[] { "<head", ">" },

new String[] { "<!--", "-->" },

new String[] { "<embed", ">" },

new String[] { "<bgsound", ">" },

new String[] { "<form", ">" },

new String[] { "<hr", ">" },

new String[] { "<select", "</select>" },

new String[] { "<ol", ">" },

new String[] { "<marquee", ">" },

new String[] { "<meta", ">" },

new String[] { "<p", ">" },

new String[] { "<table", ">" },

new String[] { "<td", ">" },

new String[] { "<tr", ">" },
```

```
new String[] { "<div", ">" },

new String[] { "<th", ">" },

new String[] { "<ul", ">" },

new String[] { "<!doctype", ">" },

new String[] { "<html", ">" },


new String[] { "<h2", ">" },

new String[] { "<h3", ">" },

new String[] { "<h4", ">" },

new String[] { "<h5", ">" },

new String[] { "<h6", ">" },

new String[] { "<body", ">" }

});
            if (pApply)
            {

                edtReadMe.Text =
this.ssmlConversor.ReplaceString(edtReadMe.Text, new String[][] {    new
String[] { "<b>", "<emphasis>" },

new String[] { "</b>", "</emphasis>" },


new String[] { "<big>", "<emphasis>" },

new String[] { "</big>", "</emphasis>" },


new String[] { "<em>", "<emphasis>" },

new String[] { "</em>", "</emphasis>" },


new String[] { "<h1>", "<emphasis>" },

new String[] { "</h1>", "</emphasis><break time=\"400ms\"/>" },


new String[] { "<i>", "<emphasis>" },

new String[] { "</i>", "</emphasis>" },


new String[] { "<small>", "<emphasis level=\"reduced\">" },

new String[] { "</small>", "</emphasis>" },
```

```
new String[] { "<strong>", "<emphasis level=\"strong\">" },

new String[] { "</strong>", "</emphasis>" },


new String[] { "<title>", "<emphasis>" },

new String[] { "</title>", "</emphasis><break/>" },


new String[] { "<br>", "<break time=\"400ms\"/>" },

new String[] { "<br />", "<break time=\"400ms\"/>" },

new String[] { "</tr>", "<break time=\"200ms\"/>" },

new String[] { "</td>", "<break time=\"200ms\"/>" },

new String[] { "</u>", "<break time=\"200ms\"/>" },

new String[] { "calebe", "kalebe" },

new String[] { "<li>", "<emphasis level=\"strong\">" },

new String[] { "</li>", "</emphasis><break time=\"200ms\"/>" },


new String[] { "</h2>", "<break time=\"300ms\"/>" },

new String[] { "</h3>", "<break time=\"300ms\"/>" },

new String[] { "</h4>", "<break time=\"300ms\"/>" },

new String[] { "</h5>", "<break time=\"300ms\"/>" },

new String[] { "</h6>", "<break time=\"300ms\"/>" },


new String[] { " ", "<break time=\"400ms\"/>" },


});

            }
            else
            {
                edtReadMe.Text =
this.ssmlConversor.RemoveString(edtReadMe.Text, new String[] { "<b>",
"</b>", "<big>", "</big>",
                    "<em>", "</em>", "<h1>", "</h1>", "<i>", "</i>",
"<small>", "</small>", "<strong>", "</strong>", "<title>",
                    "</title>", "<br>", "</tr>", "</td>", "</u>", "<li>",
" ", "</li>", "</h2>", "</h3>", "</h4>", "</h5>", "</h6>" });
            }

            /*
            edtReadme.Text = this.removeBetween(edtReadme.Text, "<title>",
"</title>");
```

```
            edtReadme.Text = this.removeBetween(edtReadme.Text, new
String[][] {    new String[] { "<title>", "</title>" },

});
            */
            edtReadMe.Text = this.ssmlConversor.RemoveString(edtReadMe.Text,
new String[] { "<center>", "</center>", "<dl>", "<dt>", "<dd>",
                "</dl>", "</dt>", "</dd>", "</font>", "</form>",
                "<menu>", "</menu>", "</ol>", "</marquee>", "</table>",
"</th>", "<u>", "</ul>", "</html>", "<head>",
                "</head>", "<body>", "</body>", "<label>", "</label>",
"</div>", "<p>", "</p>"   });

            if (pApply)
            {
                edtReadMe.Text =
this.ssmlConversor.RemoveLink(edtReadMe.Text);
                edtReadMe.Text =
this.ssmlConversor.RemoveInput(edtReadMe.Text);
                edtReadMe.Text =
this.ssmlConversor.RemoveImage(edtReadMe.Text);
            }
            else
            {
                edtReadMe.Text =
this.ssmlConversor.RemoveUntil(edtReadMe.Text, new String[][] { new String[]
{ "<a", ">" },

new String[] { "<input", ">" },

new String[] { "<img", ">" }
                                                                    });
                edtReadMe.Text =
this.ssmlConversor.RemoveString(edtReadMe.Text, new String[] { "</a>" });
            }

            edtReadMe.Text =
this.ssmlConversor.InsertHeader(edtReadMe.Text);
            edtReadMe.Text = this.ssmlConversor.InsertFooter(edtReadMe.Text,
pApply);

        }

        private void btnSSML_Click(object sender, EventArgs e)
        {
            this.edtReadMe.Text =
this.ssmlConversor.ConvertSsml(this.edtReadMe.Text, cbAlgoritm.Checked);

            //this.ApplyAlgoritm(cbAlgoritm.Checked);
        }

        private void btnOpen_Click(object sender, EventArgs e)
        {
            String linha;
            StreamReader SR = new StreamReader(fileName);
            linha = SR.ReadLine();
            this.edtReadMe.Text = "";

            while (linha != null)
            {
```

```
                edtReadMe.Text += linha.ToLowerInvariant() + "";

                linha = SR.ReadLine();
            }
            this.btnLoad.Visible = true;
            this.btnOpen.Visible = false;
            SR.Close();
        }

        private void btnLeia_Click(object sender, EventArgs e)
        {
            this.Speak(edtReadMe.Text);
        }

        private void Speak(String pTexto)
        {
            SpeechSynthesizer oFala = new SpeechSynthesizer();
            PromptBuilder savePrompt = new PromptBuilder();
            //PromptBuilder savePrompt2 = new PromptBuilder();

            //savePrompt.AppendSsml(edtReadme.Text);
            //oFala.Speak(edtReadme.Text);

            //savePrompt2.AppendText(edtReadme.Text.Trim());
            //savePrompt2.ToXml();

            //string sp = "<speak version=\"1.0\"
xmlns=\"http://www.w3.org/2001/10/synthesis\" xml:lang=\"en-US\">gino
<break/><break/><break/> gino</speak>";
            StringReader sr = new StringReader(edtReadMe.Text);
            XmlReader myXmlReader = new XmlTextReader(sr);

            savePrompt.AppendSsml(myXmlReader);

//savePrompt.AppendSsml("C:\\Users\\Calebe\\Desktop\\fala.ssml");

            oFala.Rate = this.GetRate();
            oFala.Volume = this.GetVolume();

            oFala.SetOutputToNull();

            if (cbWav.Checked)
            {
                oFala.SetOutputToWaveFile(waveFile); //path + pTexto +
".wav"

                try
                {
                    oFala.Speak(savePrompt);
                }
                catch (Exception e)
                {
                    MessageBox.Show(e.ToString());
                }
                //oFala.Speak(pTexto);

            }

            if (cbSpeak.Checked)
            {
```

```
                oFala.SetOutputToDefaultAudioDevice();
                try
                {
                    oFala.Speak(savePrompt);
                }
                catch (Exception e)
                {
                    MessageBox.Show(e.ToString());
                }
                //oFala.Speak(pTexto);
            }

            if (cbWav.Checked)
            {
                this.Compress(pTexto);
            }

        }

    }
}
```

Listagem A.4: Classe Principal.Designer.cs

```
namespace Leia
{
    partial class Principal
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
```

```
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Principal));
            this.edtReadMe = new System.Windows.Forms.TextBox();
            this.btnSpeak = new System.Windows.Forms.Button();
            this.cbSpeak = new System.Windows.Forms.CheckBox();
            this.cbWav = new System.Windows.Forms.CheckBox();
            this.progressBar = new System.Windows.Forms.ProgressBar();
            this.tbRate = new System.Windows.Forms.TrackBar();
            this.lbRate = new System.Windows.Forms.Label();
            this.edtFile = new System.Windows.Forms.TextBox();
            this.btnLoad = new System.Windows.Forms.Button();
            this.btnOpen = new System.Windows.Forms.Button();
            this.btnSSML = new System.Windows.Forms.Button();
            this.lbVolume = new System.Windows.Forms.Label();
            this.tbVolume = new System.Windows.Forms.TrackBar();
            this.cbAlgoritm = new System.Windows.Forms.CheckBox();
            this.pbMain = new System.Windows.Forms.PictureBox();

((System.ComponentModel.ISupportInitialize)(this.tbRate)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.tbVolume)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pbMain)).BeginInit();
            this.SuspendLayout();
            //
            // edtReadMe
            //
            this.edtReadMe.Location = new System.Drawing.Point(16, 47);
            this.edtReadMe.Multiline = true;
            this.edtReadMe.Name = "edtReadMe";
            this.edtReadMe.ScrollBars =
System.Windows.Forms.ScrollBars.Both;
            this.edtReadMe.Size = new System.Drawing.Size(494, 181);
            this.edtReadMe.TabIndex = 0;
            //
            // btnSpeak
            //
            this.btnSpeak.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.btnSpeak.Location = new System.Drawing.Point(451, 273);
            this.btnSpeak.Name = "btnSpeak";
            this.btnSpeak.Size = new System.Drawing.Size(59, 33);
            this.btnSpeak.TabIndex = 1;
            this.btnSpeak.Text = "Leia";
            this.btnSpeak.UseVisualStyleBackColor = true;
            this.btnSpeak.Click += new
System.EventHandler(this.btnLeia_Click);
            //
            // cbSpeak
            //
            this.cbSpeak.AutoSize = true;
            this.cbSpeak.Checked = true;
            this.cbSpeak.CheckState =
System.Windows.Forms.CheckState.Checked;
            this.cbSpeak.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.cbSpeak.Location = new System.Drawing.Point(241, 459);
```

```
            this.cbSpeak.Name = "cbSpeak";
            this.cbSpeak.Size = new System.Drawing.Size(59, 24);
            this.cbSpeak.TabIndex = 6;
            this.cbSpeak.Text = "Fale";
            this.cbSpeak.UseVisualStyleBackColor = true;
            //
            // cbWav
            //
            this.cbWav.AutoSize = true;
            this.cbWav.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.cbWav.Location = new System.Drawing.Point(353, 459);
            this.cbWav.Name = "cbWav";
            this.cbWav.Size = new System.Drawing.Size(157, 24);
            this.cbWav.TabIndex = 7;
            this.cbWav.Text = "Salve WAV e MP3";
            this.cbWav.UseVisualStyleBackColor = true;
            //
            // progressBar
            //
            this.progressBar.Location = new System.Drawing.Point(241, 430);
            this.progressBar.Name = "progressBar";
            this.progressBar.Size = new System.Drawing.Size(269, 23);
            this.progressBar.TabIndex = 8;
            //
            // tbRate
            //
            this.tbRate.Location = new System.Drawing.Point(241, 339);
            this.tbRate.Minimum = -10;
            this.tbRate.Name = "tbRate";
            this.tbRate.Size = new System.Drawing.Size(269, 45);
            this.tbRate.TabIndex = 9;
            //
            // lbRate
            //
            this.lbRate.AutoSize = true;
            this.lbRate.Font = new System.Drawing.Font("Microsoft Sans
Serif", 10F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.lbRate.Location = new System.Drawing.Point(238, 323);
            this.lbRate.Name = "lbRate";
            this.lbRate.Size = new System.Drawing.Size(88, 17);
            this.lbRate.TabIndex = 10;
            this.lbRate.Text = "Velocidade";
            //
            // edtFile
            //
            this.edtFile.Location = new System.Drawing.Point(16, 16);
            this.edtFile.Name = "edtFile";
            this.edtFile.Size = new System.Drawing.Size(408, 20);
            this.edtFile.TabIndex = 11;
            //
            // btnLoad
            //
            this.btnLoad.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.btnLoad.Location = new System.Drawing.Point(430, 8);
            this.btnLoad.Name = "btnLoad";
```

```
            this.btnLoad.Size = new System.Drawing.Size(83, 33);
            this.btnLoad.TabIndex = 12;
            this.btnLoad.Text = "Carregar";
            this.btnLoad.UseVisualStyleBackColor = true;
            this.btnLoad.Click += new
System.EventHandler(this.btnLoad_Click);
            //
            // btnOpen
            //
            this.btnOpen.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.btnOpen.Location = new System.Drawing.Point(430, 9);
            this.btnOpen.Name = "btnOpen";
            this.btnOpen.Size = new System.Drawing.Size(83, 32);
            this.btnOpen.TabIndex = 13;
            this.btnOpen.Text = "Abrir";
            this.btnOpen.UseVisualStyleBackColor = true;
            this.btnOpen.Click += new
System.EventHandler(this.btnOpen_Click);
            //
            // btnSSML
            //
            this.btnSSML.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.btnSSML.Location = new System.Drawing.Point(241, 273);
            this.btnSSML.Name = "btnSSML";
            this.btnSSML.Size = new System.Drawing.Size(119, 33);
            this.btnSSML.TabIndex = 14;
            this.btnSSML.Text = "HTML->SSML";
            this.btnSSML.UseVisualStyleBackColor = true;
            this.btnSSML.Click += new
System.EventHandler(this.btnSSML_Click);
            //
            // lbVolume
            //
            this.lbVolume.AutoSize = true;
            this.lbVolume.Font = new System.Drawing.Font("Microsoft Sans
Serif", 10F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.lbVolume.Location = new System.Drawing.Point(238, 374);
            this.lbVolume.Name = "lbVolume";
            this.lbVolume.Size = new System.Drawing.Size(61, 17);
            this.lbVolume.TabIndex = 16;
            this.lbVolume.Text = "Volume";
            //
            // tbVolume
            //
            this.tbVolume.Location = new System.Drawing.Point(241, 390);
            this.tbVolume.Maximum = 100;
            this.tbVolume.Name = "tbVolume";
            this.tbVolume.Size = new System.Drawing.Size(269, 45);
            this.tbVolume.TabIndex = 15;
            this.tbVolume.Value = 100;
            //
            // cbAlgoritm
            //
            this.cbAlgoritm.AutoSize = true;
            this.cbAlgoritm.Checked = true;
```

```
            this.cbAlgoritm.CheckState =
System.Windows.Forms.CheckState.Checked;
            this.cbAlgoritm.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.cbAlgoritm.Location = new System.Drawing.Point(241, 234);
            this.cbAlgoritm.Name = "cbAlgoritm";
            this.cbAlgoritm.Size = new System.Drawing.Size(157, 24);
            this.cbAlgoritm.TabIndex = 17;
            this.cbAlgoritm.Text = "Habilitar Algoritmo";
            this.cbAlgoritm.UseVisualStyleBackColor = true;
            //
            // pbMain
            //
            this.pbMain.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
            this.pbMain.Image =
((System.Drawing.Image)(resources.GetObject("pbMain.Image")));
            this.pbMain.InitialImage =
((System.Drawing.Image)(resources.GetObject("pbMain.InitialImage")));
            this.pbMain.Location = new System.Drawing.Point(16, 234);
            this.pbMain.Name = "pbMain";
            this.pbMain.Size = new System.Drawing.Size(216, 249);
            this.pbMain.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pbMain.TabIndex = 18;
            this.pbMain.TabStop = false;
            //
            // uPrincipal
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(525, 498);
            this.Controls.Add(this.progressBar);
            this.Controls.Add(this.pbMain);
            this.Controls.Add(this.cbAlgoritm);
            this.Controls.Add(this.btnLoad);
            this.Controls.Add(this.lbVolume);
            this.Controls.Add(this.tbVolume);
            this.Controls.Add(this.btnSSML);
            this.Controls.Add(this.btnOpen);
            this.Controls.Add(this.edtFile);
            this.Controls.Add(this.lbRate);
            this.Controls.Add(this.tbRate);
            this.Controls.Add(this.cbWav);
            this.Controls.Add(this.cbSpeak);
            this.Controls.Add(this.btnSpeak);
            this.Controls.Add(this.edtReadMe);
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.MaximizeBox = false;
            this.Name = "uPrincipal";
            this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Scooby Doo 1.0.4.1";

((System.ComponentModel.ISupportInitialize)(this.tbRate)).EndInit();
```

```
((System.ComponentModel.ISupportInitialize)(this.tbVolume)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pbMain)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.TextBox edtReadMe;
        private System.Windows.Forms.Button btnSpeak;
        private System.Windows.Forms.CheckBox cbSpeak;
        private System.Windows.Forms.CheckBox cbWav;
        private System.Windows.Forms.ProgressBar progressBar;
        private System.Windows.Forms.TrackBar tbRate;
        private System.Windows.Forms.Label lbRate;
        private System.Windows.Forms.TextBox edtFile;
        private System.Windows.Forms.Button btnLoad;
        private System.Windows.Forms.Button btnOpen;
        private System.Windows.Forms.Button btnSSML;
        private System.Windows.Forms.Label lbVolume;
        private System.Windows.Forms.TrackBar tbVolume;
        private System.Windows.Forms.CheckBox cbAlgoritm;
        private System.Windows.Forms.PictureBox pbMain;
    }
}
```

Listagem A.5: Classe Mp3Writer.cs

```
using System;
using System.IO;
using Yeti.Lame;
using Yeti.MMedia;
using WaveLib;

namespace Yeti.MMedia.Mp3
{
  /// <summary>
  /// Convert PCM audio data to PCM format
  /// The data received through the method write is assumed as PCM audio
data.
  /// This data is converted to MP3 format and written to the result stream.
  /// <seealso cref="yeti.mmedia.utils.AudioFileWriter"/>
  /// <seealso cref="yeti.Lame"/>
  /// </summary>
  public class Mp3Writer :  AudioWriter
  {
    private bool closed = false;
    private BE_CONFIG m_Mp3Config = null;
    private uint m_hLameStream = 0;
    private uint m_InputSamples = 0;
    private uint m_OutBufferSize = 0;
    private byte[] m_InBuffer = null;
```

```
    private int m_InBufferPos = 0;
    private byte[] m_OutBuffer = null;

    /// <summary>
    /// Create a Mp3Writer with the default MP3 format
    /// </summary>
    /// <param name="Output">Stream that will hold the MP3 resulting
data</param>
    /// <param name="InputDataFormat">PCM format of input data</param>
    public Mp3Writer(Stream Output, WaveFormat InputDataFormat)
      :this(Output, InputDataFormat, new BE_CONFIG(InputDataFormat))
    {
    }

    /// <summary>
    /// Create a Mp3Writer with specific MP3 format
    /// </summary>
    /// <param name="Output">Stream that will hold the MP3 resulting
data</param>
    /// <param name="cfg">Writer Config</param>
    public Mp3Writer(Stream Output, Mp3WriterConfig cfg)
      :this(Output, cfg.Format, cfg.Mp3Config)
    {
    }

    /// <summary>
    /// Create a Mp3Writer with specific MP3 format
    /// </summary>
    /// <param name="Output">Stream that will hold the MP3 resulting
data</param>
    /// <param name="InputDataFormat">PCM format of input data</param>
    /// <param name="Mp3Config">Desired MP3 config</param>
    public Mp3Writer(Stream Output, WaveFormat InputDataFormat, BE_CONFIG
Mp3Config)
      :base(Output, InputDataFormat)
    {
      try
      {
        m_Mp3Config = Mp3Config;
        uint LameResult = Lame_encDll.beInitStream(m_Mp3Config, ref
m_InputSamples, ref m_OutBufferSize, ref m_hLameStream);
        if ( LameResult != Lame_encDll.BE_ERR_SUCCESSFUL)
        {
          throw new
ApplicationException(string.Format("Lame_encDll.beInitStream failed with the
error code {0}", LameResult));
        }
        m_InBuffer = new byte[m_InputSamples*2]; //Input buffer is expected
as short[]
        m_OutBuffer = new byte[m_OutBufferSize];
      }
      catch
      {
        base.Close();
        throw;
      }
    }

    /// <summary>
    /// MP3 Config of final data
```

```
    /// </summary>
    public BE_CONFIG Mp3Config
    {
      get
      {
        return m_Mp3Config;
      }
    }

    protected override int GetOptimalBufferSize()
    {
      return m_InBuffer.Length;
    }

    public override void Close()
    {
      if (!closed)
      {
        try
        {
          uint EncodedSize = 0;
          if ( m_InBufferPos > 0)
          {
            if ( Lame_encDll.EncodeChunk(m_hLameStream, m_InBuffer, 0,
(uint)m_InBufferPos, m_OutBuffer, ref EncodedSize) ==
Lame_encDll.BE_ERR_SUCCESSFUL )
            {
              if ( EncodedSize > 0)
              {
                base.Write(m_OutBuffer, 0, (int)EncodedSize);
              }
            }
          }
          EncodedSize = 0;
          if (Lame_encDll.beDeinitStream(m_hLameStream, m_OutBuffer, ref
EncodedSize) == Lame_encDll.BE_ERR_SUCCESSFUL )
          {
            if ( EncodedSize > 0)
            {
              base.Write(m_OutBuffer, 0, (int)EncodedSize);
            }
          }
        }
        finally
        {
          Lame_encDll.beCloseStream(m_hLameStream);
        }
      }
      closed = true;
      base.Close ();
    }


    /// <summary>
    /// Send to the compressor an array of bytes.
    /// </summary>
    /// <param name="buffer">Input buffer</param>
    /// <param name="index">Start position</param>
```

```
    /// <param name="count">Bytes to process. The optimal size, to avoid
buffer copy, is a multiple of <see
cref="yeti.mmedia.utils.AudioFileWriter.OptimalBufferSize"/></param>
    public override void Write(byte[] buffer, int index, int count)
    {
      int ToCopy = 0;
      uint EncodedSize = 0;
      uint LameResult;
      while (count > 0)
      {
        if ( m_InBufferPos > 0 )
        {
          ToCopy = Math.Min(count, m_InBuffer.Length - m_InBufferPos);
          Buffer.BlockCopy(buffer, index, m_InBuffer, m_InBufferPos,
ToCopy);
          m_InBufferPos += ToCopy;
          index += ToCopy;
          count -= ToCopy;
          if (m_InBufferPos >= m_InBuffer.Length)
          {
            m_InBufferPos = 0;
            if ( (LameResult=Lame_encDll.EncodeChunk(m_hLameStream,
m_InBuffer, m_OutBuffer, ref EncodedSize)) == Lame_encDll.BE_ERR_SUCCESSFUL
)
            {
              if ( EncodedSize > 0)
              {
                base.Write(m_OutBuffer, 0, (int)EncodedSize);
              }
            }
            else
            {
              throw new
ApplicationException(string.Format("Lame_encDll.EncodeChunk failed with the
error code {0}", LameResult));
            }
          }
        }
        else
        {
          if (count >= m_InBuffer.Length)
          {
            if ( (LameResult=Lame_encDll.EncodeChunk(m_hLameStream, buffer,
index, (uint)m_InBuffer.Length, m_OutBuffer, ref EncodedSize)) ==
Lame_encDll.BE_ERR_SUCCESSFUL )
            {
              if ( EncodedSize > 0)
              {
                base.Write(m_OutBuffer, 0, (int)EncodedSize);
              }
            }
            else
            {
              throw new
ApplicationException(string.Format("Lame_encDll.EncodeChunk failed with the
error code {0}", LameResult));
            }
            count -= m_InBuffer.Length;
            index += m_InBuffer.Length;
          }
```

```
          else
          {
            Buffer.BlockCopy(buffer, index, m_InBuffer, 0, count);
            m_InBufferPos = count;
            index += count;
            count = 0;
          }
        }
      }
    }

    /// <summary>
    /// Send to the compressor an array of bytes.
    /// </summary>
    /// <param name="buffer">The optimal size, to avoid buffer copy, is a
multiple of <see
cref="yeti.mmedia.utils.AudioFileWriter.OptimalBufferSize"/></param>
    public override void Write(byte[] buffer)
    {
      this.Write (buffer, 0, buffer.Length);
    }

    protected override AudioWriterConfig GetWriterConfig()
    {
      return new Mp3WriterConfig(m_InputDataFormat, Mp3Config);
    }
  }
```

Listagem A.6: Classe Mp3WriterConfig.cs

```
using System;
using System.Runtime.Serialization;
using Yeti.MMedia;
using WaveLib;

namespace Yeti.MMedia.Mp3
{
      /// <summary>
      /// Config information for MP3 writer
      /// </summary>
  [Serializable]
      public class Mp3WriterConfig : Yeti.MMedia.AudioWriterConfig
      {
    private Lame.BE_CONFIG m_BeConfig;

    protected Mp3WriterConfig(SerializationInfo info, StreamingContext
context)
      :base(info, context)
    {
      m_BeConfig = (Lame.BE_CONFIG)info.GetValue("BE_CONFIG",
typeof(Lame.BE_CONFIG));
    }
```

```
    public Mp3WriterConfig(WaveFormat InFormat, Lame.BE_CONFIG beconfig)
      :base(InFormat)
    {
      m_BeConfig = beconfig;
    }

    public Mp3WriterConfig(WaveFormat InFormat)
      :this(InFormat, new Lame.BE_CONFIG(InFormat))
    {
    }

    public Mp3WriterConfig()
      :this(new WaveLib.WaveFormat(44100, 16, 2))
            {
            }

    public override void
GetObjectData(System.Runtime.Serialization.SerializationInfo info,
System.Runtime.Serialization.StreamingContext context)
    {
      base.GetObjectData(info, context);
      info.AddValue("BE_CONFIG", m_BeConfig, m_BeConfig.GetType());
    }

    public Lame.BE_CONFIG Mp3Config
    {
      get
      {
        return m_BeConfig;
      }
      set
      {
        m_BeConfig = value;
      }
    }
  }
}
```

Anexo B Tabela: Classses da System.Speech.Synthesis

A tabela abaixo traz as classes encontradas na API para sintetização da Microsoft juntamente com uma breve descrição.

Tabela B.1: Principais recursos do namespace System.Speech.Synthesis

Classes	Descrição
BookmarkReachedEventArgs	Returns data from the [E:System.Speech.Synthesis.SpeechSynthesizer. BookmarkReached] event.
FilePrompt	Represents a prompt spoken from a file.
InstalledVoice	Represents an installed Voice object.
PhonemeReachedEventArgs	Returns data from the SpeechSynthesizer.PhonemeReached event.
Prompt	Plays a prompt from text or from a PromptBuilder.
PromptBuilder	Creates an empty Prompt object and provides methods for adding content.
PromptEventArgs	Represents the base class for SpeechSynthesizer EventArgs classes.
PromptStyle	Defines a style of prompting that consists of settings for emphasis, rate, and volume.
SpeakCompletedEventArgs	Returns notification from the SpeechSynthesizer.SpeakCompleted event.
SpeakProgressEventArgs	Returns data from the SpeechSynthesizer.SpeakProgress event.
SpeakStartedEventArgs	Returns notification from the SpeechSynthesizer.SpeakStarted event.
SpeechSynthesizer	Supports the production of speech and DTMF output.
StateChangedEventArgs	Returns data from the SpeechSynthesizer.StateChanged event.
VisemeReachedEventArgs	Returns data from the VisemeReached event.
VoiceChangeEventArgs	Returns data from the VoiceChange event.
VoiceInfo	Represents a text-to-speech (TTS) voice.

Anexo C Tabela: Atributos SSML, SML e SAPI

A tabela abaixo foi extraída de (PIRKER H, 2002) e faz uma comparação entre atributos considerando as três principais tecnologias de reconhecimento de voz assim como possíveis observações.

Tabela C.1: Sumário de atributos em SSML, SML e SAPI

Funcionalidade	**SSML**	**SML**	**SAPI**	**Observações**
Root element	<speak>	N.A.	<sapi>	
Defining language	“lang” attribute for <speak> <sentence> and <paragraph>	N.A.	<lang>-element or “lang”-attribute of <voice> element	
Structuring of text	<paragraph> <sentence>	<paragraph>	N.A.	<vhml:paragraph> is not part of SML proper, but available in VHML
Specifying how a certain content is to be interpreted. Attributes are, e.g., *email, number, ordinal*	<say-as>	<say-as>	<context>	
Specify, that contained text is to be spelled out (e.g. “USA”)	<say-as type=spellout>		<spell>	
Provide part of speech information	N.A	N.A	<PartOfSp>. Values are: “Unknown”, “Noun”, “Verb”, “Modifier”, “Function”,	

			“Interjection ”	
Provide phonetic pronunciation	<phoneme>	<phoneme>	<pron>	
Indicate that a specified text substitutes another	<sub>	N.A.	N.A.	
Specify/change the voice used	<voice> Attributes are lang, gender, age, variant, name	<voice>	<voice>	
Mark content as emphasized	<emphasis> Attribute: level [Values: “strong”, “moderate”, “none”, “reduced”]	<emphasizesyllable> extends SSML with Attribute: affect [Values “pitch”, “duration”, “both”] and target [the phoneme to be emphasized]	<emph>	In SAPI only <EMPH> available w/o further differentiation. In SML the syllable and optionally the prosodic means to signal emphasis and the exact position of the phoneme to be emphasized can be specified.
Control of pausing/prosodic boundaries.	<break> Attribute: size [Values: “none”, “small”, “medium”, “large”] Attribute: time (optional): duration of pause	<break>	<silence msec=100> Inserts silence of 100ms length	In SAPI only length of pause specified – no further differentiation
Control prosody.	<prosody> Attributes: pitch, contour,	<prosody>	<pitch> <rate><speed> <volume>	SSML allows for a finer grained control

	range, rate, duration, volume			of prosody. Equivalents to “contour” and “duration” are missing entirely in SAPI
Insert arbitrary audio file (e.g., recorded speech, music)	<audio>	<embed>	NOT AVAILABLE	<vhml:embed> is not part of SML proper but of VHML – it allows for embedding of foreign filetypes in general
Specify emotion of speaker	N.A.		N.A.	
Place a marker in the text: An event will be issued by the synthesizer when reaching the mark	<mark>	<mark>	<bookmark>	<ssml:mark> can have content, <sapi:bookmark> has to be empty

Anexo D Listagem: Template inicial para conversão do HTML

Apresentamos um exemplo inicial da forma que concebemos algumas tags do HTML para serem utilizadas durante a conversão para SSML.

Listagem D.1: Modelo para conversão do HTML

Retirar entre:

<!—e -->
<embed e >
<bgsound e >
A principio, retirar entre
<form e >
<hr e >
<select> e </select>
<li e >, porem colocar break após o <li
<ol e >
<marquee e >
<meta e >
<p e >
<table e >
<td e >
<tr e >
<div e >
<th e >
<ul e >

Ênfase em:

 e
<big> e </big>
<EM> E </EM>
<h1> e </h1>
 e
<small> e </small>, ênfase fraca
 e , ênfase forte
<title> e </title>

Break após:

, , <li, </TR>, </td>
</u>

Retirar Tags

, , <DL>,<DT>, <DD>,</DL>,</DT>, </DD>,, </form>,
<h2>,<h3>,<h4>,<h5>,<h6>,</h2>,</h3>,</h4>,</h5>,</h6>,<menu>, </menu>,, </marquee>,
</table>, </th>, <u>,

Ler como:

<A HREF="http://www.yourdomain.com">Visit Our Site</A>
Apos o <A, procure o href e leia entre as aspas, precedido do texto, LINK PARA. Apagar o resto até >
Colocar um break após o final em </A>

<INPUT name="Name" value="" size="10">
Ler como ~Campo ~ ler o name= e retirar o resto.

<IMG SRC="Earth.gif" WIDTH="41" HEIGHT="41" BORDER="0" ALT="a sentence about your web site">
Apos o <img, procure o src e leia entre as aspas, precedido do texto, imagem. Apagar o resto até >

Anexo E Artigo:Text-to-Speech

Text-To-Speech:
Sintetizador de Voz para Documentos como Ferramenta de Apoio a Deficientes Visuais

Calebe Augusto dos Santos / Gino Dornelles

calebes@gmail.com / ginnusd@gmail.com

Curso de Sistemas de Informação
UFSC / CTC / INE

Florianópolis
2008

Resumo

O desenvolvimento de software não deve apenas se preocupar com o domínio e a aplicação de novas tecnologias, deve também e principalmente dotar os diferentes tipos de usuários de mecanismos para utilizar diferentes aplicações. Sabendo que existem pessoas com deficiências visuais em todo o mundo assim como um contingente elevado no Brasil, este público deve ser dotado de formas a incluí-los socialmente através da inclusão digital. Para isso, fazemos uso das chamadas Speech Technologies, tecnologias empregadas na leitura ou reconhecimento de voz. Especificamente na leitura, que é o cerne deste trabalho, e sua sintetização através da utilização de interfaces e linguagens próprias para este emprego. Dentro das Speech Tecnologies, temos as Speech Synthesis (ou popularmente TTS), encarregadas da criação da fala, sua sintetização. Esta sintetização utiliza mecanismos de geração de prosódia e fonemas, portanto pedem um estudo do idioma que se pretende adotar.

Abstract

The development of software should not only be concerned with the field and application of new technologies, must also and mainly provide the different types of mechanisms for users to use different applications. Knowing that there are people with visual disabilities around the world as well as a large contingent in Brazil, the public should be provided with ways to incorporate them socially through digital inclusion. For this reason, we do use of so-called Speech Technologies, technologies employed in reading or speech recognition. Specifically in reading, which is the core of this work, and its synthesis through the use of interfaces and languages suitable for this job. Within the Speech Technologies, we have the Speech Synthesis (or popularly TTS), responsible for the creation of speech, its synthesis. This synthesis uses mechanisms to generate phonemes and prosody, so ask a study of the language that you want to adopt.

Palavras-chave: TTS (Text-To-Speech), DOTNET FRAMEWORK, SAPI (Speech Application Programming Interface), SSML (Speech Synthesis Markup Language).

1 Introdução

A idéia levada a cabo na execução deste projeto veio por duas vertentes, de um lado tínhamos a identificação de público especial dotado com características próprias (falta total ou parcial de visão) que precisa de condições criadas visando sua inserção na sociedade. Esta inserção abrangendo a vida estudantil, acadêmica, profissional etc. Por outro lado, tínhamos também a possibilidade de utilizar ferramentas de ponta no que se refere a sintetização de voz, pois a UFSC fez uma parceria com a Microsoft disponibilizando a utilização de sua suíte para desenvolvimento de aplicações (Visual Studio 2008).

Analisando alguns sistemas de TTS, ou seja, ferramentas empregadas na sintetização de voz, a exemplo do Narrator encontrado no sistema operacional Windows XP, percebe-se certa carência em alguns aspectos.

Primeiro uma certa privação de liberdade para o usuário optar pelo tipo de leitura a ser empregada. Sabemos que o processo de leitura é variável entre os leitores, alguns lêem mais rapidamente, outros se demoram em alguns itens, pro exemplo, esta liberdade é negada por muitos TTSs.

Outro senão é quanto ao escopo de leitura, a grande maioria restringe-se a ler diretamente da caixa de texto e somente. Possibilitar leituras em outros documentos garante mais flexibilidade e contribui e muito para o portador de deficiência visual se inserir no meio didático, acompanhar suas aulas em casa, por exemplo.

O objetivo principal, tendo como análise inicial o estado da arte na área de sintetizadores de voz, foi criar uma aplicação que realiza-se a sintetização de forma a diminuir GAP semântico entre aquilo que o TTS lê (o texto original) e aquilo que chega aos ouvidos do usuário (a leitura sintetizada). Diminuir o ruído incutido na mensagem de forma a acrescentar maior qualidade no processo.

De posse do ambiente de desenvolvimento, utilizamos o DOTNET FRAMEWORK 3.5 e a interface para utilizar as Speech Technologies chamada de SAPI. A SAPI em suas últimas versões permite a utilização de uma linguagem de marcação (XML) para síntese de voz. Esta linguagem baseada em XML provê mecanismos para dotar o programador de liberdade a fim de aplicar uma sintetização mais personalizada. Esta personalização contribui para diminuir o GAP semântico e diferenciar a aplicação das demais existentes.

2 Speech Synthesis

Dentro das Speech Technologies, temos a sintetização a cargo da Speech Synthesis ou TTS (Text-to-Speech).

Um Text-To-Speech (TTS) é, portanto um sintetizador que deverá ser capaz de ler qualquer texto em voz alta, se este texto foi introduzido diretamente no computador por um operador ou digitalizado e submetido a um sistema de reconhecimento óptico de caracteres (OCR) [1]. A aplicação desenvolvida se utiliza dos dois meios, tanto leitura através de caixas de texto como em documentos (no caso, formato HTML).

No TTS, temos, internamente, dois módulos claramente distintos, que requerem para a sua realização uma metodologia e conhecimento de base distintos: o processamento lingüístico-prosódico e o processamento acústico [2].

O objetivo do processamento lingüístico-prosódico é determinar, a partir da entrada de informação necessária para proporcionar ao processamento acústico, dados que lhe permitam gerar uma fala o mais natural possível. Estes dois tipos de informação são conhecidos como informação segmental e informação supra-segmental [3].

A informação segmental está associada aos sons elementares que compõe a mensagem. Para cada língua existe um conjunto limitado de sons base ideais que permitem produzir, quando corretamente combinados, todas as particularidades da fala nessa língua. São criadas representações abstratas denominadas fonemas e variáveis entre as línguas [3].

A informação supra-segmental está associada à prosódia. Reflete tanto elementos lingüísticos (frases, pausas, acentuação, etc.), como elementos não lingüísticos. Muitos autores [2] consideram a chave para se obter naturalidade na fala sintetizada.

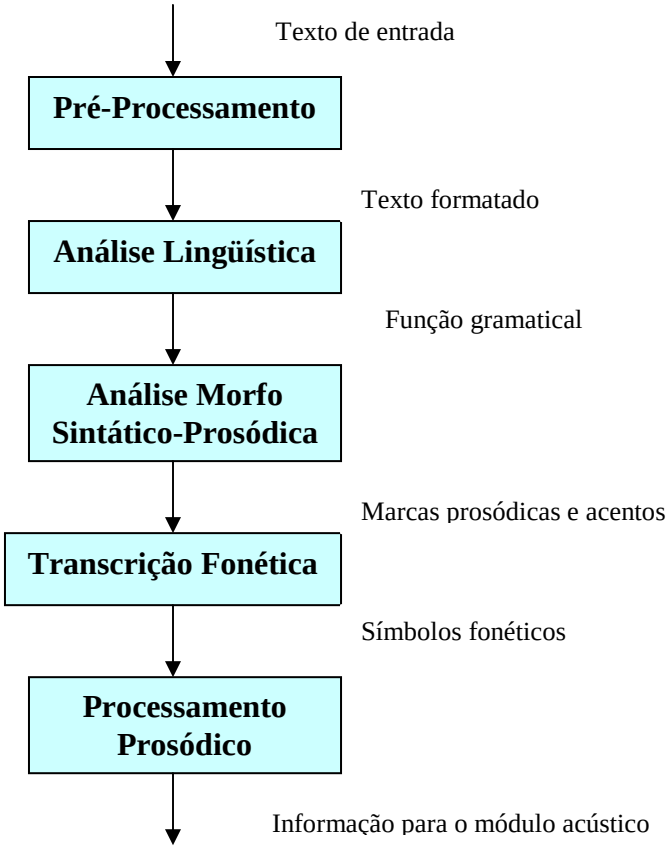


Figura 1 Diferentes tarefas do processamento lingüístico prosódico

A figura 1 apresenta as etapas sucessivas desde o momento de entrada de texto (seja digitado ou através de um OCR) até o momento em este passa para o processamento acústico, onde será submetido para o ouvinte.

Uma das formas de conseguir bons resultados no processo de sintetização seria acrescentar uma forma de aplicar mais qualidade na informação supra-segmental, ou seja, utilizar um algoritmo que possa aplicar regras de sintetização definidas pelo programador para determinados trechos do texto, realçando-os e aumento a compreensão final por parte do deficiente visual.

3 SAPI

Para poder usufruir das tecnologias de sintetização, utilizamos uma interface criada pela Microsoft em 1.995 e que já apresenta várias versões. Trata-se da SAPI (Speech Application Programming Interface), é parte integrante da Speech SDK que por sua vez integra o sistema operacional Windows.

Atualmente está na família 5 versão 5.3, versão esta utilizada no projeto e que permite a utilização da linguagem de marcação SSML, peça fundamental para tratar a vocalização em textos assim como dota as aplicações de maior independência e melhor desempenho, tanto na síntese quanto no reconhecimento de voz.

Embora a SAPI não seja uma linguagem de marcação, ela oferece algumas funcionalidades para a pronúncia na forma de XML [4], porém existem outras tecnologias mais específicas e mais poderosas para desempenhar esta função.

4 SSML

A SSML (Speech Synthesis Markup Language) é uma especificação da W3C (World Wide Web Consortium) e foi desenhada para fornecer uma linguagem de marcas mais rica, baseada em XML para que seja possível a criação de aplicações de TTS mais dinâmicas e poderosas.

A regra essencial para a elaboração desta linguagem de marcas é a de estabelecer mecanismos que permitam aos autores de conteúdos sintetizáveis controlarem aspectos do diálogo como a pronúncia, o volume, a ênfase de determinadas palavras ou frases nas diferentes plataformas [5].

A figura 2 apresenta os doze elementos encontrados na SSML, uma breve descrição de cada um e os atributos utilizados par aplicar os controles.

Elemento	Descrição	Atributos
<speak>	Elemento raiz	lang
<paragraph>e <sentence>	Representam as estruturas internas dos textos	
<say-as>	Informações sobre o tipo de texto	type
<phoneme>	Pronúncia fonética	alphabet
<sub>	Sintetiza os substitutos para o texto contido em um atributo “alias”	alias
<voice>	Especifica a voz utilizada	lang,gender,age, name, variant
<emphasis>	Ênfase no texto falado	level
<break>	Pausa entre as leituras	size, time
<prosody>	Pronúncia das palavras	pitch,contour,range, rate,duration, volume
<audio>	Incorpora arquivos de áudio	src
<mark>	Marcador de texto	name

Figura 2 Elementos, descrição e atributos da SSML

5 SCOOBY DOO

De posse das tecnologias vistas anteriormente, partimos para a execução do projeto criando uma aplicação batizada de Scooby Doo que atendesse aos principais requisitos encontrados nos sistemas de TTS (leitura, gravação, controles de volume e taxa de leitura). Além disso, que pudesse fazer usufruto de uma mais elaborada forma de leitura, onde o usuário pudesse perceber diferenças entre diferentes partes do texto, que pudesse optar por formas diferentes de leitura e que também tivesse a disposição sintetização em outros formatos.

Como base para a criação da aplicação, tomamos o TTS Narrator que já vem incluso no Windows XP da Microsoft, percebemos algumas restrições da ferramenta, principalmente a falta de controle dado ao usuário, que fica a mercê apenas de textos que são digitados, sem poder contar com um domínio maior neste sentido. Também notamos que não havia a possibilidade de gravar a leitura efetuada, assim como a sintetização de textos longos tornava-se um processo enfadonho e pouco perceptível para o usuário.

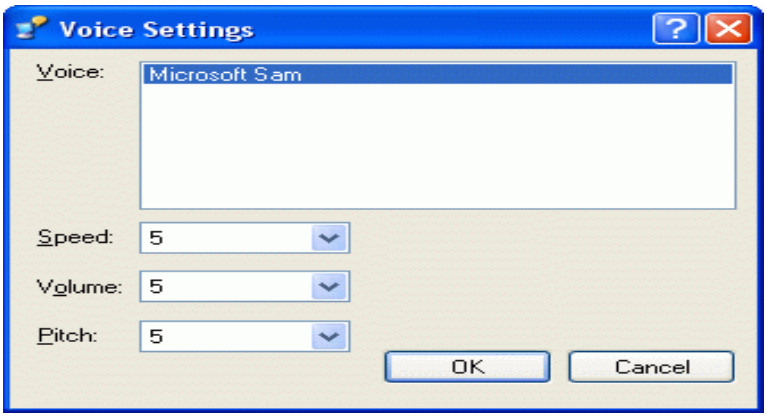


Figura 3 Narrator no Windows XP

Com já mencionado, fizemos uso do DOTNET FRAMEWORK da Microsoft através da suíte Visual Studio 2008 e com a linguagem de programação C# (C Sharp).

Com o C Sharp, pudemos utilizar o namespace System.Speech, este provendo o uso de várias classes já predefinidas para sintetização de voz, ainda utilizamos o namespace Yeti para fazer a conversão para o formato MP3 durante o salvamento do arquivo de leitura.

5.1 Requisitos da Aplicação

Analisando o projeto em função das suas funcionalidades, chegamos em uma lista de requisitos pretendidos inicialmente para o TTS Scoody Doo:

- A aplicação deve executar a leitura de páginas em formato HTML, de forma que o usuário possa compreender as palavras e também ter uma noção da estrutura do documento, noção esta oriunda da ênfase dada a certos elementos presentes numa página HTML (imagens, tabelas, links, quebra de linha, etc.);

- A aplicação deve fornecer suporte audível para elementos não textuais, tais como imagens, links e campos. Informações pertinentes no universo de origem HTML;

- Deve ser possível através de um arquivo .HTML armazenado, fazer a sua carga na aplicação;

- A aplicação deve salvar a leitura do documento nos formatos de áudio (mp3 e wave) para posteriormente ser ouvida;

- A aplicação deve oferecer ao usuário a escolha da leitura simples do texto ou ainda com o algoritmo de inferência de informação não textual.

5.2 O Algoritmo

O processo de sintetização, ou seja, a criação da voz digitalizada se dá por concatenação sintética, onde a SAPI utiliza trechos de voz gravadas para compor as sentenças. A aplicação executa a leitura de documentos no formato HTML, escolhido pela grande popularidade e possibilidade de leitura ampla.

Para realizar a leitura usando a SAPI junto à SSML e poder usufruir de todos os seus atributos, inicialmente foi preciso desenvolver um algoritmo que realizasse duas etapas: limpeza do HTML e conversão para SSML.

A limpeza do HTML se deve ao fato de excluir as TAGS que não elementos dispensáveis nas páginas, todavia, foi preciso tratar casos especiais de TAGS que continha texto em seu espoco, estas foram classificadas como TAGS com restrições e foram tratadas dentro do algoritmo. Algumas TAGS (<title></title>,,<big></big>,<h1></h1>) foram destacadas por pedir ênfase em seu texto interno, em função do contexto dentro de uma página HTML, como o caso da TAG para o título do documento.

Outras TAGS (
, , <li, </tr>, </td>) foram selecionadas para apresentar uma parada na leitura facilitando o entendimento, exemplo desse processo se encontra numa quebra de linha ou dentro de uma tabela.

Também foram tratadas TAGS referentes a links, elementos presentes na maioria das páginas e que facilitam a navegação pelos documentos. Assim o leitor pode saber quando a leitura estiver passando por um link. As TAGS para imagens foram selecionadas para ler o nome da imagem dentro da TAG <img>, para que o usuário possa criar de uma forma mais verossímil seu modelo mental acerca da página lida.

Utilizando a Aplicação

O TTS Scooby Doo apresenta uma interface apresentada na figura 4. Inicialmente é selecionado um arquivo HTML através do botão Carregar; este mesmo botão (agora transformado em Abrir) vai mostrar na tela a página selecionada com todas as TAGS. Com o algoritmo habilitado, clicando-se em HTML -> SSML acrescentamos os elementos desta linguagem ao texto, pode-se perceber pela figura 4 alguns dos elementos como o break, emphasis e o prosody. Percebe-se que alguns destes, como o emphasis, tem seu atributo level com o valor “strong” (nível mais forte) no título de um capítulo, assim como um break após a leitura do título do capítulo, estas mudança, embora sutis, fazem muita diferença no resultado final e demandam muito tempo em testes para se conseguir um resultado interessante. Ao chegar ao fim da página foi acrescida leitura de “end page” para situar o usuário do término do documento.

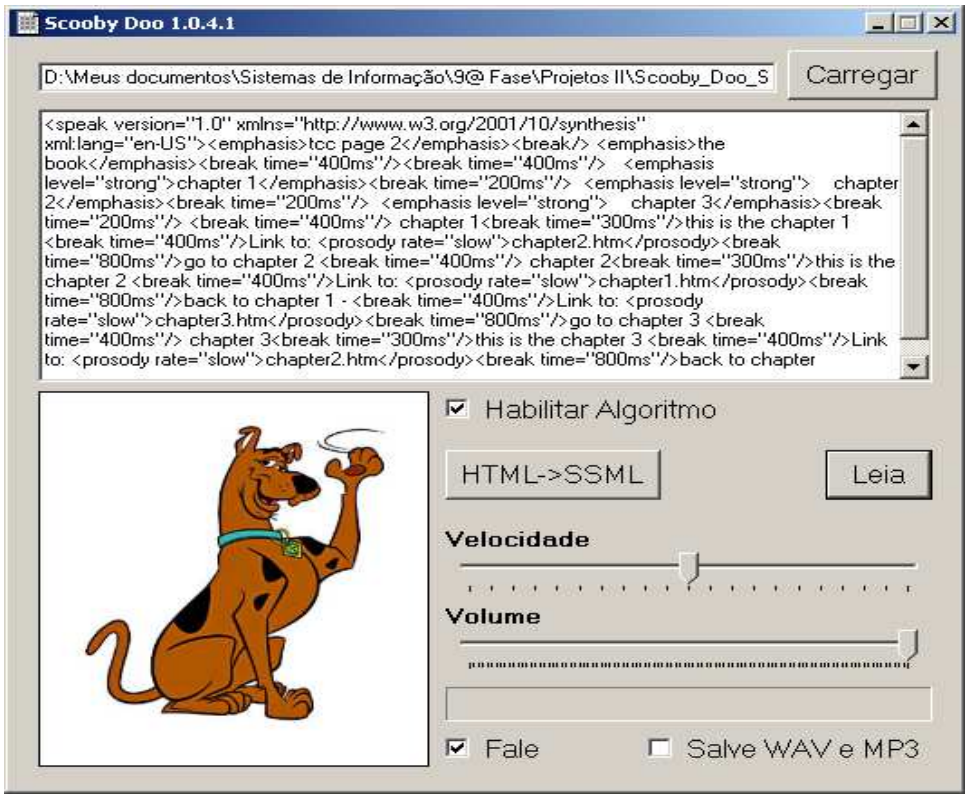


Figura 4 TTS Scooby Doo executando leitura de página HTML

Ainda na interface, temos opções para aumentar a velocidade e o volume da leitura, bem como salvar a página para leitura posterior. A opção Fale habilita a leitura.

Comparativo

Com o intuito de comparar algumas ferramentas de TTS, verificamos certos requisitos encontrados e sua presença ou ausência.

Tabela 1 Quadro comparativo entre as ferramentas de TTS

	Ferramentas de TTS		
Características	Narrator	Festival	Scooby Doo
Velocidade de leitura	V	V	V
Volume da leitura	V	V	V
Ler HTML	X	X	V
Habilitar tipo de leitura	X	X	V
Salvar arquivo	X	V	V
Gratuita	V	V	V

Legenda:
V Presença da característica
X Ausência da característica

Analisando a tabela 1, vemos que todas as ferramentas soa gratuitas, embora o Narrator mostra-se muito simples no atendimento dos requisitos, a opção de gravar arquivo encontra-se apenas no Festival e no Scooby Doo, e este é o único que apresenta leitura em HTML e define uma opção para o usuário escolher como se dará a leitura.

6 Conclusões

Poder contribuir de alguma forma para facilitar a inclusão de portadores de deficiência visual é muito gratificante. Criar um TTS que execute uma leitura com qualidade e se destaque dos demais foi o objetivo principal e alcançamos este intento.

Existem alguns aspectos a serem considerados, um deles é que a ferramenta desenvolvida serve como apoio, como seu próprio nome ressalva, demandando que o portador de deficiência tenha um auxiliar para utilizar a aplicação.

Alguns acréscimos podem ser realizados, entre eles, a inclusão de uma API na SAPI para o português do Brasil, ficando o Scooby Doo mais orientado à realidade brasileira. Outra alteração se dá o acréscimo de outros formatos de arquivo para leitura (PDF) a fim de tornar o aprendizado mais amplo.

Referências Bibliográficas

[1] DUTOIT, THIERRY. High-quality text-to-speech synthesis: an overview. Faculte Polytechnique de Mons, TCTS Lab. Acessado em: http://lands.let.ru.nl/TSpublic/strik/onderwijs/fonetiek/literatuur/Dutoit.pdf . 2008.

[2] LÓPEZ, EDUARDO. Estudio de Técnicas de Processado Linguistico y Acústico para Sistemas de Conversión Texto-Voz en Espanhol baseado em Concatenatión de Unidades. Tesis Doctoral. Universidad Politécnica de Madrid, 1993.

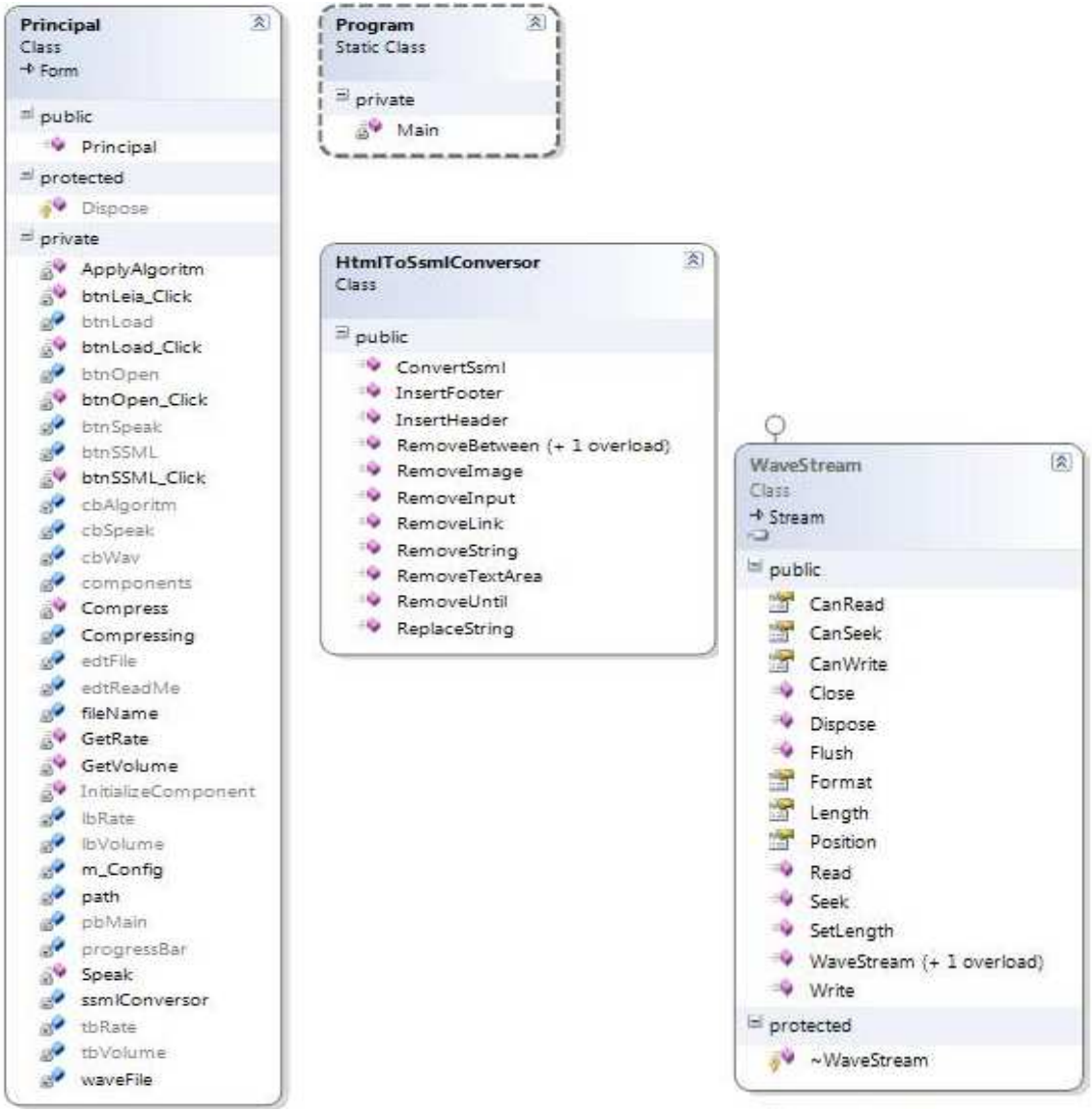
[3] OLASZY, GÁBOR; NÉMETH, GÉZA. THE MULTIVOX – Multilingual text-to-speech converter, Talking Machines, 1992.

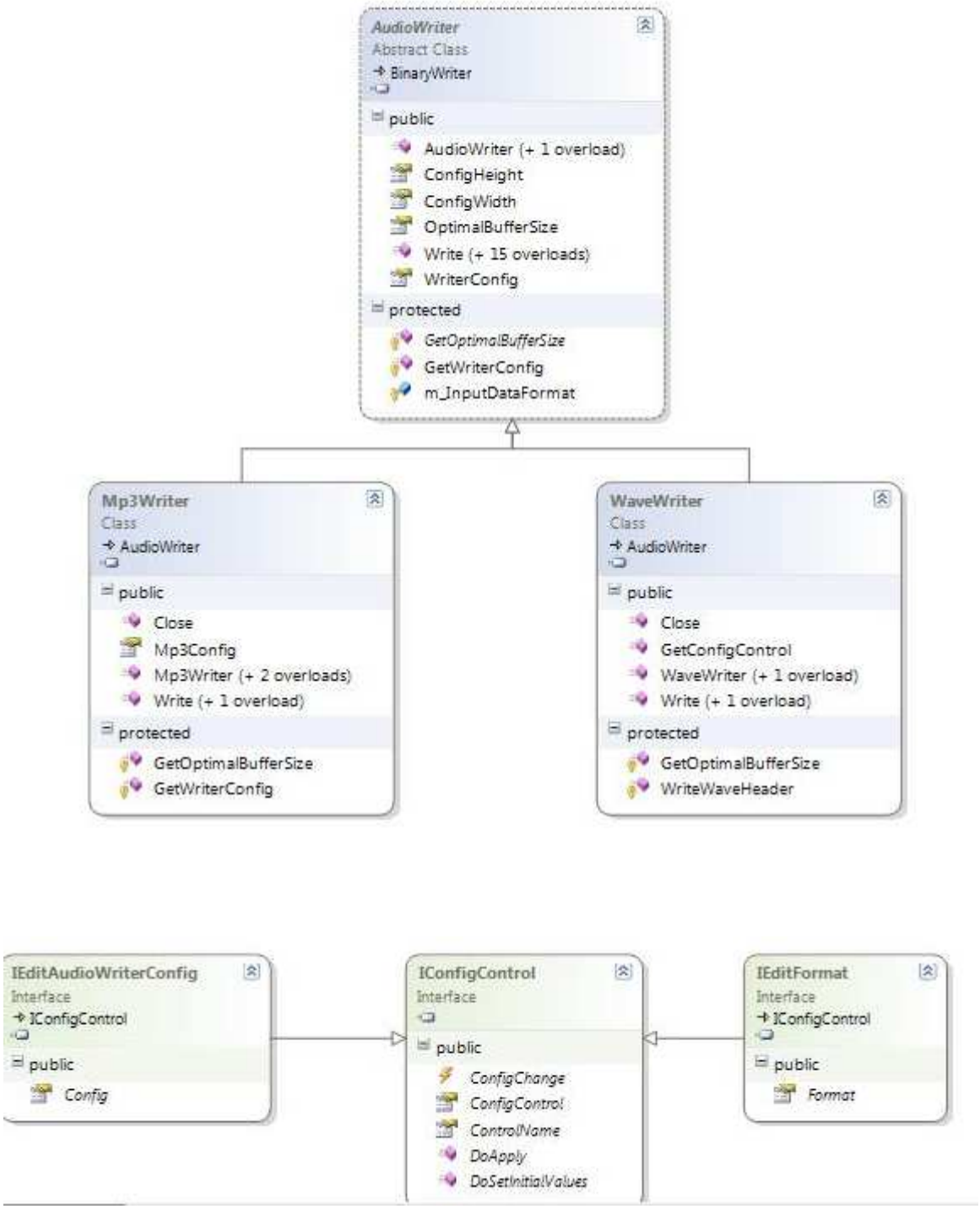
[4] PIRKER, H; KRENN B. B. Assessment of Assessment of Markup Languages for Avatars, Multimedia and Multimodal Systems, NECA-project, Deliverable D9c. 2002.

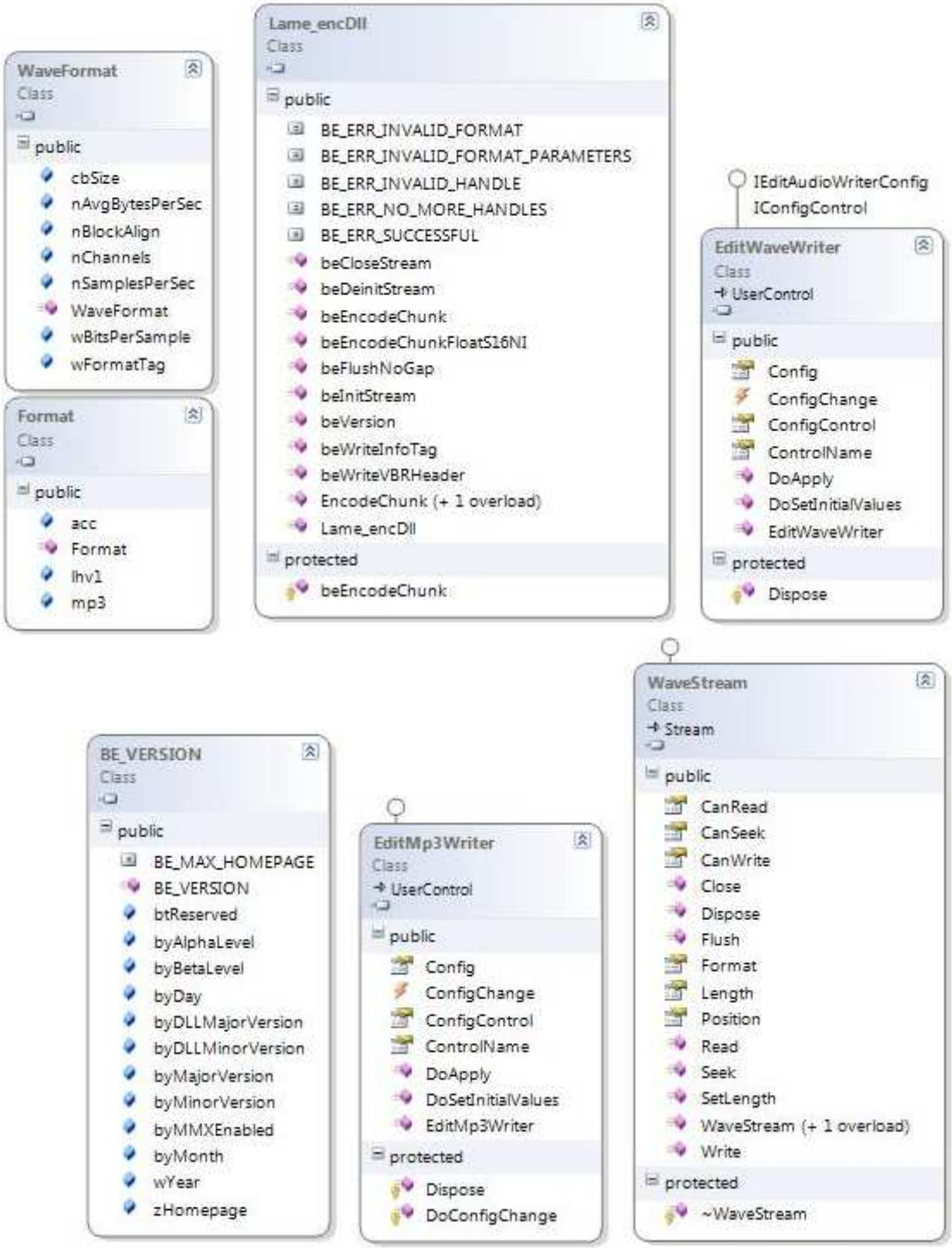
[5] SILVA, TELMO EDUARDO; OLIVEIRA, JOSÉ LUÍS. Interfaces de voz para a Web. 2001. Acessado em: http://www.deetc.isel.ipl.pt/jetc05/CCTE02/papers/finais/jetc/313.pdf. 2008.

Anexo F Diagrama de Classes: Scooby-Doo Speaker

Para apresentar a modelagem estática da aplicação dispomos de um diagrama de classes com sua estrutura interna.




AudioWriter
Abstract Class
BinaryWriter
public
AudioWriter (+ 1 overload)
ConfigHeight
ConfigWidth
OptimalBufferSize
Write (+ 15 overloads)
WriterConfig
protected
GetOptimalBufferSize
GetWriterConfig
m_InputDataFormat
Mp3Writer
Class
AudioWriter
public
Close
Mp3Config
Mp3Writer (+ 2 overloads)
Write (+ 1 overload)
protected
GetOptimalBufferSize
GetWriterConfig
WaveWriter
Class
AudioWriter
public
Close
GetConfigControl
WaveWriter (+ 1 overload)
Write (+ 1 overload)
protected
GetOptimalBufferSize
WriteWaveHeader
IEditAudioWriterConfig
Interface
IConfigControl
public
Config
IConfigControl
Interface
public
ConfigChange
ConfigControl
ControlName
DoApply
DoSetInitialValues
IEditFormat
Interface
IConfigControl
public
Format


WaveFormat
Class
public
cbSize
nAvgBytesPerSec
nBlockAlign
nChannels
nSamplesPerSec
WaveFormat
wBitsPerSample
wFormatTag
Format
Class
public
acc
Format
lhv1
mp3
Lame_encDll
Class
public
BE_ERR_INVALID_FORMAT
BE_ERR_INVALID_FORMAT_PARAMETERS
BE_ERR_INVALID_HANDLE
BE_ERR_NO_MORE_HANDLES
BE_ERR_SUCCESSFUL
beCloseStream
beDeinitStream
beEncodeChunk
beEncodeChunkFloatS16NI
beFlushNoGap
beInitStream
beVersion
beWriteInfoTag
beWriteVBRHeader
EncodeChunk (+ 1 overload)
Lame_encDll
protected
beEncodeChunk
IEditAudioWriterConfig
IConfigControl
EditWaveWriter
Class
UserControl
public
Config
ConfigChange
ConfigControl
ControlName
DoApply
DoSetInitialValues
EditWaveWriter
protected
Dispose
BE_VERSION
Class
public
BE_MAX_HOMEPAGE
BE_VERSION
btReserved
byAlphaLevel
byBetaLevel
byDay
byDLLMajorVersion
byDLLMinorVersion
byMajorVersion
byMinorVersion
byMMXEnabled
byMonth
wYear
zHomepage
EditMp3Writer
Class
UserControl
public
Config
ConfigChange
ConfigControl
ControlName
DoApply
DoSetInitialValues
EditMp3Writer
protected
Dispose
DoConfigChange
WaveStream
Class
Stream
public
CanRead
CanSeek
CanWrite
Close
Dispose
Flush
Format
Length
Position
Read
Seek
SetLength
WaveStream (+ 1 overload)
Write
protected
~WaveStream

Referências Bibliográficas

ANDRADE , ANA ELIS NOGUEIRA DE MAGALHÃES; APPA , RENIRA CIRELLI. Fonologia: Prosódia e Ortoépia - Um estudo com base nas transcrições de conversações em telemarketing entre pessoas jurídicas (BankBoston). http://www.letramagna.com/Fonoestilistica.pdf, 2007.

BLACK, ALAN W; TAYLOR, PAUL; CALEY, RICHARD. The Festival Speech Synthesis System, http://fife.speech.cs.cmu.edu/festival/manual-1.4.1/festival-1.4.1.ps.gz, 1999.

BORGES, ANTONIO. Entidades Parceiras do Projeto Dosvox. http://intervox.nce.ufrj.br/dosvox/parceiros.htm, 2007.

BRANCO, CAMILO, EUSTÁQUIO. As origens da língua portuguesa. http://www.eduquenet.net/origemlingua.htm, 2008.

COOK, STEPHEN. Speech Recognition Howto. http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/pdf/Speech-Recognition-HOWTO.pdf , 2002.

DUTOIT, THIERRY. An Introduction to Text-To-Speech Synthesis. 2007.

EFRON, M. Efron visual acuity test. Columbia, SC: Marvin Efron (West Columbia Optometric Group, P. O. Box 4045, West Columbia, SC 29171), 1980.

FESTIVAL. http://www.cstr.ed.ac.uk/projects/festival/onlinedemo.html, 2007.

IBGE. Instituto Brasileiro de Geografia e Estatística. http://www.ibge.gov.br/home/, 2007.

JOHNSON, R. E.; FOOTE, B. Designing reusable classes. Journal of Object-Oriented Programming, v. 1, n. 2, p. 23–35, 1988.

LUCENA, PAULA SALGADO. Experimentos com Sincronização de Áudio e Vídeo. Pontifícia Universidade Católica do Rio de Janeiro, RJ, 2001.

M, MATTSSON. ObjectOriented Frameworks: a survey of methodological issues. PhD thesis. Dept. of Computer Science, UCK. 1996.

MATTAR, M. E. Eficiência com as diferenças, RITS, La Insígnia, maio, 2003, http://www.lainsignia.org/2003/mayo/soc_024.htm, 2008.

MICROSOFT. Windows Vista. http://msdn.microsoft.com/en-us/magazine/cc163663.aspx, 2008.

MICROSOFT SYSTEMS JOURNAL, Talk to Your Computer and Have It Answer Back with the Microsoft Speech API. http://www.microsoft.com , 1996.
MSS. Microsoft Speech Server, http://www.microsoft.com/speech/, 2008.

NERI, M. C. Retratos das pessoas com deficiências ao longo dos tempos, disponível http://www.saci.org.br , Seção ARTIGOS, junho, 2003.

OMS. Organização Mundial da Saúde. 2008.

PIRKER, H; KRENN B. B. Assessment of Assessment of Markup Languages for Avatars, Multimedia and Multimodal Systems, NECA-project, Deliverable D9c. 2002.

Portal da Retina. http://www.drvisao.com.br, 2008.

PROJETO DOSVOX. http://intervox.nce.ufrj.br/dosvox/, 2008.

REDE SACI. http://saci.org.br/, 2007.

SAPI. Microsoft speech technology, http://www.microsoft.com/speech , 2007.

SAUVÉ, JACQUES PHILIPPE. O Que é um Framework, http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm, 2008

SILVA, RICARDO PEREIRA E. Análise e Projeto de Sistemas. UFSC. 2004.

SILVA, TELMO EDUARDO; OLIVEIRA, JOSÉ LUÍS. Interfaces de voz para a Web. 2001.

SSML. http://www.w3.org/TR/speech-synthesis/#edef_emphasis, 2008.

TAYLOR P., ISARD A. SSML: A speech synthesis markup language, Speech Communication, (21), pp.123-133, 1997.

TORRES, ELISABETH FÁTIMA; MAZZONI, ALBERTO ANGEL. Conteúdos digitais multimedia: o foco na acessibilidade e usabilidade, Brasilia, maio/agosto, 2004.

W3C. http://www.w3.org/html/, 2008.

WEINSTEIN, E. Master Thesis - SpeechBuilder: Faciliting Spoken Dialogue System Development. Massachusetts Institute of Technology, 2001.