

PROJETO COMUNICA

Relatório Final

Convênio de Cooperação Geral FINEP – SEBRAE
Plano de trabalho nº 72/2007

Março de 2012

PORTO ALEGRE, RS – CEI/UFRGS – FAURGS

APRESENTAÇÃO

O Projeto “Comunica – Acesso a Bases de Dados pelo Telefone” foi um projeto desenvolvido em parceria, tendo como integrantes o Instituto de Informática da UFRGS e as empresas Conexum, DFL e Intext. O projeto foi financiado pelo Fundo Nacional de Desenvolvimento Científico e Tecnológico (FNDCT), no âmbito da chamada pública MCT/SEBRAE/FINEP/ Ação Transversal 04/2007 – PITCE, envolvendo também o CEI e o SEBRAE.

O objetivo do Projeto Comunica consistiu no desenvolvimento de um sistema de atendimento automático que permita o acesso por voz a uma base de dados, utilizando o telefone como meio.

Para atingir o objetivo e demonstrar uma aplicação real, fez-se uma parceria com a Federação das Associações dos Municípios do Rio Grande do Sul (FAMURS), cliente da empresa DFL. A aplicação desenvolvida é um Sistema de Perguntas e Respostas capaz de receber ligações dos representantes dos municípios para tratar as dúvidas acerca dos valores de transferência de impostos constitucionais. Dessa forma, as questões faladas são transcritas através do reconhecimento de voz e traduzidas (ou mapeadas) a um formato compreensível para acesso ao Banco de Dados da FAMURS. Após realizada a consulta, é criada uma frase de resposta, a qual é sintetizada em áudio e passada aos representantes dos municípios pelo telefone.

Dentro desse contexto, este documento contém o relatório técnico do projeto Comunica, descrevendo o seu desenvolvimento. Para tanto, são apresentados os resultados, as atividades desenvolvidas, as dificuldades encontradas e as soluções tomadas. Apresenta-se também uma visão geral da arquitetura do sistema, onde são descritos os diferentes módulos, suas funcionalidades e responsáveis.

O relatório técnico levanta uma série de contribuições relevantes do projeto, a saber:

- Uma versão de demonstração do sistema utilizada nos diferentes eventos técnico-científicos e empresariais (a lista de eventos encontra-se no Apêndice II);
- Módulo de perguntas e respostas com interface Web (sem reconhecimento ou síntese de voz). Tal produto já pode ser colocado em prática na FAMURS e poderia ser facilmente adaptado, sem o módulo de voz, para dispositivos móveis e outros contextos;
- Técnicas e ferramentas computacionais para busca de informações estruturadas em bancos de dados a partir de frases em linguagem natural, para domínio específico;
- Ontologia de domínio (FAMURS) e ferramentas para auxiliar na construção de ontologias de domínio com base em amostras de informações do domínio;
- Instalação de ferramenta experimental da DFL no ambiente da FAMURS para consulta multidimensional (OLAP), integrado ao processo de *ETL* do Comunica (descrito no capítulo 4), permitindo análise dos dados coletados;
- Produto *Expressão Direta*, já em comercialização pela Conexum, para atendimento telefônico por palavras-chave. Este sistema substitui a atual digitação de números por palavras faladas, em atendimentos telefônicos

automáticos, também chamadas de Unidade de Resposta Audível (URA). O sistema pode ser testado no telefone da Conexum: (51)3308-6239;

- Resultados alcançados nos diferentes módulos e submódulos, principalmente os que se referem a estudos de tratamento de voz e linguagem.

Apesar de ter atingido as metas estabelecidas e já poder ser colocado em prática, restam algumas melhorias a serem feitas para que o sistema torne-se totalmente comercializável, resumidas a seguir:

- Ainda são necessárias algumas melhorias na síntese, pois apesar de sintetizar todas as palavras completas, a voz ainda apresenta certas alterações de entonação durante a fala. Apesar disso, a síntese é feita com sucesso, assim como sua integração ao sistema do Projeto Comunica;
- Aperfeiçoamento do módulo de reconhecimento de voz contínua.

Esses refinamentos são justificados pela complexidade do software e suas evoluções, perdas de continuidade de bolsistas e ausência de especialistas para as carências apresentadas no decorrer da execução do projeto, além do fato de que se optou por substituir alguns componentes proprietários por componentes próprios, de código aberto.

O documento está estruturado da seguinte forma. Primeiramente apresenta-se uma visão geral do sistema, incluindo sua arquitetura e módulos. Em seguida, tais módulos são detalhados, informando-se sua evolução, tecnologia envolvida e estágio atual. Finalmente, apresentam-se algumas considerações finais e dois apêndices: o primeiro contendo informações sobre os mecanismos de gerência e coordenação de equipe utilizados, além de ferramentas de trabalho colaborativo; o segundo descrevendo a lista de eventos e feiras onde o produto foi apresentado, assim como artigos técnicos correspondentes.

1 ARQUITETURA DO SISTEMA

O sistema foi idealizado de modo a possuir diferentes módulos e componentes responsáveis por funcionalidades específicas. Visando uma arquitetura moderna e orientada a serviços, optou-se por utilizar *Web services* (WS) como tecnologia de base para a construção dos componentes. WS provê um mecanismo padrão e aberto para a interoperação entre aplicações em diferentes plataformas. Com isso, os diferentes grupos de desenvolvimento podem utilizar os componentes, *frameworks* e linguagens de programação que lhes forem mais convenientes, e mesmo assim sua integração e interação serão permitidas. Além disso, tal tecnologia permite com que os componentes sejam dinamicamente substituídos ou compostos sem que isso exija reprogramação.

A figura seguinte apresenta os principais módulos do sistema, a saber: Controlador, Diálogo, Processamento da Língua, Acesso à Base e Carga.

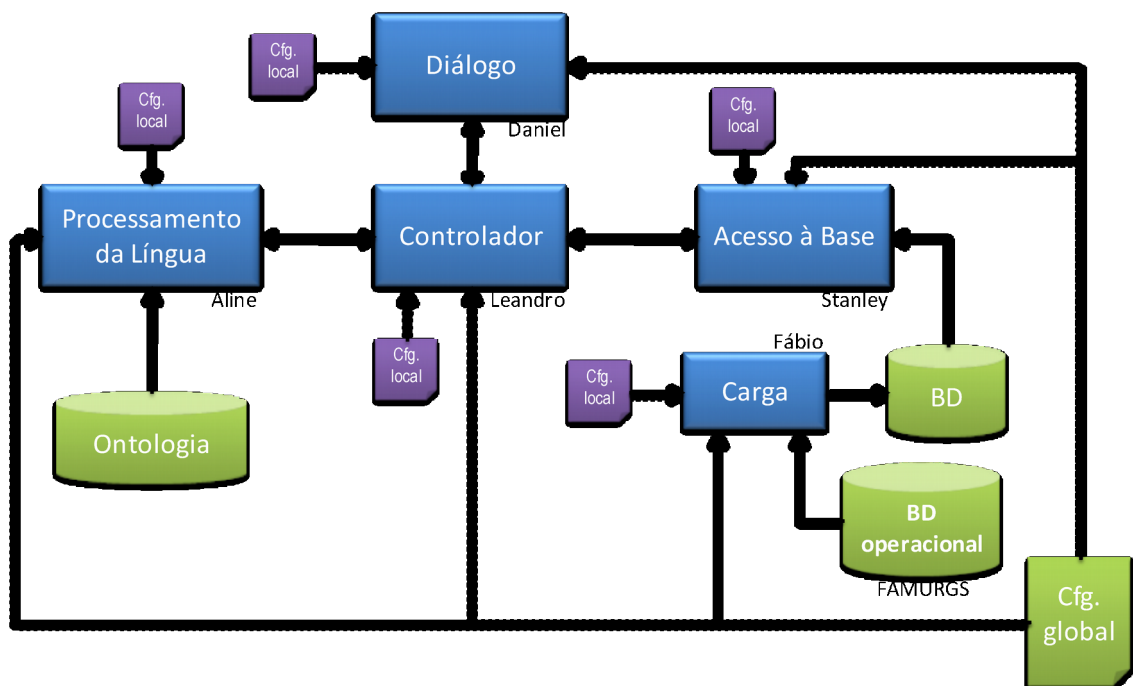


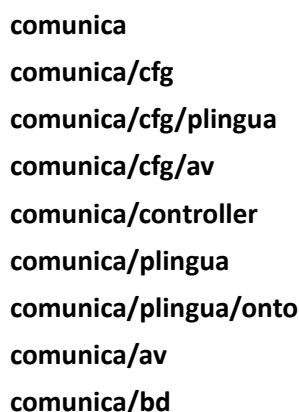
Figura 1.1 – Arquitetura Global do Sistema

Tais módulos são detalhados a seguir. Antes de apresentá-los, no entanto, cabe salientar que cada módulo pode estar em um nodo ou servidor diferente da rede. Eventualmente, dependendo da situação ou tipo de processamento, alguns módulos deverão estar na mesma máquina para evitar atrasos.

Os módulos devem estar fisicamente organizados em uma estrutura de pastas ou diretórios como descritos na Figura 1.2.

Como o sistema foi idealizado para funcionar de maneira distribuída, para que se possa rapidamente localizar os módulos dentro de uma máquina, ou para que se possa integrá-los, tal estrutura de pastas/diretórios é fortemente recomendada. Obviamente, em um ambiente distribuído, somente os arquivos e pastas do módulo em questão estarão disponíveis (mas saber-se-á onde encontrar seus arquivos).

Para que os módulos (principalmente o de diálogo e o controlador) possam se comunicar uns com os outros (e para que se possa mudar a localização dos mesmos), optou-se por definir um arquivo ou serviço de configuração global. Tal arquivo é importante para especificar a URL ou pasta onde os demais módulos estão localizados, pois sem sua existência os diferentes módulos acabam utilizando referências absolutas no código, especificadas localmente, tornando difícil a sua transposição para outros servidores.



comunica
comunica/cfg
comunica/cfg/plingua
comunica/cfg/av
comunica/controller
comunica/plingua
comunica/plingua/onto
comunica/av
comunica/bd

O diagrama mostra uma estrutura de pastas dentro de um retângulo. As pastas são listadas uma abaixo da outra: 'comunica', 'comunica/cfg', 'comunica/cfg/plingua', 'comunica/cfg/av', 'comunica/controller', 'comunica/plingua', 'comunica/plingua/onto', 'comunica/av' e 'comunica/bd'.

Figura 1.2 – Arquitetura de Pastas do Sistema

Além disso, verificou-se também a necessidade de se especificar configurações específicas para os módulos. Cada módulo tem, então, um arquivo próprio de configuração.

Logo, dentro da estrutura de diretórios propostos, a pasta cfg serve para centralizar os arquivos de configuração (global e individuais). A pasta cfg/geral armazena a URL dos diferentes componentes (onde estão instalados), indica a versão atual do XML sendo utilizado e a linguagem utilizada na interface do sistema (pt-br, us-en, etc).

Para que todos os módulos funcionem de maneira adequada, sugere-se que o módulo controlador seja o primeiro a ser ativado e que esse, ao ser iniciado, atualize suas configurações específicas e envie mensagem aos demais componentes indicando sua localização (um *script* de sincronização pode, eventualmente, replicar os arquivos de configuração nos diferentes computadores onde os módulos estiverem).

2 MÓDULO CONTROLADOR

O módulo controlador é o núcleo do sistema. Ele gerencia a comunicação (troca de dados) entre os diferentes módulos do sistema. Ele está sendo desenvolvido pelo grupo coordenado pelo prof. Dr. Leandro Krug Wives.

A primeira versão do módulo foi desenvolvida em linguagem PHP pelo bolsista Nicolas Dullius Mallmann, e focava na interação textual com o usuário (demo do sistema via Web). Os dois primeiros “demos” do sistema utilizaram tal versão e foram apresentados em conferências científicas. O papel do controlador é receber requisições do módulo de Diálogo, analisá-las e direcioná-las ao módulo de tratamento correspondente, transformando-as em um formato compatível e esperado.

Com a saída do bolsista Nicolas no início de agosto, o bolsista Lucas Magrini Rigo assumiu sua posição e alterou o código PHP para que ele se tornasse independente de interface. Uma versão que utiliza tal módulo pode ser encontrada na URL <http://www.conexum.com.br/comunica/> (opção “Demo”). A figura seguinte demonstra sua tela principal, com duas regiões importantes (A e B).

comunica

O que é Como funciona **Demo** Contato

Português [English](#)

Digite sua pergunta ...

☐ (Audio)

☐ (Audio)

...ou escolha uma das perguntas abaixo.

Quanto Agudo vai receber de ICMS em 15 de Janeiro, 2010?

Quanto Agudo vai receber de ICMS em 15 de Janeiro, 2010?

(A)

(B)

Figura 2.1 – Sistema de demonstração

A região “A” permite ao usuário escrever uma pergunta em linguagem natural. Após digitar e selecionar a opção “consulta” ou “consulta rápida”, o sistema identifica os elementos que fazem parte da consulta, com base em uma ontologia de domínio (tal processo é descrito no Capítulo 5 - Módulo de Processamento da Língua), e retorna o resultado. A diferença entre “consulta” e consulta rápida é que a primeira consiste numa demonstração passo-a-passo de todas as etapas do processo, incluindo (1) identificação dos elementos ou conceitos encontrados na frase, (2) encaminhamento para o assistente virtual, e

(3) frase resultante a ser encaminhada ao sintetizador, enquanto que a segunda opção executa todas de uma vez, mostrando somente o resultado final. Cada uma dessas etapas é realizada por um módulo diferente do sistema. Tais módulos são descritos nas suas respectivas seções. A região "B" apresenta uma lista de consultas predefinidas, envolvendo as diferentes possibilidades de consulta que o sistema oferece. Ambas as regiões foram desenvolvidas para permitir a demonstração do sistema, seja de maneira mais detalhada (objetivando eventos acadêmico-científicos) quanto sucinta (possíveis clientes e parceiros).

O próximo passo consistiu em desenvolver uma versão totalmente nova do módulo em linguagem Java. Isso porque a linguagem nos oferece maior autonomia. O novo módulo segue o modelo de software orientado a serviço, funcionando como um serviço Web (*Web service*) que permite com que as requisições ao sistema sejam feitas de qualquer lugar, utilizando o protocolo HTTP, padrão na Web. Além disso, permite com que o sistema seja totalmente desacoplado, facilitando o desenvolvimento, substituição e atualização de componentes, que podem ser desenvolvidos em qualquer linguagem.

O *Web service* (WS) de controle, feito em linguagem Java, oferece uma única funcionalidade, denominada "answerMe".

Funcionalidade	Entradas e Saídas
answerMe	<i>Entrada:</i> XML contendo a pergunta feita pelo usuário <i>Saída:</i> XML contendo a resposta à pergunta feita pelo usuário.

Com base na entrada recebida pelo método (funcionalidade) "answerMe", o WS Controle repassa a pergunta do usuário para o Módulo de Processamento da Língua (descrito no Capítulo 5). Tal módulo analisa a frase e devolve os elementos (conceitos) reconhecidos (tipo de pergunta e elementos da pergunta) caso a pergunta esteja correta e nenhuma informação esteja faltando. Com base nesse resultado, o Controlador encaminha uma consulta ao Módulo de Acesso à Base (Capítulo 4). A resposta retornada é então devolvida ao módulo chamador (normalmente o módulo de diálogo), que devolve a resposta ao usuário. Caso alguma informação esteja faltando, o Controlador retorna ao chamador um pedido de reformulação de pergunta, informando os itens faltantes.

O serviço controlador oferece ainda uma interface para testes, localizada no servidor interno da empresa Conexum, com acesso somente local.

3 MÓDULO DE DIÁLOGO

A Figura seguinte descreve os componentes do Módulo de Diálogo, cuja equipe é gerenciada pelo Dr. Daniel Nehme Müller, da empresa Conexum.

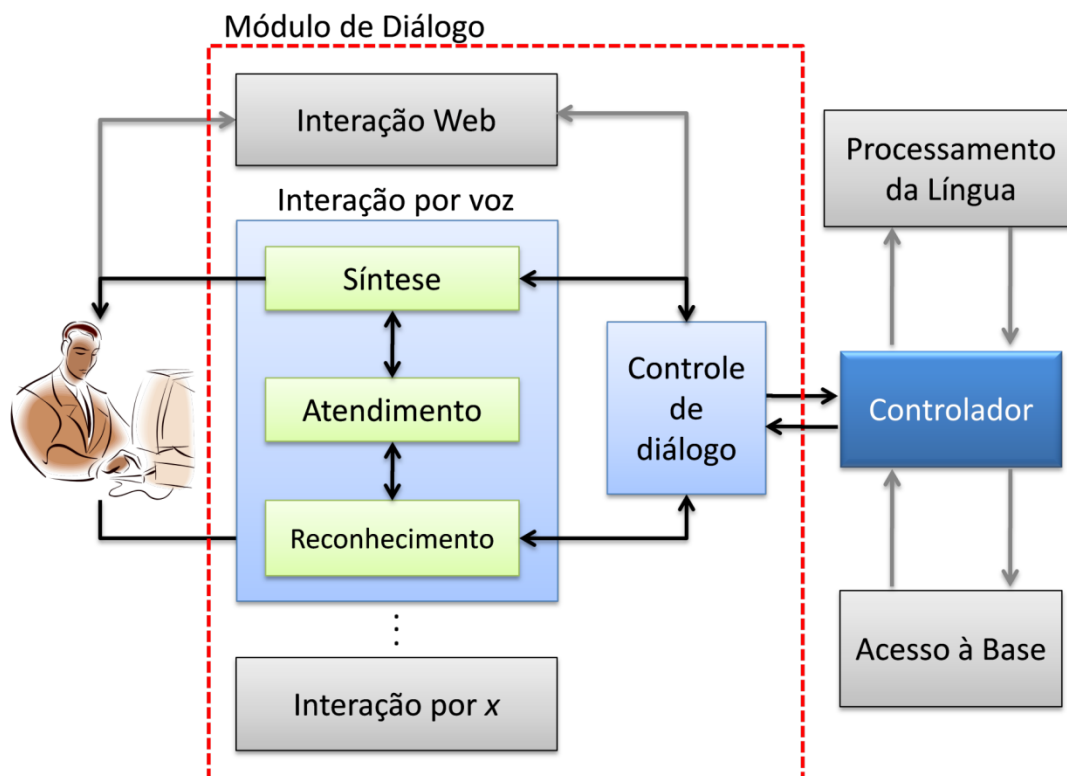


Figura 3.1 – Componentes do Módulo de Diálogo

Como pode ser observado, o módulo de diálogo é multimodal, ou seja, permite ao usuário interagir com sistema de diferentes maneiras.

O módulo é composto por um componente de Controle de Diálogo e componentes de interação diferenciados (um para cada "modo" de interação). Inicialmente a interação deve ser vocal, mas já foi implementado (para a versão de demonstração apresentada anteriormente) o módulo textual (via Web). A tecnologia sugerida para o módulo permitiria com que os componentes de interação pudessem ser substituídos dinamicamente, sem que isso exigisse reprogramação do módulo de controle de diálogo.

O Controle de Diálogo é responsável por executar alguns passos para controlar a entrada e saída das informações dentro do Módulo de Diálogo Comunica. O primeiro passo após o atendimento da chamada é acessar o arquivo de texto transcrito pelo reconhecedor, interpretá-lo e transferi-lo através de um arquivo XML via *WebService* (WS) para o Módulo de Controle. Após processamento da resposta, o Módulo de Diálogo é responsável por acessar e interpretar o XML de resposta também transferido via e WS gerando um arquivo texto para o sintetizador de fala. O sistema WS foi desenvolvido em colaboração com o bolsista Aleksandro Rosa dos Anjos, da Conexum.

O componente de Interação Vocal possui os seguintes subcomponentes: Asterisk, Reconhecimento e Síntese. O Asterisk é um software controlador de atendimentos telefônicos. O componente de interação vocal é, na verdade, uma

Unidade de Resposta Audível (URA) inteligente, ou seja, simula um atendente humano, perguntando e respondendo ao usuário utilizando linguagem natural. O módulo de reconhecimento de voz, realizado através do sistema Julius, é capaz de analisar o áudio da fala e convertê-la em texto. O sintetizador de fala realiza o processo inverso, transformando um texto recebido em áudio para o usuário. O controlador de diálogo recebe os textos gerados pelo Julius e os repassa para o módulo de controle. O módulo de síntese gera o áudio da fala descrita em um texto encaminhado pelo módulo de controle.

Esses módulos são detalhados a seguir.

3.1 Asterisk (interface de telefonia)

O bolsista Lucas Magrini Rigo foi destacado para configuração da interface de telefonia, de forma a realizar a recepção do áudio do ramal telefônico da Conexum para testes de gravação, síntese e reconhecimento de voz.

Foi criado guia de instalação Asterisk, o qual foi disponibilizado no wiki do projeto, que pode ser encontrado no servidor interno da Conexum.

Os itens do guia são:

- Instalação da placa da Digium;
- Instalação do Asterisk;
- Atualizações do linux e bibliotecas que podem ser necessárias para o dahdi;
- Instalação do dahdi opção 1;
- Instalação do dahdi opção 2;
- Configuração do dahdi para reconhecimento da placa e Asterisk.

Atualmente, está instalada a versão Asterisk 1.6 para o atendente da sala da Conexum e o gravador de chamadas em nosso servidor localizado na FAMURS (local do teste piloto). O atendente usa os arquivos de áudio da pasta /var/lib/asterisk/sounds/en, e grava os arquivos de teste em /var/lib/asterisk/sounds/. O sistema Asterisk da FAMURS foi instalado e configurado pelo bolsista Jeferson Antunes Rubert.

Para cada um dos locais de atendimento telefônico foram criados scripts de gerenciamento da linha telefônica através da placa Digium instalada nos dois servidores, gerenciadas pelo Asterisk. Maiores detalhes também foram disponibilizados na wiki do projeto.

3.2 Módulo de reconhecimento de voz

O módulo consiste em um transcritor de sinal acústico para texto. Os primeiros testes do reconhecimento de voz foram procedidos pelos membros da Conexum, Daniel Müller e Peter Neto. O sistema de reconhecimento de palavras faladas desenvolvido por Peter, que era utilizado no sistema de demonstração online (descrito no capítulo anterior), foi substituído pela conjugação dos sistemas HTK e Julius, que serão descritos a seguir.

Inicialmente planejou-se utilizar o software Coruja do projeto FalaBrasil (<http://www.laps.ufpa.br/falabrasil/>). No entanto, devidas às muitas dificuldades que ocorreram na implantação e uso de tal software, além do fato do mesmo ainda

estar em desenvolvimento e possuir pouca documentação, optou-se por utilizar o software Julius (<http://sourceforge.jp/projects/julius/>), que é open source. Outra vantagem do Julius é que já existem projetos em andamento para proporcionar sua integração com o módulo Asterisk.

O Julius funciona recebendo um stream de áudio (arquivo ou sinal do computador) e retornando sua transcrição na forma de um arquivo-texto. O programa presume um modelo acústico e um modelo linguístico sobre o qual o reconhecimento será realizado. Uma das etapas consiste, portanto, no desenvolvimento de tais modelos (acústico e linguístico).

Para desenvolver tais modelos, foram coletadas diversas frases que seguiam consultas comuns de prefeituras feitas por telefone à FAMURS, solicitando informações. Tais frases foram gravadas pela equipe, transcritas e anotadas com etiquetas fonéticas. A partir daí foi constituída uma base preliminar de voz para treinamento no sistema HTK (<http://htk.eng.cam.ac.uk/>) e posterior reconhecimento dos termos através do software Julius. O Julius, portanto, está sendo utilizado junto com modelos acústicos e linguísticos construídos através do HTK.

Para a construção do sistema acústico, foram colhidas gravações dentro de nosso escopo de trabalho através do sistema Asterisk, descrito na seção anterior. Assim, foram transcritas 470 frases, dentre as quais 167 foram utilizadas no treinamento e as demais nos testes e validação. Para as frases de treinamento também foi necessária a etiquetagem fonética para seu treinamento de monofones e trifones. No trabalho de transcrição colaboraram os bolsistas Bruno Campos, Jeferson Antunes Rubert e Lucas Magrini Rigo. Todo o processo levou 3 meses para ser concluído e o sistema de edição utilizado foi WaveSurfer 1.8.5.

A seguir tem-se uma figura que demonstra o processo de transcrição e etiquetagem de frases. A frase transcrita em palavras nesta figura é "Quem transfere o IPI".

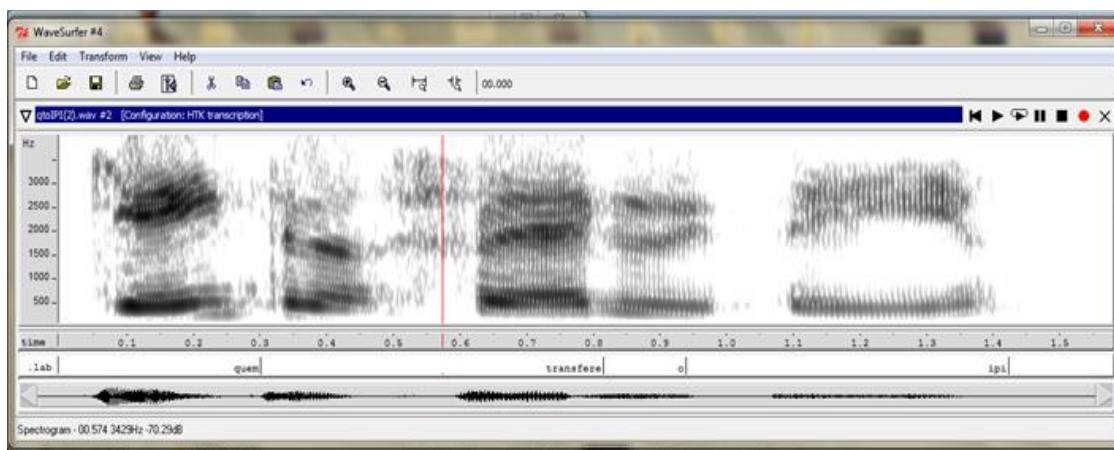


Figura 3.2 – Transcrição das frases

Paralelamente a esse processo, um servidor foi fisicamente instalado na sede da FAMURS para coletar gravações de ligações reais de prefeituras, tais gravações serão utilizadas para aprimorar o reconhecimento futuramente. Para tal foi instalado o PABX Asterisk, de forma que as gravações são feitas sem interferência do atendimento. Ao fim do dia todas elas são transferidas automaticamente para o servidor do Comunica.

Depois de realizadas as gravações das ligações reais, foram novamente realizadas transcrições de 34 diálogos, com a respectiva etiquetagem fonética. Esses diálogos estão sendo utilizados em novo projeto de modelo acústico e de linguagem para novos conjuntos de treinamento e reconhecimento de voz.

3.3 Módulo de reconhecimento contínuo

O módulo consiste em um transcritor de sinal acústico para texto, que por sua vez será lançado ao reconhecimento. Para todo o processo, foram utilizadas as orientações do Projeto Julius. O módulo foi desenvolvido por Daniel Müller e o estagiário da Conexum, Lisardo Kist.

O módulo funciona recebendo um *streaming* de áudio (arquivo ou sinal do computador) e retorna a transcrição na forma de arquivo de texto. O programa presume um modelo acústico e um modelo linguístico sobre o qual o reconhecimento será realizado.

Essa transcrição será posteriormente utilizada pelo diálogo. A interação com o módulo se dá pelo shell do Linux. O tipo do sinal de entrada (arquivo, microfone, etc.) é passado como argumento na chamada.

Utilizou-se inicialmente o software Coruja do Projeto Falabrasil, do LaPS (Laboratório de Processamento de Sinais), da Universidade Federal do Pará, que utiliza o sistema Julius, para reconhecimento. Houve muitas dificuldades até os primeiros resultados, porque tanto a versão para Linux do programa quanto os modelos acústicos eram para 32-bits, ao passo que nosso ambiente de execução era de 64-bits. O Coruja ainda está em desenvolvimento e a documentação ainda é insipiente. Houve dificuldades para se compilar e obter o programa funcionando.

Por esses e outros motivos se resolveu utilizar o próprio reconhecedor Julius, do qual o Coruja é uma interface. Outra vantagem do Julius é já existir projetos de teste para integração direta entre Julius e Asterisk.

O bolsista Arthur Carvalho Rauter desenvolveu vários modelos de gramáticas baseadas no contexto FAMURS, para reger o reconhecimento de voz do Julius em parceria com o modelo acústico construído pelo LaPS (<http://www.laps.ufpa.br/>). Diversas formas de gramáticas foram testadas, e após grandes quantidades de testes, foi concluído que um modelo de várias gramáticas independentes gerava melhores resultados. Assim, foi concebido o modelo utilizado inicialmente. O diagrama de máquina de estados que foi desenvolvido para demonstrar o correlacionamento das gramáticas utilizadas é apresentado na Figura 3.3.

Entretanto, apenas este recurso não apresentou os resultados esperados quanto ao reconhecimento de frases contínuas, então foi cogitada a possibilidade de realizar o treinamento de um modelo acústico próprio e direcionado ao contexto. A fim de facilitar o processo de treinamento, buscou-se fazer uso dos scripts utilizados pelo LaPS na criação do seu modelo, variando apenas as amostras sonoras e configurações utilizadas. O bolsista Arthur foi para outro projeto e em seu lugar a bolsista Maitê Dupont assumiu o processo da remodelagem das gramáticas e treinamento do modelo acústico específico para o projeto.

O primeiro procedimento realizado pela bolsista foi o aperfeiçoamento das sete gramáticas regulares previamente criadas para uso no programa de reconhecimento Julius. As gramáticas continham problemas de encadeamento e não forneciam os resultados esperados. As regras foram corrigidas e o dicionário

fonético de cada uma foi aprimorado. Para isso foi utilizado um dicionário fonético criado pelo LaPS.

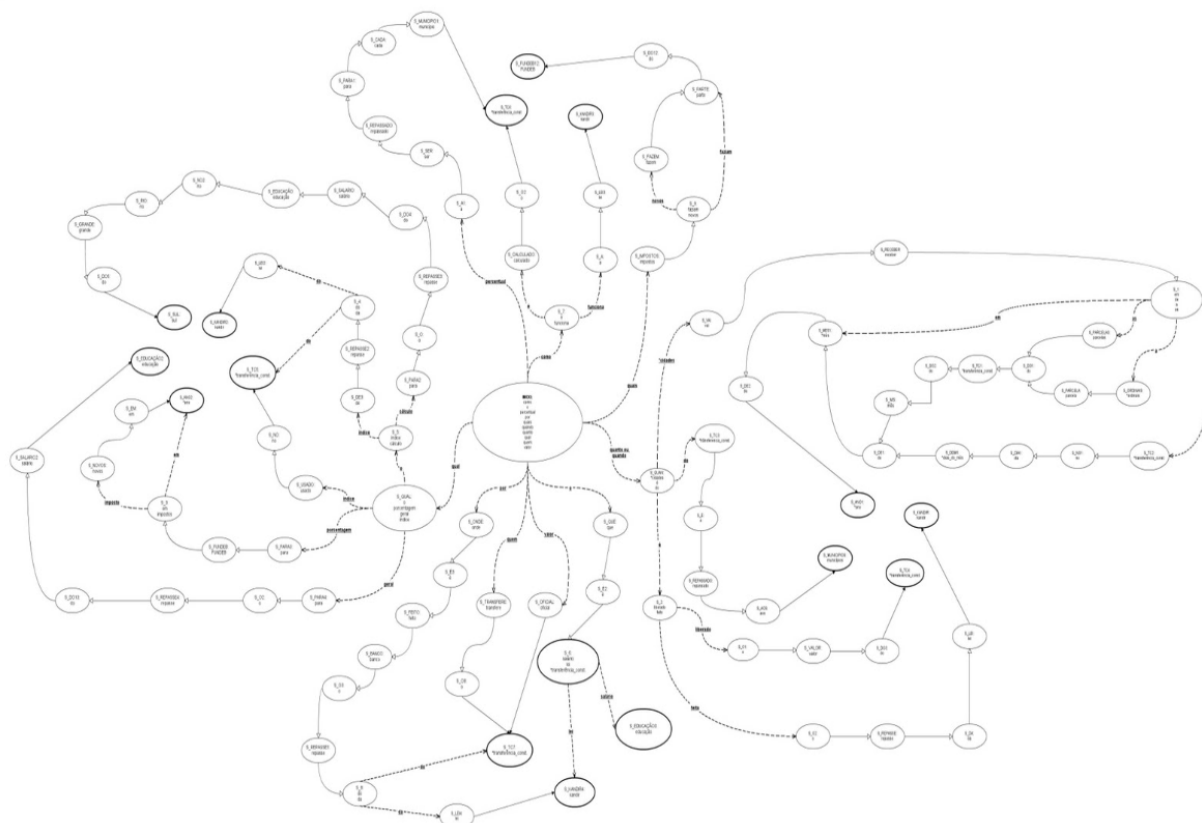


Figura 3.3 – Correlacionamento das gramáticas

Em seguida foram feitos vários testes de desempenho utilizando as novas gramáticas e os sistemas HTK e Julius junto com um modelo acústico do LAPS. As amostras usadas para teste foram gravadas na empresa Conexum pelos seus bolsistas.

O primeiro objeto de pesquisa foram as técnicas probabilísticas aplicadas à Inteligência Artificial, como as Cadeias Ocultas de Markov (ou Hidden Markov Models, HMMs). Essas técnicas consistem em aplicar processamentos matemáticos aos arquivos de entrada de forma a aproximar, através de cálculos de probabilidades, o que foi falado pelo locutor e possibilitar a transcrição desta fala. O objetivo deste estudo era criar um modelo acústico para o projeto Comunica a fim de possibilitar o reconhecimento de linguagem natural.

Para o treinamento do modelo acústico foi utilizado o software livre HTK (Hidden Markov Toolkit) para treinamento de modelos acústicos e linguísticos aplicados no Julius. Grande parte da documentação do HTK e do Julius. Informações sobre o processo de treinamento podem ser encontradas facilmente com uma busca na WEB, entretanto, quando ocorre algum erro não previsto pelo “tutorial”, informações sobre como corrigi-lo são difíceis de adquirir.

Ao longo do processo de treinamento surgiram diversos erros complicados e sem documentação alguma. Sua resolução foi quase por tentativa e erro, modificando parâmetros, formatos de arquivos, ordem das listagens de arquivos etc. Alguns erros decorriam de transcrições de áudio que eram utilizadas e que estavam com problemas de digitação ou correspondência de termos (ausência ou

excesso de palavras). Ao final do treinamento e de vários testes do processo e dos seus resultados, a conclusão foi que o treinamento de um modelo acústico para reconhecer linguagem natural precisaria de muitas horas de gravações, ao menos 15 horas de áudio transcrito, como encontrado no modelo do LaPS. Devido ao custo e tempo demandado, este ramo de pesquisa foi deixado temporariamente de lado.

O foco retornou para as gramáticas utilizadas no reconhecimento em conjunto com a ferramenta Julius. Essas gramáticas e seus respectivos dicionários fonéticos foram ainda mais refinados, pois o dicionário fonético utilizado havia sido produzido no Pará. Como a pronúncia das palavras é diferente no Sul e no Norte do Brasil (assim como em todas as outras regiões), a sequência de fonemas que estava sendo utilizada não condizia com a sequência de fonemas das palavras pronunciadas no Rio Grande do Sul.

A parametrização do Julius também foi refinada, de forma que ele conseguisse as melhores taxas de reconhecimento dentro do possível. Para isso, cada parâmetro precisou de estudo cuidadoso e de vários testes para verificar a qualidade da sua alteração. Esses testes foram realizados sobre as mesmas amostras da Conexum, e com as mesmas gramáticas, possibilitando a análise da modificação apenas dos parâmetros.

Com o Julius em funcionamento pleno dentro dos limites impostos pelas gramáticas regulares, que não são capazes de reconhecer fala contínua como as gramáticas estatísticas, o foco retornou ao treinamento dos modelos acústicos.

Modelos acústicos restritos ao reconhecimento de apenas algumas palavras dentro de um contexto específico (ex.: um modelo acústico que reconhecesse apenas nomes de municípios do RS, ou apenas meses do ano) foram criados, na hipótese de que este tipo de modelo precisaria de uma menor quantidade de gravações. Diversas formas de treinamento (diferentes números de iterações, diferentes parametrizações, etc.) sobre os mesmos conjuntos de dados foram utilizadas. A conclusão obtida foi que uma média de dez amostras de cada palavra a ser reconhecida ainda era uma quantidade insuficiente. Mais uma vez esta linha foi deixada de lado, por causa da inviabilidade de adquirir a quantidade de gravações necessárias.

Utilizando o modelo acústico do LAPS, o módulo de reconhecimento de voz restrito foi integrado ao restante do projeto. Este módulo consiste em perguntas feitas ao usuário, de forma a direcionar sua resposta, já que o reconhecimento de voz ainda não trabalha com fala contínua. Cada resposta é transcrita separadamente e depois incluída em uma frase padrão utilizada para consulta ao banco de dados.

A sequência de chamadas para reconhecimento de voz é feita através do sistema Asterisk, sendo composta de cinco perguntas:

1. Deseja saber valor ou data da transferência constitucional?
2. Qual transferência constitucional dentre FPM e Lei Kandir?
3. Qual município dentre Agudo, Bagé, Canoas, Guabiju e Pelotas?
4. Qual mês?
5. Qual ano dentre 2010 e 2011?

As respostas destas questões são compostas de forma a constituir a frase de busca que será analisada pelo restante do sistema.

3.4 Módulo de síntese de fala

O Módulo de síntese recebe um texto como entrada e transforma-o em resposta audível ao usuário. Tal módulo está em desenvolvimento e é atualmente capaz de identificar as palavras do texto, selecionar os arquivos correspondentes a elas e agregar tudo em um único arquivo a ser encaminhado à saída do sistema para que o usuário receba uma resposta audível. O próximo passo é desenvolver um módulo que seja capaz de processar fonemas e entonações de voz, tornando a fala mais natural.

O sistema de síntese foi baseado na dissertação de mestrado de André Cardon, realizado no Instituto de Informática da UFRGS, e modificado para este projeto com autorização do autor. Para trabalhar no sistema, foi destacado o bolsista Bruno Campos.

O sistema de síntese realiza a leitura de um arquivo texto, separa uma frase em sílabas e usa tais sílabas para abrir os arquivos de áudio correspondente a cada uma delas. Após isso o software concatena todos esses arquivos em um único arquivo de áudio, o qual gera o áudio da entrada, como mostram as figuras a seguir.

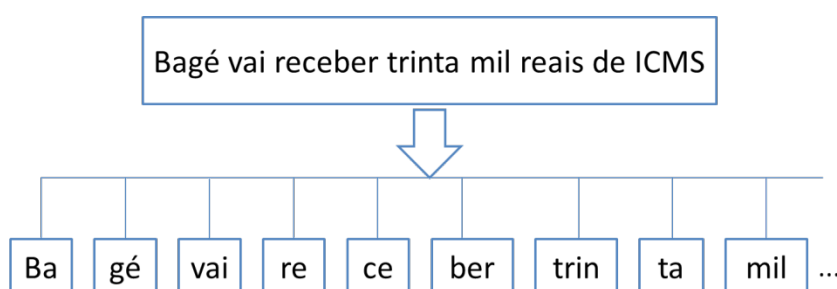


Figura 3.4 – Separação dos fonemas

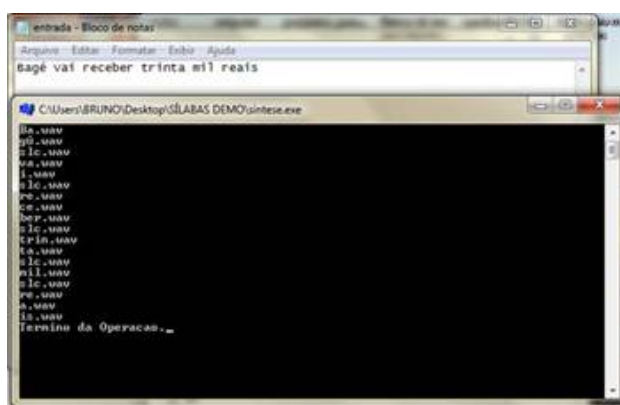


Figura 3.5 – Concatenação de arquivos de fonemas

As figuras anteriores demonstram o programa em execução. O programa lê a frase de um arquivo txt, separa ela em sílabas, também dando ênfase aos espaços em branco (assinalados como slc), e abre cada um dos arquivos wav para concatenar as sílabas em um único arquivo final, nomeado como novo arquivo wav.

Em um segundo momento do trabalho, foi feito um software com o mesmo objetivo acima, o de, a partir de uma entrada (frase), separá-la em componentes menores, porém agora em palavras, e, a partir de cada um desses componentes, usar o arquivo de áudio respectivo e, assim, finalmente, concatená-los em um arquivo final, conforme figura seguinte.

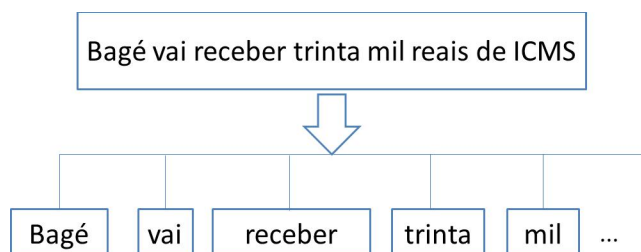


Figura 3.6 – Separação das palavras

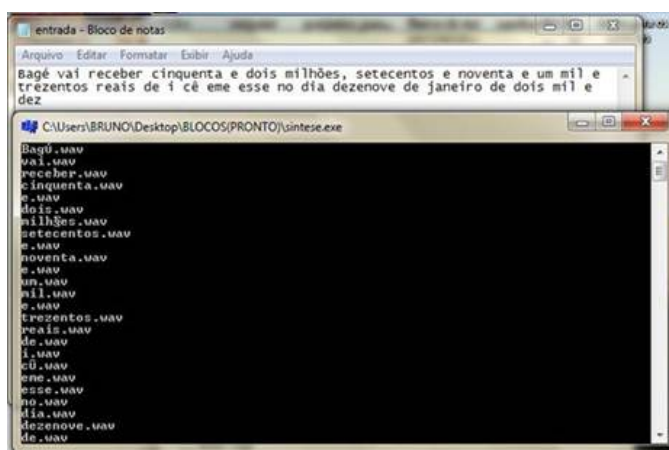


Figura 3.7 – Concatenação de arquivos de palavras

Como mostram as Figuras 3.6 e 3.7, o programa separa a frase lida do arquivo txt em palavras e usa cada arquivo de áudio correspondente a cada uma delas para juntá-las em um só arquivo, nomeado como "novo.wav".

O arquivo "novo.wav" conterá tanto na síntese por concatenação silábica tanto na síntese por concatenação de palavras o áudio final que será o que o usuário escutará na página demo do projeto comunica.

É importante deixar claro que, para que o arquivo final não tenha nenhuma falha, os arquivos WAV concatenados devem estar gravados da mesma forma e no mesmo formato. Caso contrário, pode haver divergências de som ou o arquivo final nem mesmo ser executado.

O objetivo primordial desses softwares é fazer parte de um sistema demo de perguntas e respostas para a fundação de apoio aos municípios do Rio Grande do Sul. Sua participação no projeto se resume a sintetizar as respostas para as perguntas feitas a FAMURS, no caso, ao demo que se encontra no site do comunica.

A pesquisa iniciada pelos bolsistas Arthur e Bruno, que saíram para outros projetos em fevereiro de 2011, foi continuada pelo novo bolsista Anderson Foscarini para encontrar um sintetizador de fala open source que forneça melhores resultados do que o sintetizador silábico.

Primeiramente foi trabalhado o eSpeak, software livre que é distribuído juntamente com o Ubuntu, extremamente leve e fácil de ser operado. Apesar de possuir síntese para português brasileiro, esse software não é bom o suficiente para os fins do Projeto, pois sua síntese produz uma voz robótica, de difícil compreensão. Além disso, mesmo alterando alguns parâmetros da síntese, não se chega perto de uma voz aceitável.

Uma vez descartado o eSpeak, buscou-se outra alternativa de software na internet. Foi encontrado um artigo escrito por membros do LaPS (Laboratório de Processamento de Sinais) da UFPA, que descrevia uma maneira de criar uma voz em português brasileiro para síntese utilizando o MARY (Modular Architecture for Research on speech sYnthesis), software livre escrito em Java que realiza síntese e possibilita a adição de idiomas e criação de novas vozes para a síntese, vozes essas baseadas na voz de uma pessoa. O artigo do LaPS pode ser acessado em <http://www.laps.ufpa.br/falabrasil/files/ITS2010-LaTex.pdf>.

O artigo descrevia parte do processo de criação, e o site do MARY (<http://mary.dfki.de>) possui alguns tutoriais para adição de novos idiomas ao sistema de síntese. Após análise do material, foi escolhido o MARY como base da pesquisa de síntese para o Comunica.

Antes de criar a voz, dentro do MARY deve existir a “linguagem” português brasileiro. Isso consiste de um arquivo de alofones que possui a representação de fonemas e suas características na linguagem (vogais nasais, orais, semivogais, etc). Neste trabalho foi utilizado um arquivo já disponibilizado pelo LaPS, que se baseia no padrão SAMPA (como descrito no artigo do LaPS). Além dos alofones, é necessário um dicionário de palavras com suas transcrições fonéticas no modelo apresentado nestes alofones. No Comunica foi utilizado um dicionário de mais de 7 mil palavras em português, e a maioria dessas palavras foi automaticamente traduzida para fonemas utilizando um programa disponibilizado pelo LaPS em <http://www.laps.ufpa.br/falabrasil/files>, que faz a tradução grafema-fonema. Tendo esse dicionário e os alofones, usa-se o componente *Transcription Tool* do MARY (Figura 3.8) para criar arquivos que ajudam no momento da síntese, em termos de prever transições fonéticas e composições de fonemas.

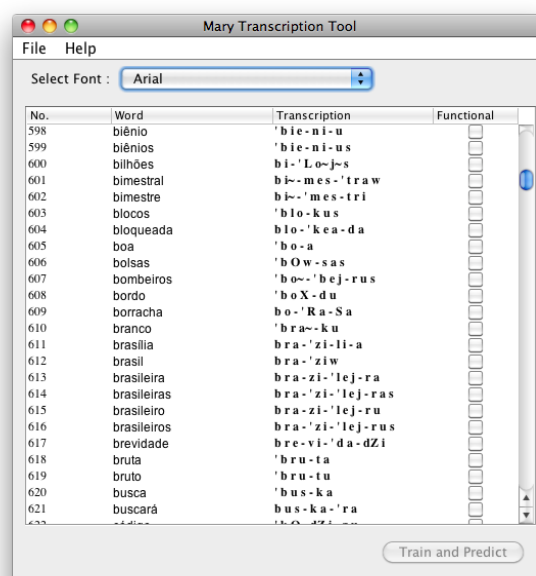


Figura 3.8 – Concatenação de arquivos de palavras

Era preciso então criar uma voz em português brasileiro para o MARY. Dentre os requisitos mais gerais estão programas que o MARY utiliza internamente durante o treinamento da voz: speech-tools, praat, SPTK, HTK e hts-engine. Além desses programas, para a criação da voz é necessário um conjunto de alofones, disponibilizado pelo grupo LaPS, além de arquivos de áudio de frases faladas, com suas respectivas transcrições em arquivos de texto, formando uma espécie de “banco de fala”. Inicialmente, foram utilizados áudios disponibilizados pelo LaPS, que também havia iniciado uma pesquisa na área de síntese, porém sem muito empenho. Esses áudios consistem em 613 frases gravadas por um voluntário, que estão disponíveis em <http://www.laps.ufpa.br/falabrasil/files>. É importante destacar que, destas 613 frases, as 200 primeiras são provenientes do artigo “Frequência de ocorrência dos fones e lista de frases foneticamente balanceados no português falado no Rio de Janeiro”¹, e o restante são frases criadas pelo grupo do LaPS. Nenhuma frase possuía transcrição, então foi necessário transcrever todas as 613 frases.

A criação da voz é feita a partir destes áudios. O MARY possui uma interface gráfica bastante amigável, e os tutoriais presentes no site ajudaram no processo de criação. Para criar a voz, deve-se executar o “Voice Import Tools” (Figura 3.9), configurando alguns parâmetros e selecionando alguns dos módulos disponíveis (seguindo o que o tutorial do site indica). A partir da execução dos mesmos, temos, no fim do processo, arquivos prontos para serem introduzidos no sistema MARY.

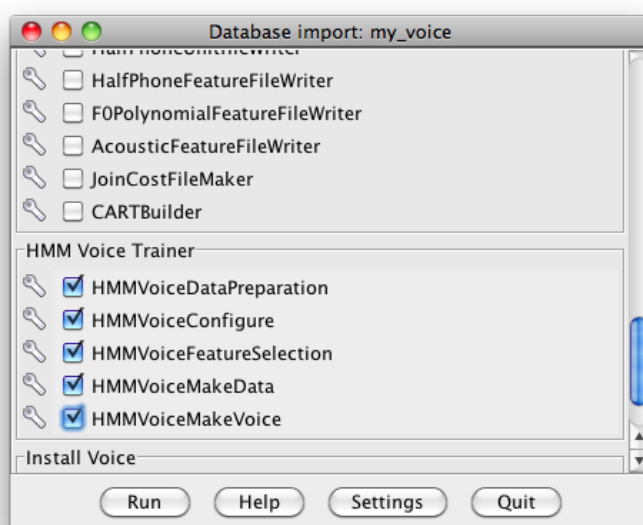


Figura 3.9 – Voice Import Tools

O processo é demorado, pois o MARY trabalha com treinamento utilizando HMM (Hidden Markov Models) para analisar as amostras e transcrições a fim de construir a base de informações sobre a qual a síntese trabalha. Ele analisa as transcrições e suas amostras, captando a ocorrência de fonemas, já definidos (pelo arquivo de alofones citado anteriormente) e baseados no idioma português brasileiro. Baseando-se nessa análise é atribuído um som ao fonema em questão, levando em conta o padrão de onda que se fez mais presente dentre as amostras no momento que este fonema apareceu nas mesmas.

¹ Disponível em http://iecom.dee.ufcg.edu.br/~jcis/dezembro1992/volume%207/JCIS_1992_7_02.pdf.

Feitos alguns testes e tendo resultados satisfatórios, foi decidido criar um corpus com amostras de frases próprio, do Comunica. Para isso, foi escolhida a bolsista Maitê Dupont como a “voz do Comunica”. Inicialmente ela gravou todas as 613 frases que já haviam sido gravadas pelo LaPS, e com essas foi feita a primeira versão do sintetizador do Comunica. O resultado foi uma síntese robusta, mas com qualidade pobre, pois havia muitas mudanças na entonação durante a síntese, o que a deixava com audição estranha.

Então foi decidido aumentar a quantidade de frases no “banco de voz”. Foram gravadas algumas outras frases e palavras individuais (nomes de municípios do Rio Grande do Sul e numerais, gravações estas que foram utilizadas também para outras finalidades no Projeto). No fim, o total de amostras chegou a 1096. Houve um problema durante as gravações: o ruído de fundo na sala onde foram gravadas as frases era alto, então foi necessário aplicar remoção de ruído em todos os arquivos, individualmente, pois a remoção de ruído em grupo acabava alterando o cabeçalho WAV dos arquivos no momento de salvá-los, fazendo com que não fossem aceitos pelo MARY. Foi utilizado o software *Audacity* para a redução de ruído, pois o processo com o sistema SOX não foi satisfatório.



Figura 3.10 – Interface web

Após o tratamento dos áudios e transcrição para arquivos texto correspondentes, foi feito o processo no MARY para criar a voz baseada na voz da Maitê. O resultado foi uma síntese melhor do que a anterior, provando que, se aumentado o “banco de voz”, a qualidade da síntese também aumenta, pois o

MARY consegue mais amostras das partes de fala correspondentes aos fonemas, e assim consegue treinar melhor as transições e composições de fonemas.

Foi também feita a incorporação do MARY no sistema do Comunica. Como o MARY trabalha como um servidor Web, ele só necessita de um chamada HTTP com a frase a ser sintetizada para retornar a síntese em áudio. O MARY possui uma interface Web, que foi modificada para ser usada em testes no Comunica (Figura 3.10).

A comunidade de desenvolvimento para o MARY é bastante ativa, visto que ainda é um sistema que está sendo aperfeiçoado. Isso é outro ponto positivo do MARY, pois existem vários projetos de síntese que não são atualizados há anos (Mbrola Project, por exemplo). Durante o processo de criação da voz no Comunica, foi feito contato tanto com pesquisadores do LaPS quanto pesquisadores de outras universidades no mundo, a fim de corrigir problemas técnicos que surgiram durante o processo.

Ainda são necessárias algumas melhorias na síntese, pois apesar de estar falando todas as palavras completas, a voz ainda apresenta certas alterações de entonação durante a fala. Apesar deste problema com a qualidade, pode-se dizer que a síntese é feita com sucesso, assim como sua integração ao sistema do Projeto Comunica.

4 MÓDULO DE ACESSO À BASE

O Módulo de Acesso à Base possui dois subgrupos: o grupo de preparação, mapeamento e carga dos dados (componente de carga), coordenado por Fábio Moreira da Silva (DFL) e o grupo de acesso à base efetivo (demais componentes), coordenado pelo Prof. Dr. Stanley Loh (InText Mining).

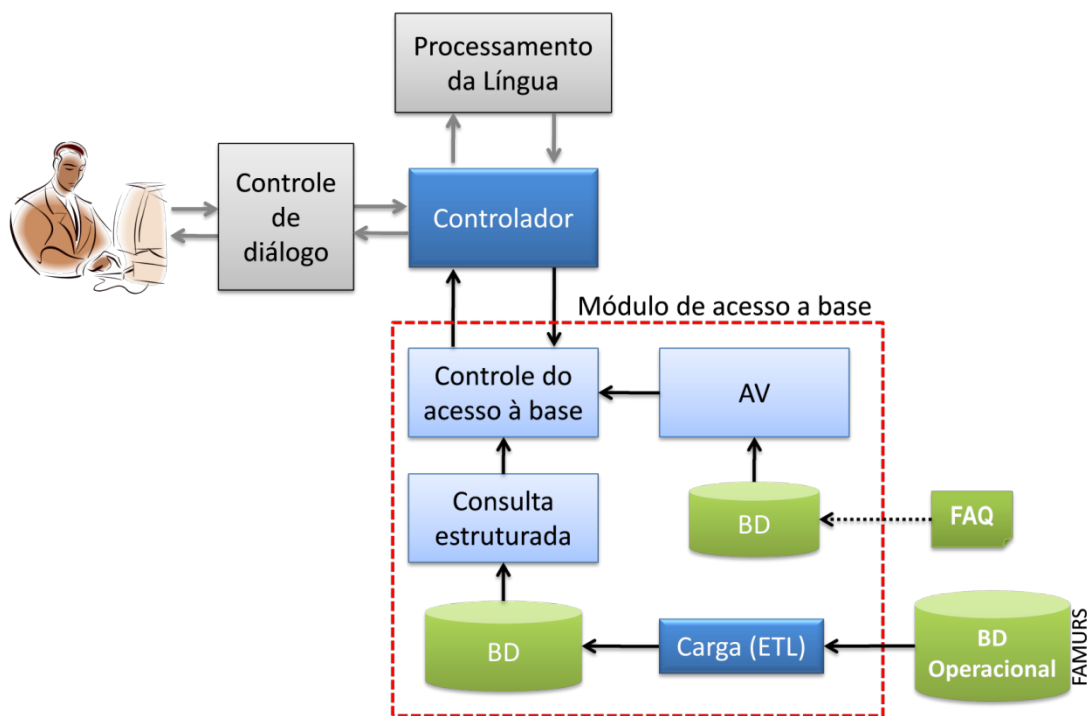


Figura 4.1 – Módulo de Acesso à Base

Quando o controlador recebe uma consulta processada pelo módulo de Processamento da Língua, ele repassa-a para o componente de controle do acesso à base. Tal componente realiza uma consulta estruturada, utilizando uma base de dados preparada, modelada e organizada (segundo o esquema estrela) em dimensões próprias para as consultas identificadas como relevantes no domínio da aplicação (FAMURS, no caso). Quando o módulo de controle de acesso à base identifica que a consulta não pode ser atendida pela BD, ele utiliza o Assistente Virtual (AV), que é um componente encarregado de formular uma resposta relevante para o usuário, tentando fazer com que ele reformule a pergunta de maneira que possa ser atendida. O AV utiliza uma base de dados específica que é construída a partir das consultas mais frequentes (FAQ) dos usuários e outros documentos relevantes, obtidos da FAMURS e considerados por ela relevantes.

Por questões de performance e segurança, o sistema utiliza, como já descrito, bases de dados próprias, sendo que os dados originais da FAMURS são mantidos em seus sistemas originais (legados), denominados na figura de BD operacional. Eventualmente e em momentos predefinidos e oportunos, os dados são atualizados e consolidados no BD da aplicação pelo componente de Carga (ATL).

O módulo de controle do acesso à base é um Serviço Web cuja funcionalidade principal chama-se "hello". Para chamá-lo, basta executar tal função e passar

como parâmetro um arquivo XML descrevendo os parâmetros de consulta a serem utilizados.

Exemplo de chamada do componente utilizando linguagem PHP:

```
hello(utf8_encode($_POST['xml'])).
```

O padrão XML para troca de informações foi definido juntamente com o grupo de ontologias. Esse XML é usado na busca estruturada, a fim de fornecer dados necessários para busca de informações.

A primeira versão do XML definido foi:

```
<?xml version="1.0" encoding="UTF-8" ?>
<query>
  <Phrase>
    Quanto Bagé vai receber de ICMS no dia quinze de janeiro de
    dois mil e dez?
  </Phrase>
  <Concepts>
    <Municipio>Bagé</Municipio>
    <Data>
      <Dia>quinze</Dia>
      <Mes>janeiro</Mes>
      <Ano>dois mil e dez</Ano>
    </Data>
    <Transferencia_constitucional>
      ICMS
    </Transferencia_constitucional>
  </Concepts>
  <Action>value_of_one_transfer</Action>
</query>
```

Porém, o XML acima não contemplava intervalos de datas, tendo sido, portanto, modificado para:

```
<?xml version="1.0" encoding="UTF-8" ?>
<query>
  <Phrase>
    Quanto Bagé vai receber de ICMS em Janeiro de 2010?
  </Phrase>
  <Concepts>
    <Municipio>Bagé</Municipio>
    <Data intervalo = "inicio" >
      <Dia>um</Dia>
      <Mes>Janeiro</Mes>
      <Ano>dois mil e dez</Ano>
    </Data>
    <Data intervalo = "Fim" >
      <Dia>trinta e um</Dia>
      <Mes>Janeiro</Mes>
      <Ano>dois mil e dez</Ano>
    </Data>
```

```

        <Transferencia_constitucional>
            ICMS
        </Transferencia_constitucional>
    </Concepts>
    <Action>value_of_one_transfer</Action>
</query>

```

Ao sair, a resposta da base estruturada a mesma deverá ser em forma de XML conforme o exemplo seguinte:

```

<?xml version="1.0" encoding="UTF-8" ?>
<ANSWER>
    <TEXT>
        O ICMS DO PRÓXIMO MÊS PARA O MUNICÍPIO DE PELOTAS É DE CEM
        MIL E QUINHENTOS REAIS
    </TEXT>
</ANSWER>

```

4.1 Padrões aceitos atualmente

Os padrões de 1 a 4 foram definidos em conjunto com a equipe de ontologias e com a FAMURS. Eles são utilizados para procura de respostas pela base estruturada.

Abaixo, segue um exemplo de como é trabalhado o XML recebido do modulo controlador:

```

1  <?
2  include_once("xml5.php");
3  $xml = $_POST["txtxml"];
4  $e = $_POST["escolha"];
5
6  //$xml = simplexml_load_string($xml);
7
8  $xml = str_replace("\\", "", $xml);
9
10 $xml = simplexml_load_string(utf8_encode($xml));
11 $Phrase = $xml->Phrase;
12 $Municipio = $xml->Concepts->Municipio;
13 //$Instance = utf8_decode($xml->Concept[0]->Instance) ;
14
15 if ($xml->Concepts->Data->attributes() != ""){
16     $Intervalo1 = $xml->Concepts->Data[0]->attributes()->intervalo;
17     $Dia1 = $xml->Concepts->Data[0]->Dia;
18     $Mes1 = $xml->Concepts->Data[0]->Mes;
19     $Ano1 = trim($xml->Concepts->Data[0]->Ano);
20
21     $Intervalo2 = $xml->Concepts->Data[1]->attributes()->intervalo;
22     $Dia2 = $xml->Concepts->Data[1]->Dia;
23     $Mes2 = $xml->Concepts->Data[1]->Mes;
24     $Ano2 = trim($xml->Concepts->Data[1]->Ano);
25 }else{
26     $Intervalo1 = null;
27     $Dia1 = $xml->Concepts->Data->Dia;
28     $Mes1 = $xml->Concepts->Data->Mes;
29     $Ano1 = trim($xml->Concepts->Data->Ano);
30
31     $Intervalo2 = null;
32     $Dia2 = null;
33     $Mes2 = null;
34     $Ano2 = null;
35 }
36 $TransferenceType = $xml->Concepts->Transferencia_constitucional;

```

A seguir, são listados os padrões de pergunta e resposta aceitos pelo sistema atualmente:

- **Padrão 1: Resposta é um valor simples**

Quanto Bagé vai receber de ICMS no dia 15 de abril de 2010?

Resposta completa: **Bagé** vai receber **R\$ 2.500,00** de **ICMS** no dia **15 de abril de 2010**.

Resposta codificada: **<Município>** receberá R\$ **<Resposta>** de **<Transf. Const.>** em **<Data>**.

Padrão 1a

ACTION = value_of_one_transfer

Resposta é um valor.

Parâmetros obrigatórios: município, transf. e dia (se não tiver ano, assume-se o ano atual; se não tiver mês, assume-se o mês atual)

Padrão 1b

Variação da pergunta: Quanto Bagé vai receber de ICMS em abril de 2010?

ACTION = value_of_one_transfer

Pegar o 1º valor disponível na base para o mês indicado.

Resposta é um valor.

Parâmetros obrigatórios: município, transf. e mês (se não tiver ano, assume-se o ano atual)

Padrão 1c

Variação da pergunta: Quanto Bagé vai receber de ICMS ?

ACTION = value_of_one_transfer

Pegar o 1º valor disponível na base para o mês e ano corrente.

Resposta é um valor.

Parâmetros obrigatórios: município e transf. (se não tiver dia, mês e ano, assume-se o mês e ano atual)

- **Padrão 2: Resposta é uma sequência de valores, incluindo a soma**

Quanto Bagé vai receber em abril de 2010?

Resposta completa: **Bagé** vai receber **R\$ 10.000,00** em **abril de 2010**: **R\$ 5.000,00** de **ICMS**; **R\$ 300,00** de **Salário Educação**; **R\$ 250,00** da **Lei Kandir**; **R\$ 4.450,00** de **FPM**.

Resposta codificada: **<Município>** receberá R\$ **<Resposta>** em **<Data>**: R\$ **<Valor>** de **<Transf. Const.>**; R\$ **<Valor>** de **<Transf. Const.>**; R\$ **<Valor>** de **<Transf. Const.>**; R\$ **<Valor>** de **<Transf. Const.>**, num total de **<soma dos valores>**.

Padrão 2a

ACTION = value_of_all_transfers (retornar valor, data e tipo de transferência, para todas as transferências encontradas)

Resposta são vários valores e uma soma no final.

Parâmetros obrigatórios: município e mês (se não tiver ano, assume-se o ano atual)

Padrão 2b

Variação da pergunta: Quanto Bagé vai receber em 15 de abril de 2010?

ACTION = value_of_all_transfers (retornar valor, data e tipo de transferência, para todas as transferências encontradas)

Pegar somente valores para este dia específico

Resposta são vários valores e uma soma no final.

Parâmetros obrigatórios: município e dia (se não tiver mês e ano, assume-se os correntes)

- **Padrão 3: Resposta é uma data**

-

Quando Bagé vai receber a 1ª parcela do ICMS do mês de abril de 2010?

Resposta completa: **Bagé** vai receber a 1ª parcela do **ICMS** no dia **15 de abril de 2010**.

Resposta codificada: **<Município>** receberá a 1ª **<Pagamento>** de **<Transf. Const.>** em **<Data>**.

Padrão 3a

ACTION = next_date

Resposta é uma data completa.

Pegar na base a 1ª data com valor do transf.

Parâmetros obrigatórios: município e transf. (se não tiver mês ou ano, assume-se o corrente)

- **Padrão 4: Resposta são várias datas**

Quando Bagé vai receber as parcelas do FPM do mês de abril de 2010?

Resposta completa: **Bagé** vai receber as parcelas do **FPM** nos dias **12, 19 e 26 de abril de 2010**.

Resposta codificada: **<Município>** receberá as parcelas de **<Transf. Const.>** em **<Dia>**, **<Dia>** e **<Data>**.

Padrão 4a

ACTION = all_dates

Resposta são várias datas completas.

Pegar na base todas as datas com valor do transf.

Parâmetros obrigatórios: município e transf. (se não tiver mês ou ano, assume-se o corrente).

4.2 Módulo para conversão de números e siglas por extenso

Por demanda da equipe de voz, foi necessário desenvolver um módulo para gerar números, valores ou datas por extenso. Tal solicitação se dá pelo sintetizador não conseguir sintetizar números. Com isso, foram feitos métodos onde são recebidos valores, datas ou números e são retornados seus valores por extenso. O contrário também acontece para fazer consultas nas bases de conhecimento.

Ex.: Recebe R\$ 10.000

Devolve dez mil reais

Também, a pedido da equipe de voz, foi construído um método que recebe uma sigla e devolve a mesma por extenso. O funcionamento é simples. É passada uma sigla e cada sílaba da mesma é repassada para o método que retorna os fonemas de cada palavra. Isso não é feito para as siglas que são faladas por extenso, como FEX.

4.3 Web service do módulo de controle do acesso à base

Foi criado um *Web service* que processa requisições do modulo controlador. Esse *Web service* recebe perguntas elaboradas pelos usuários, já convertidas para texto e marcadas (com os conceitos relevantes identificados) e retorna respostas contendo o resultado do processamento da pergunta pela base Estruturada ou do assistente virtual.

O referido *Web service* foi escrito em linguagem PHP 5, utilizando SOAP e adotando o formato UTF-8 para a troca de dados, permitindo a utilização de caracteres universais e para evitar qualquer tipo de incompatibilidade com os demais sistemas.

Foram instalados scripts PHP em dois URLs distintos para ilustrar o funcionamento do *Web service*.

No primeiro URL (<http://www.adsdigital.com.br/ad/comunica/server.php>) são processadas requisições (perguntas) feitas através dos demais módulos do projeto COMUNICA e este a pesquisa para fornecimento da resposta adequada através do sistema de inteligência artificial denominado de assistente virtual. Segue um trecho do seu código fonte.

```

1  <?php
2
3  function hello($someone)
4  {
5      //Referência às bibliotecas de funções do AV
6      require ("funcoes.php");
7      require ("globais.php");
8
9      //Conexão com banco de dados
10     gfun_conecta_banco();
11
12
13     if (!is...
21
22     $str_resposta=fun_busca_resposta_do_banco_de_dados_4($someone , $int_id_sessao,0);
23
24     return utf8_encode($str_resposta[0]);
25 }
26 $server = new SoapServer(null,
27     array('uri' => "urn://www.herong.home/res", 'encoding'=>'utf-8' ));
28 $server->addFunction("hello");
29 $server->handle();
30 ?>
31

```

No segundo (<http://www.brmais.com/teste/client2.php>) encontra-se um módulo cliente que utiliza o *Web service* anteriormente descrito. Um trecho de seu código encontra-se a seguir.

```

1  <pre>
2  <?php
3
4  ini_set("soap.wsdl_cache_enabled", "0");
5
6  // A seguir você deverá informar a URL do webservice.
7  $oSoapClient = new SoapClient("http://www.adsdigital.com.br/teste/service.wsdl");
8
9  $aOptions = array (
10     "start_debug"=> "1",
11     "debug_port"=> "10000",
12     "debug_host"=> "localhost",
13     "debug_stop"=> "1");
14
15  foreach($aOptions as $key => $val) {
16     $oSoapClient->__setCookie($key,$val);
17  }
18  var_dump($oSoapClient->getUser());
19  ?>
20  </pre>
21  [<a href=".">Voltar</a>]

```

O diagrama seguinte ilustra o funcionamento do módulo *Web service* que foi desenvolvido.

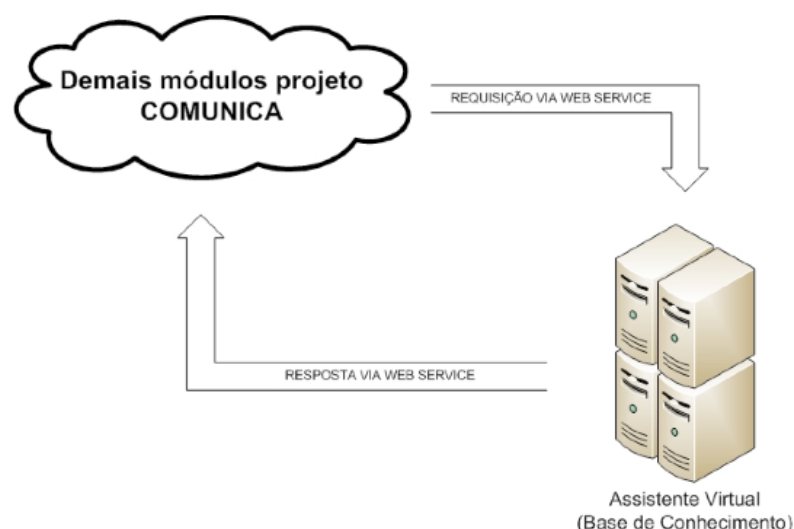


Figura 4.2 – Esquema do módulo de acesso à base

4.4 Módulo de carga (ETL)

Para realizar a importação dos dados da base operacional para a base estruturada do projeto Comunica, faz-se uso de um código *ETL* em PHP. Este módulo foi desenvolvido pelo bolsista Jeferson Antunes Rubert.

Já atingimos a meta necessária ao projeto de prover a carga dos dados do sistema "Boletim Eletrônico da Receita" para a base do Comunica. Desde então, fizemos diversos melhoramentos no processo de carga de dados, incluindo *scripts* que detectam as novas edições do boletim (são dados mensais) e agora fazem a importação de forma automática, sem a necessidade de intervenção. Também foi implementado um sistema de mensagens por e-mail. Sempre que uma nova importação é feita um e-mail é enviado, listando informações sobre a mesma, como por exemplo o local do *log* da importação. Além disso um sistema de verificação de erros também foi implantando, com o fim de minimizar o monitoramento dos *scripts* Automáticos.

Considerando a implantação do produto no mercado em novos clientes, o fluxo de dados varia de acordo com o universo de cada cliente. Por exemplo, a base de dados de uma dada empresa poderia ser atualizada a cada mês com novos dados. Nesse caso, a entrada de dados seria mensal. No entanto, podem existir Clientes com dados estáticos. Além disso, também existe a parte da interação com o Cliente, por exemplo, colhendo dados de voz e o entendimento da base de dados usada pelo mesmo (ou criação de uma).

Portanto, para novos clientes, os dois submódulos precisam ser readaptados.

Entretanto, o formato escolhido para a base de dados, conhecido como "esquema estrela", é um padrão de mercado para bases de apoio a tomada de decisão (*datamarts/datawarehouses*), o que permitirá manter boa parte do trabalho realizado nos futuros clientes e também permitirá evoluções do produto para que as consultas e cargas sejam cada vez mais adaptáveis.

5 MÓDULO DE PROCESSAMENTO DA LÍNGUA

O módulo de Processamento de Língua foi desenvolvido pelo grupo coordenado pela Prof^a. Dr^a. Aline Villavicencio, da UFRGS, sendo o grupo composto pelos alunos Rodrigo Wilkens, Leonardo Zilio, Kassius Prestes e Cristhian Herrera. Este módulo foi desenvolvido para apresentar como fluxo básico o seguinte processamento sobre a versão escrita da pergunta do usuário: reconhecimento de termos relevantes, casamento de padrão dos termos e identificação da completude da pergunta.

O módulo foi implementado utilizando a linguagem de programação Java (versão 6) e a biblioteca JENA² para o tratamento de ontologias. Como ferramentas, foram utilizadas o analisador sintático *Palavras*³, para realizar a análise da pergunta (indicando termos como substantivos, nomes próprios, verbos), e da ontologia relativa ao domínio da FAMURS. Pelo fato do *Palavras* ser um software proprietário e seu custo ser muito elevado, no período final do projeto, optou-se por desenvolver um *parser* próprio (descrito na subseção 5.3).

5.1 Comunicação com outros módulos

Sob a ótica do módulo de Processamento de Língua interação pode ocorrer de duas formas, a primeira quando a pergunta do usuário é identificada como um padrão e a segunda quando é considerada como adequada a um padrão, mas está incompleta.

Na primeira interação ocorrem quatro trocas de mensagens:

1. Envio do texto da pergunta do usuário para o controlador;
2. O controlador passa a pergunta para o módulo de processamento da língua;
3. O processamento da língua passa a sua pergunta para o controlador;
4. A pergunta é repassada para o módulo de acesso à base para recuperar a informação desejada.

Na segunda interação ocorre um número não determinado de mensagens, pois depende do número de esclarecimentos com o usuário que serão necessários para identificar os conceitos necessários, sendo as mensagens:

1. Envio do texto da pergunta do usuário para o controlador;
2. O controlador passa a pergunta para o módulo de processamento da língua;
3. O módulo de processamento da língua identifica um padrão incompleto e passa esta informação para o controlador;
4. O controlador solicita ao módulo de diálogo que esclareça os termos não presentes;
5. O módulo de diálogo informa as informações incompletas esclarecidas com o usuário;

² Biblioteca de funções (em linguagem de programação Java) desenvolvida pela HP e disponível no endereço <http://jena.sourceforge.net/>.

³ Com documentação disponível no endereço <http://beta.visl.sdu.dk/>.

6. O controlador informa as novas informações repassadas pelo usuário e a mensagem 3;
7. O módulo identifica um padrão de pergunta (caso identifique um padrão incompleto é realizado novamente o pedido de esclarecimento);
8. A pergunta é repassada para o módulo de acesso à base para recuperar a informação desejada.

Tal fluxo é representado na figura seguinte. Os módulos do fluxo são aqueles descritos nos outros capítulos deste relatório.

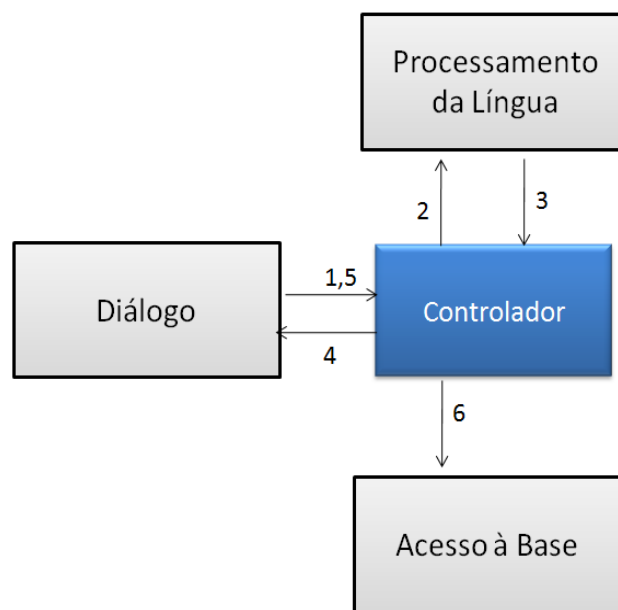


Figura 5.1 – Fluxo do processamento da língua

5.2 Funcionamento interno

São considerados como recursos fundamentais para o funcionamento do módulo: (1) a ontologia de domínio, (2) a ontologia genérica e (3) os padrões de perguntas.

5.2.1 Ontologia de domínio

A ontologia é descrita em OWL⁴ e foi desenvolvida pelo linguista Leonardo Zilio através de entrevistas com especialistas no domínio e análise de textos do domínio.

⁴ Linguagem de representação de ontologias indicada pelo World Wide Web Consortium (W3C).

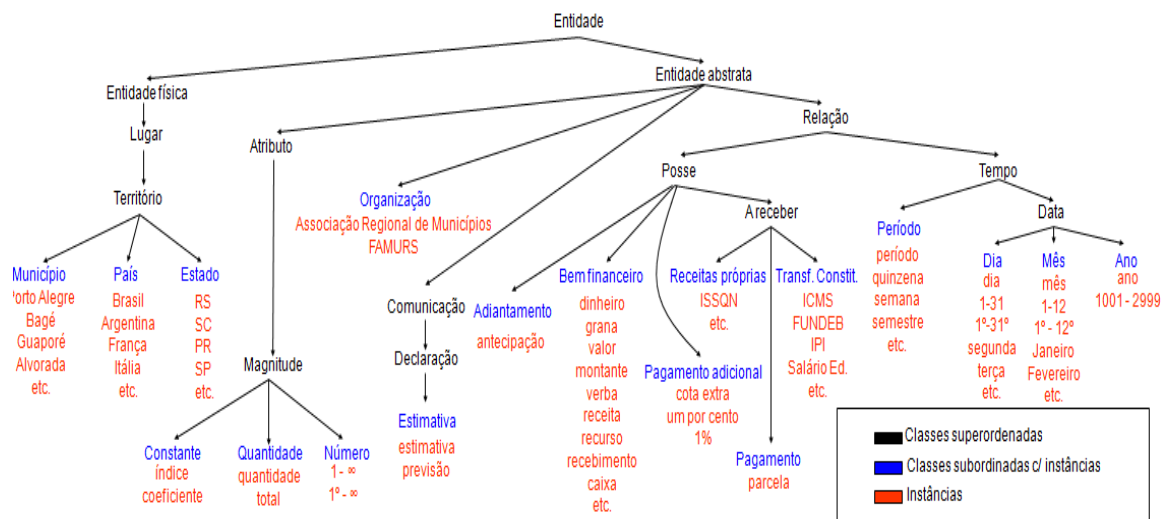


Figura 5.2 – Conceitos gerais da ontologia desenvolvida

5.2.2 Criação de ontologia para a língua portuguesa

Com objetivo de alcançar um maior acerto do sistema, foi desenvolvida uma ontologia geral do português, assim permitindo que o sistema possa identificar palavras que são sinônimos e hiperônimos de outras. Com isto permitimos que o usuário possa desfrutar de uma maior flexibilidade e aumentamos o conforto no uso do sistema.

A ontologia foi gerada a partir de dados extraído de forma semiautomática do dicionário de Língua Portuguesa da Editora Porto, que contém relações entre palavras como sinonímia e hiperonímia, de um recurso léxico (que existe para o português) chamado Papel. O Papel consiste de um conjunto de relações entre termos. Existem diversos tipos de relações entre palavras que estão descritas no papel, entre elas causa, finalidade, sinonímia e hiperonímia. Para a importação do Papel para uma ontologia utilizamos as relações de hiperonímia. Hiperonímia é uma palavra que dá a ideia de um todo, da qual se originam diversas ramificações, por exemplo, veículo é hiperônimo de carro, barco e avião.

Para a representação dos dados na linguagem OWL foi necessária a conversão das relações. Para hiperonímia foi criada uma ontologia automaticamente. Porém as relações de sinonímia não puderam ser convertidas por causa da polissemia, ou múltiplos sentidos, das palavras. Assim foi criado um sistema que informa quando há conflito e permite que um linguista realize correções (usando como base dicionários da língua portuguesa).

Após o término das correções nas relações de sinonímia pelo linguista, elas foram integradas com as relações de hiperonímia de forma automática.

5.2.3 Base de padrões de perguntas

Os padrões de perguntas descrevem o formato das perguntas mais comuns feitas à FAMURS. Esses contêm os conceitos da ontologia que devem ser informados na pergunta para ela possa ser respondida corretamente. São três os conceitos envolvidos: Município, Data e transferência constitucional. Os seguintes padrões foram identificados e estão sendo utilizados:

- Valor de uma transferência – O usuário pergunta quanto o município vai receber de uma transferência constitucional e em determinada data (sendo a data opcional), a resposta é um valor simples. Exemplo: Quanto Bagé vai receber de ICMS no dia vinte e três de julho de dois mil e dez?
- Valor de todas as transferências – O usuário pergunta quanto o município vai receber em determinada data (sendo a data opcional), a resposta é uma lista de valores de todas as transferências constitucionais recebidas pelo município naquela data e a soma desses valores. Exemplo: Quanto Bagé vai receber no dia quinze de janeiro de dois mil e dez?
- Próxima Data – O usuário pergunta quando o município vai receber a primeira/segunda/terceira/quarta parcela de um determinada transferência constitucional, a resposta é uma data. Exemplo: Quando Bagé vai receber a terceira parcela do FPM de julho de dois mil e dez?
- Todas as datas - O usuário pergunta quando o município vai receber as parcelas de uma determinada transferência constitucional em um determinado período, a resposta é uma lista de datas. Exemplo: Quando Bagé vai receber as parcelas do FPM de julho de dois mil e dez?

Para o reconhecimento dos termos relevantes é utilizado o analisador sintático *Palavras* para a identificação dos municípios, através da identificação de substantivos e padrões de nomes. O reconhecimento de datas usa expressões regulares como base, sendo identificadas as diferentes maneiras de dia, o mês, o ano e intervalo de datas (como quinzena, bimestre). Para o reconhecimento de datas intervalares é assumido que os intervalos são sempre determinísticos, por exemplo, a primeira quinzena vai sempre do dia 1 ao dia 15, e a segunda do dia 16 ao dia 30, o primeiro semestre vai de janeiro a junho e o segundo de julho a dezembro. A única exceção é a semana, que pode variar de mês para mês, mas mesmo assim é assumido que as semanas sempre começam no domingo e acabam no sábado. Também é assumido que não serão feitas perguntas com períodos que incluam mais de um ano, por isso não é possível, em uma única pergunta saber os valores para, por exemplo, um semestre que inicie em novembro e termine em abril.

A identificação dos conceitos presentes na pergunta, com base na ontologia, ocorre por simples casamento de padrões. Quando um termo da frase encontra-se na ontologia, ele é identificado como um conceito relevante e, portanto, o par conceito instância é armazenado pelo sistema.

O módulo recebe com entrada uma frase, essa frase é passada pelo módulo de reconhecimento de datas, que transforma uma data escrita por extenso para uma data no formato dd/mm/aaaa, desse modo, o analisador sintático pode identificar que há uma data na frase de forma mais acurada. O próximo passo consiste em passar a frase com a data já identificada pelo analisador sintático (*Palavras*) para obtenção dos termos da frase que podem ser relevantes. Cada um desses termos da frase é procurado na ontologia de domínio para ver quais são os conceitos identificados na frase. Os termos identificados na frase são colocados em uma tabela, com o conceito identificado e o conceito propriamente dito, por exemplo, uma entrada nessa tabela seria *Município = Viamão*.

Depois de identificados os conceitos, são verificados em quais dos padrões a frase mais se adequa. Por exemplo, uma possível consulta perguntaria: "Quanto Viamão vai receber de ICMS no dia quinze de janeiro de dois mil e dez?". Seriam identificados os conceitos de município, transferência constitucional e data, assim o

padrão de pergunta que necessita desses três conceitos será 100% adequado ao caso de valor de uma transferência. Caso nenhum padrão seja 100% adequado, há a possibilidade de pedir para o usuário informar os conceitos que faltaram para o padrão ser completamente adequado. Caso a pergunta tenha menos de 50% de adequação são retornados quais conceitos foram identificados e é dito que nenhum padrão foi identificado.

Um tratamento que é realizado é a identificação de que um padrão é provável, mas não tem informações suficientes na pergunta, então sendo realizado um retorno diferenciado indicando quais as informações estão faltando na pergunta. Com a capacidade de ser realimentado, ou seja, é possível continuar uma consulta anteriormente feita, é possível se manter um contexto sobre as frases do usuário. Para que isso seja feito, o módulo tem como um de seus parâmetros de entrada o XML retornado por ele na sua última consulta. Assim ao receber como entrada um XML, junto com uma nova frase, a tabela montada a partir da última frase recebida é reconstruída a partir do XML, e é necessário buscar os novos conceitos na nova frase a ser analisada.

5.3 Substituição do parser Palavras pelo MST Parser

Pelo fato do parser palavras ser proprietário e possuir um alto custo, ele foi substituído por outro, cujo código fonte é aberto, o MST parser⁵. Para funcionar corretamente, tal parser necessita ser treinado para a língua portuguesa (do Brasil), além de atuar em conjunto com uma ferramenta de pré-processamento, o PoS Tagger do OpenNLP⁶.

O PoS Tagger anota as palavras de uma frase, indicando suas respectivas classes gramaticais. Ele utiliza dois modelos disponibilizados pela OpenNLP, um deles treinado com o algoritmo Perceptron, outro com o de máxima entropia, e cada um desses modelos anota as palavras com uma classe gramatical. Caso elas sejam diferentes, um algoritmo de aprendizado decide o correto. O PoS Tagger obteve uma acurácia de 79,59% sob o corpus Amazônia⁷.

O MST parser foi treinado para o português brasileiro com o corpus Bosque⁸, e integrado com o PoS Tagger para sua utilização no projeto Comunica. Esse treinamento foi feito usando a técnica *10-fold cross validation*, e teve uma acurácia de 82,86%.

Como o MST parser utiliza um modelo que tem um longo tempo de carga, para evitar com que o processamento de uma consulta demore, foi criado um serviço que faz a carga do modelo em memória e fica esperando por conexões que devem enviar frases que serão analisadas pelo parser. Desse modo, o parser funciona como um serviço no servidor e é acompanhado de um módulo cliente, que intermedia as requisições externas. O cliente desenvolvido foi integrado ao módulo de processamento da linguagem do Comunica.

O módulo de processamento de linguagem foi testado com o novo parser e seus resultados foram muito próximos dos que eram obtidos com o Palavras.

5 Disponível em: <http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html>

6 Disponível em: <http://incubator.apache.org/opennlp/>.

7 Disponível em: <http://www.linguatca.pt/floresta/corpus.html#amazonia>

8 Disponível em: <http://www.linguatca.pt/floresta/corpus.html#bosque>

6 CONSIDERAÇÕES FINAIS

Neste documento apresentou-se o relatório técnico do projeto Comunica, contemplando seu estágio atual. Diversas versões dos diferentes módulos foram desenvolvidas. Para cada versão, um sistema de demonstração foi elaborado e apresentado em um evento técnico-científico ou empresarial. Tais demonstrações foram importantes para apresentar o produto ao público acadêmico e geral, obtendo-se *feedback* importante sobre as técnicas utilizadas. Também foram importantes para demonstrar o produto a possíveis clientes e parceiros, obtendo-se *feedback* sob a aceitação do mesmo no mercado e possíveis aplicações.

Algumas dificuldades foram encontradas ao longo do desenvolvimento do projeto. A primeira diz respeito ao início do projeto, que só foi possível em novembro de 2009, com a liberação dos recursos. Além disso, um aspecto importante foi a falta de experiência dos bolsistas envolvidos no processo de desenvolvimento de grande parte dos componentes. Isso porque são todos alunos, acadêmicos em estágio inicial de formação, com bolsas de iniciação tecnológica (além do fato dos alunos terem o costume de trocar muito de projeto, a fim de obter uma experiência acadêmico-científica mais ampla ou a fim de obter maior remuneração). Logo, ocorram casos de desenvolvedores que estavam preparados e totalmente conscientes do projeto terem migrado para outros projetos. Com isso, novo treinamento era necessário com um novo bolsista. Por outro lado, oferecer tal oportunidade aos alunos é uma contribuição importante do projeto. Além disso, apesar dessas dificuldades, **o resultado atual do sistema é satisfatório e pode ser colocado em prática.**

É importante citar que alguns dos módulos inicialmente utilizados no projeto, apesar de funcionarem corretamente e terem auxiliado muito o seu desenvolvimento, tornaram-se inviáveis no sentido em que encareceriam muito o produto final. Isso porque são softwares proprietários, com licenças de alto custo. Um exemplo disso é o *parser* "Palavras", cuja licença de que dispúnhamos era de cunho acadêmico. Isso significa que, na prática, para que o produto pudesse ser utilizado e comercializado, necessitaríamos adquirir uma licença de uso comercial para cada cliente, o que possui alto custo. Logo, como se optou por desenvolver um módulo simplificado de perguntas e respostas, capaz de realizar processamento semelhante ao que é feito pelo software "Palavras", mas com escopo reduzido ao contexto da FAMURS. Tal módulo deve ser customizado para cada cliente, ou seja, adaptado para cada contexto.

Apesar disso, um subproduto importante do projeto consiste em um módulo de perguntas e respostas com interface Web. Tal produto já pode ser colocado em prática na FAMURS e poderia ser facilmente adaptado para dispositivos móveis e outros contextos (sem a parte de voz).

APÊNDICE I - MECANISMOS DE GERÊNCIA, COORDENAÇÃO E COLABORAÇÃO

A maior parte da comunicação entre os participantes do projeto foi feita em reuniões semanais que envolveram todos os bolsistas e a coordenação, além da interação pessoal, já que toda a equipe de bolsistas trabalhou na sala da Conexum, no Centro de Empreendimentos em Informática, que se localizava nas instalações do Instituto de Informática da UFRGS. Contudo, o Comunica contou com um espaço no Google Grupos⁹ (lista de discussão) onde todos recebiam e-mails enviados por algum integrante do projeto, fosse ele bolsista, coordenador, consultor ou colaborador. A Figura seguinte apresenta algumas das funcionalidades de tal lista.



Figura I.1 - Espaço do projeto no Google Grupos

Na questão da documentação do conhecimento desenvolvido no projeto, foi amplamente usado o wiki do Comunica (atualmente disponível em um servidor com acesso restrito, interno). A figura seguinte apresenta uma visão geral do wiki usado para documentação do desenvolvimento do sistema.

⁹ Disponível no URL <http://groups.google.com/group/comunica-ufrgs/>.



Figura I.2 – Wiki com a documentação do projeto

Também foi usado o Bugzilla (Figura I.2), um software *open-source* que permitiu o gerenciamento de erros e funcionalidades necessárias. Com ele, quando algum dos integrantes do grupo detectava algum erro ou acreditava haver a necessidade de se fazer alguma modificação ou implementar alguma funcionalidade, cadastrava-a no sistema (Figura I.3) e o responsável pelo respectivo módulo recebia uma notificação. Com base nisso, ao entrar no sistema, observava-se uma descrição do erro ou funcionalidade e poder-se-ia resolvê-lo. O sistema permitiu a geração de relatórios de erros em aberto e erros corrigidos, para controle.

Bugzilla – Bug 19
Cadastramento de usuários
Last modified: 2010-06-01 08:57:26

[Home](#) | [New](#) | [Search](#) | | [Reports](#) | [My Requests](#) | [My Votes](#) | [Preferences](#) | [Log out](#) wives@inf.ufrgs.br

Bug List: (3 of 8) [First](#) [Last](#) [Prev](#) [Next](#) [Show last search results](#)

Details

Summary:

Bug#: [19](#)

Product:

Component:

Status: ASSIGNED

Resolution:

Hardware:

OS:

Version: 1

Priority:

Severity:

URL:

Depends on:

Blocks:

[Show dependency tree](#) - [Show dependency graph](#)

People

Reporter: [Leandro V](#)
[wives@inf.ufrgs.br](#)

Assigned To: [Daniel Ne](#)
[danieln@inf.ufrgs.br](#)

Add CC:

CC: [wives@inf.ufrgs.br](#)

Orig. Est.	Current Est.	Hours Worked	Hours Left	%Complete	Gain	Deadline
<input type="text" value="0.3"/>	<input type="text" value="0.3"/>	<input type="text" value="0.1"/> + <input type="text" value="0"/>	<input type="text" value="0.2"/>	<input type="text" value="33"/>	<input type="text" value="-0.0"/>	<input type="text" value="2010-05-29"/> (YYYY-MM-DD)

[Summarize time \(including time for bugs blocking this bug\)](#)

Attachments

[Add an attachment](#) (proposed patch, testcase, etc.)

Figura I.3 - Edição de bug/funcionalidade no Bugzilla

Utilizou-se, ainda, o sistema de versionamento de código SVN para controle de versões de código dos módulos e também da base de transcrição de telefonemas. O sistema SVN é um conhecido software livre de uso comum nas comunidades de programação em equipes. Atualmente também começa a ser utilizado por equipes de outras áreas com a finalidade de sincronização de documentos. Esse sistema é de uso restrito interno, por questões de sigilo de desenvolvimento.

APÊNDICE II - FEIRAS E EVENTOS TÉCNICO-CIENTÍFICOS E EMPRESARIAIS

Durante a execução do projeto, os subprodutos (sistema de demonstração e artigos técnicos) foram apresentados nos seguintes eventos acadêmicos:

- International Conference on Computational Processing of the Portuguese Language, PROPOR, 27-30 de abril de 2010, PUCRS, Porto Alegre, Brasil.
- International Conference on Computational Linguistics, COLING, 23-27 de agosto de 2010, Beijing, China.
- North American Chapter of the Association for Computational Linguistics, NAACL, 1-6 de junho de 2010, Millennium Biltmore Hotel, Los Angeles, Estados Unidos.

Como resultado de tais eventos, podemos destacar a publicação dos seguintes artigos:

- PRESTES, K. V. ; WILKENS R. ; ZILIO L. ; VILAVICENCIO A. . Extração e Validação de Ontologias a partir de Recursos Digitais. In: Ontobras-MOST, 2011, Gramado. Proceedings of Joint IV Seminar on Ontology Research in Brazil and VI International Workshop on Metamodels, Ontologies and Semantic Technologies, 2011. v. 776. p. 183-188.
- Wilkens, Rodrigo Souza. Villavicencio, Aline. Muller, Daniel Nehme. Wives, Leandro Krug. Silva, Fabio da. Loh, Stanley. COMUNICA : a voice question answering system for portuguese [recurso eletrônico]. In: International Conference on Computational Processing of Portuguese (9. : 2010 April : Porto Alegre, RS). PROPOR 2010
- WILKENS, R. ; VILLAVICENCIO, Aline ; MULLER, D. N. ; WIVES, L. K. ; SILVA, F. M.; LOH, S. COMUNICA - A Question Answering System for Brazilian Portuguese. In: 23rd International Conference on Computational Linguistics (Coling 2010), 2010, Pequim. Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010). Morristown, NJ, USA : Association for Computational Linguistics, 2010. p. 21-24.

O produto também foi apresentado nas seguintes feiras e eventos empresariais:

- Rio Info 2010, promovido pela RioSoft e SEPRORJ, no Rio de Janeiro, de 31/08 a 02/09/2010.
- Estande da Conexum na Feira do Empreendedor, promovido pelo SEBRAE/RS, em Porto Alegre, de 09 a 12/09/2010.
- Projeto Vendedor de Software na Feira MERCOPAR 2010, promovido pelo SEBRAE/RS, em Caxias do Sul, em 21/10/2010.