CENTRO UNIVERSITÁRIO FEEVALE

PAULO RICARDO MUNIZ BARROS

MODELAGEM E IMPLEMENTAÇÃO DO AGENTE DE DOMÍNIO DO AMBIENTE AMPLIA

Novo Hamburgo
2009

PAULO RICARDO MUNIZ BARROS

MODELAGEM E IMPLEMENTAÇÃO DO AGENTE DE DOMÍNIO DO AMBIENTE AMPLIA

Trabalho de conclusão de curso apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação pelo Centro Universitário Feevale

Orientador: Marta Rosecler Bez
Co-Orientador: Cecília Dias Flores

Novo Hamburgo
2009

PAULO RICARDO MUNIZ BARROS

Trabalho de Conclusão do Curso de Sistemas de Informação, com o título de MODELAGEM E IMPLEMENTAÇÃO DO AGENTE DE DOMÍNIO DO AMBIENTE AMPLIA, submetido ao corpo docente do Centro Universitário Feevale, como requisito necessário para obtenção do Grau Bacharel em Sistemas de Informação.

Aprovado por:

_____
Professor Orientador

_____
Professor (Banca examinadora)

_____
Professor (Banca examinadora)

Novo Hamburgo, dezembro de 2009

AGRADECIMENTOS

Agradeço...

...primeiramente a Deus, que me deu tudo que sempre precisei.

...aos meus pais, Paulo e Glaussa, por me ensinarem a retidão do caminho.

...a minha noiva Júlia, que sempre me apoiou incondicionalmente durante esta trajetória.

...aos professores Marta e Cecília que me orientaram neste trabalho.

...aos colegas de trabalho que sempre me apoiaram e em especial ao Fabiano que me incentivou a iniciar a graduação.

...a todas as pessoas que de alguma forma contribuíram para a conclusão deste trabalho.

RESUMO

Com intuito de criar um recurso adicional para auxílio a educação médica, o Grupo de Inteligência Artificial do Instituto de Informática da Universidade Federal do Rio Grande do Sul (II/UFRGS) em parceria com outras instituições, desenvolveu o AMPLIA, um Ambiente Multiagente Probabilístico Inteligente de Aprendizagem, que possibilita a aprendizagem colaborativa. Este é um ambiente aberto, em que o aluno constrói um modelo gráfico da representação de sua hipótese diagnóstica, a partir de um caso clínico oferecido pelo ambiente. Inicialmente, o projeto teve algumas limitações, em virtude de componentes utilizados. Uma delas foi o uso em uma rede aberta, a Internet. A nova proposta aqui apresentada é remodelar o ambiente, corrigindo essas e outras limitações; criando uma ferramenta que possa ser integrada facilmente através da Internet para comunicação entre os agentes, possibilitando então, a disseminação do conhecimento e amplificando as possibilidades e uso na educação médica. O AMPLIA é composto por três agentes: *Agente de DOMÍNIO*, *Agente APRENDIZ* e *Agente MEDIADOR*. A proposta deste trabalho refere-se ao estudo, modelagem e desenvolvimento de uma nova versão do *Agente de DOMÍNIO*. A principal função deste agente é a avaliação qualitativa e quantitativa do modelo bayesiano, construído pelo aluno, através do *Agente APRENDIZ*. Esta avaliação é baseada no modelo previamente construído pelo especialista em uma base de casos reais. Para isto, é proposto o uso de algumas ferramentas tecnológicas, tais como as API – JADE e UnBBayes, através da linguagem JAVA. Como resultado esperado, tem-se o agente de DOMÍNIO implementado na plataforma Java, sob um novo paradigma, que permitirá o uso do AMPLIA com recursos de comunicação da WEB.

Palavras-chave: Inteligência Artificial. Redes Bayesianas. Agentes. Ensino de medicina. UnBBayes. AMPLIA.

ABSTRACT

In order to create an additional resource to aid in medical education, the Artificial Intelligence Group of the Institute of Informatics, Federal University of Rio Grande do Sul (II / UFRGS) in partnership with other institutions, developed the AMPLIA, a probabilistic Multiagente Intelligent Environment Learning, which enables collaborative learning. This is an open environment in which the student builds a model of graphic representation of your diagnosis from a clinical case provided by the environment. Initially, the project had some limitations, because the components used, one was for use in an open network, Internet. The new proposal is presented here is reshaping the environment correcting these and other limitations, creating a tool that can be integrated easily over the Internet for communication between agents, allowing then the dissemination of knowledge and expanding the possibilities and use in medical education . The AMPLIA consists of three agents: Domain Agent, Learner Agent and Mediator Agent. The purpose of this study refers to the study, modeling and development of a new version of Domain Agent. The main function of this agent is the qualitative and quantitative evaluation of the Bayesian model, built by the student, through the Learner Agent. This assessment is based on the model previously built by a specialist based on real cases. For this, it is proposed the use of certain technological tools, such as API - JADE and UnBBayes by JAVA language. As expected, the agent has been implemented in the field of Java platform under a new paradigm that will allow the use of AMPLIA using the communication resources of the Web.

Keywords: Artificial Intelligence. Bayesian networks. Agents. Teaching of medicine. UnBBayes. AMPLIA.

LISTA DE FIGURAS

Figura 1.1 – Modelo lógico do banco de dados 15
Figura 1.2 – Modelo Físico do banco de dados 27
Figura 2.1 –Estrutura básica do funcionamento do Hibernate. 37
Figura 3.1 – Interface de desenvolvimento Eclipse 41
Figura 3.2 – Arquitetura da plataforma Eclipse 42
Figura 3.3 – Editor gráfico de desenvolvimento 43
Figura 3.4 – Assistente Wizards 43
Figura 3.5 – Gerenciador de módulos. 44
Figura 4.1 – Interface Diagnósticos 47
Figura 4.2 – Interface Dados Gerais 48
Figura 4.3 – Interface Prioridade Nodo 48
Figura 4.4 – Interface Nodos Bogos 49
Figura 4.5 – Interface Arquivos 50
Figura 4.6 – Interface Imagens 50
Figura 4.7 – Interface Casos Reais 51
Figura 4.8 – Interface do módulo de manutenção das propriedades do Nodo 52
Figura 4.9 – Interfaces de Probabilidades Condicionais 53
Figura 4.10 – Interface de Informações Adicionais do Nodo 53
Figura 4.11 – Interface de Diagnósticos Associados ao Nodo. 54

LISTA DE TABELAS

Tabela 1 – Definição da tabela casoEstudo gerada pelo DbDesigner 29
Tabela 2 – Definição da tabela casosReais gerada pelo DbDesigner 29
Tabela 3 – Definição da tabela diagAssNodo gerada pelo DbDesigner 29
Tabela 4 – Definição da tabela nodos gerada pelo DbDesigner 29
Tabela 5 – Definição da tabela arquivoscasoestudo gerada pelo DbDesigner 30
Tabela 6 – Definição da tabela estado gerada pelo DbDesigner 30
Tabela 7 – Definição da tabela imagensCasoEstudo gerada pelo DbDesigner 31
Tabela 8 – Definição da tabela imagensNodo gerada pelo DbDesigner 31
Tabela 9 – Definição da tabela nodoCasoReal gerada pelo DbDesigner 31
Tabela 10 – Definição da tabela nodosBogus gerada pelo DbDesigner 32
Tabela 11 – Definição da tabela relacionamentoNodosPais gerada pelo DbDesigner 32
Tabela 12 – Definição da tabela seqProb gerada pelo DbDesigner 32
Tabela 13 – Definição da tabela valorProbNodo gerada pelo DbDesigner 33
Tabela 14 – Definição da tabela cabSeqProb gerada pelo DbDesigner 33
Tabela 15 – Definição da tabela nodosCaso gerada pelo DbDesigner 33
Tabela 16 – Exemplo tabela Mapeada pelo Hibernate 38
Tabela 17 – Possíveis classificações da propriedade do nodo 49

LISTA DE ABREVIATURAS E SIGLAS

AMPLIA	Ambiente Multiagente Probabilístico Inteligente de Aprendizagem
CIC/UNB	Departamento de Ciência da Computação da Universidade de Brasília
CNRM	Comissão Nacional de Médicos Residentes
DI/UL	Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa
FACIL	Biblioteca Delphi de comunicação entre Agentes
FAMED	Faculdade de Medicina da Universidade Federal do Rio Grande do Sul
FIPA-ACL	Comitês técnicos responsáveis pela elaboração dos padrões para uma linguagem de comunicação entre agentes
HCPA	Hospital de Clínicas de Porto Alegre
HQL	Hibernate Query Language
IA	Inteligência Artificial
JADE	Biblioteca de comunicação entre Agentes
LAN	Rede Local
SEAMED	Editor de redes bayesianas
UFCSPA	Universidade Federal de Ciências da Saúde de Porto Alegre
UFRGS	Universidade Federal do Rio Grande do Sul
UPA	Agentes de controle de perfil de usuário
WEB	*W*orld *W*ide *W*eb
XML	Extensible Markup Language

SUMÁRIO

INTRODUÇÃO 11

1 BANCO DE DADOS DO AGENTE DE DOMINIO 14
1.1 MODELAGEM DO BANCO DE DADOS DO AGENTE DE DOMÍNIO 14
1.2 IMPORTAÇÃO DOS DADOS 34

2 FRAMEWORK DE COMUNICAÇÃO COM BANCO DE DADOS 36
2.1 HIBERNATE 36

3 IDE PARA O DESENVOLVIMENTO 40
3.1 ECLIPSE 40
3.2 NETBEANS 42
3.3 INTERFACE DE DESENVOLVIMENTO UTILIZADA 44

4 INTERFACE DO AGENTE DE DOMINIO 47
4.1 CASO DE ESTUDO 47
4.2 INFORMAÇÕES SOBRE OS NODOS 51

CONSIDERAÇÕES FINAIS 55

REFERÊNCIAS BIBLIOGRÁFICAS 58

APÊNDICES 61

INTRODUÇÃO

Atualmente, existem inúmeros recursos pedagógicos à disposição de educadores e educandos, através do uso de ambientes virtuais de aprendizagem colaborativa. Cada ferramenta tem como foco uma determinada área do conhecimento, em virtude de adequar-se melhor ao resultado obtido.

Um ambiente virtual de aprendizagem colaborativo tem como principal objetivo disponibilizar um espaço de colaboração para construção de conhecimento, com o intuito de desenvolver atividades educativas.

Na formação da área médica isto não é diferente, contudo, nesta área o conhecimento envolve domínios incertos, onde o aluno deve construir um raciocínio lógico traçado através de variáveis, sintomas apresentados pelo paciente, a fim de definir um possível diagnóstico da doença apresentada. Atualmente, os principais recursos de informática utilizados para desenvolver este raciocínio lógico são as listas de discussão, *chats*, teleconferências (FLORES, 2005).

Com o objetivo de criar um recurso adicional para auxílio à educação médica apoiando o desenvolvimento do raciocínio diagnóstico, o Grupo de Inteligência Artificial do Instituto de Informática da Universidade Federal do Rio Grande do Sul (II/UFRGS), em parceria com a Faculdade de Medicina da Universidade Federal do Rio Grande do Sul (FAMED/UFRGS), o Departamento de Ciência da Computação da Universidade de Brasília (CIC/UnB) e o Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa (DI/UL) desenvolveu o AMPLIA. Este é um ambiente aberto, em que o aluno constrói um modelo gráfico da representação de sua hipótese, que leva ao diagnóstico para um dado caso clínico.

O AMPLIA é um ambiente multiagente constituído por três tipos de agentes cognitivos (Agente APRENDIZ, Agente MEDIADOR e Agente de DOMÍNIO). Estes agentes mantêm uma comunicação com um servidor (ComServer) (FLORES, 2005). Esta estrutura vale-se também do uso de uma interface de comunicação com um editor de redes bayesianas - o SEAMED (FLORES, 2002). Os agentes fazem uso dos recursos de comunicação em rede, viabilizados através de uma biblioteca denominada FACIL (GLUZ, 2002). Esta biblioteca foi desenvolvida baseada nos padrões FIPA-ACL (comitês técnicos responsáveis pela elaboração dos padrões para uma linguagem de comunicação entre agentes) (GLUZ, 2005), sendo a responsável pela troca de mensagem entre os agentes.

Nesta estrutura, o Agente APRENDIZ é responsável por enviar informações geradas pelo aluno, o qual desenvolve uma rede bayesiana através do seu raciocínio lógico por meio do caso clínico apresentado. Já o Agente de DOMÍNIO tem a função de comparar a rede construída pelo aluno com a rede construída pelo especialista, que deve identificar os prováveis conflitos. O resultado gerado é então encaminhado para o Agente MEDIADOR, que será responsável pela seleção das estratégias pedagógicas mais convenientes para auxílio ao correto diagnóstico (FLORES, 2005).

Esta estrutura está limitada ao uso em uma rede local (LAN) em função das tecnologias aplicadas. Contudo, um novo projeto foi iniciado pelo grupo de pesquisa da UFCSPA, com intuito de utilizar tecnologias que comportem o uso de recursos de Internet. Com isto, aumenta-se a abrangência e o alcance, possibilitando expandir o conhecimento às demais universidades e encurtando distâncias, permitindo a criação de uma grande rede de repositórios com diagnósticos e casos clínicos à disposição de acadêmicos da área da saúde.

Esta nova versão do AMPLIA mantém as funcionalidades e estruturas já empregadas no projeto original; porém, foi alterada a linguagem de programação, que inicialmente era DELPHI e passa a ser JAVA (SUN, 2009); com intuito de facilitar a comunicação através da WEB. Outra grande alteração empregada foi a troca da biblioteca FACIL, pelo JADE (*Java Agent Development Framework*) – um framework desenvolvido em JAVA para viabilizar a comunicação entre agentes (JADE, 2009).

Este trabalho de conclusão é focado na re-engenharia do Agente de DOMÍNIO, voltado para a WEB. Segundo Flores (2005), o papel do Agente de DOMÍNIO é avaliar o modelo bayesiano construído pelos alunos, representado pelo Agente APRENDIZ, quanto à sua viabilidade (se o modelo corresponde a uma rede bayesiana), corretude (se não existem nodos desnecessários ou excludentes), completude (se não faltam nodos ou arcos no modelo) e eficácia (qual é o seu comportamento diagnóstico diante de uma base de dados de casos médicos reais). Para isto, o agente de DOMÍNIO também se valerá do uso de uma API Java chamada UnBBayes (UnBBayes, 2009), usada para modelar e avaliar a rede Bayesiana construída pelo especialista e pelo aprendiz.

Diante do exposto, o objetivo principal deste trabalho é implementar, pesquisar e desenvolver melhorias no Agente de DOMÍNIO, de um ambiente multiagente; que se vale do uso de uma rede probabilística bayesiana para alcançar o conhecimento através de um sistema colaborativo.

O desafio foi grande, uma vez que o sistema AMPLIA, apesar de possuir grande volume de publicações, teses, dissertações, trabalhos de conclusão e "*papers*", carece de

documentação técnica. Outro fator importante a ser destacado é o fato de ter sido desenvolvido em partes, por “muitas mãos”, sem uma padronização.

O Apêndice A, deste trabalho, contempla os estudos e análises teóricas do sistema AMPLIA, apresentando as funcionalidades do sistema como um todo. Esta análise teórica foi desenvolvida pelo grupo que vem trabalhando no AMPLIA nos últimos dois anos. O aluno Regis Leandro Sebastiani, no semestre anterior, publicou este material em seu trabalho de conclusão, pois foi um dos principais autores deste.

No primeiro capitulo, é apresentado um estudo detalhado do banco de dados do agente de DOMÍNIO do ambiente AMPLIA, bem como sua validação através da carga de dados inicial, que servirá de base para o desenvolvimento deste projeto.

O segundo capítulo é dedicado a apresentação do Framework Hibernate, bem como as suas vantagens no desenvolvimento do agente de DOMÍNIO. No quarto capítulo será apresentado um estudo detalhado sobre IDE’s de desenvolvimento, bem como as suas vantagens em seu uso, as tecnologias empregadas, os critérios e os pontos que foram decisórios para o seu uso.

O último capítulo contém a descrição das interfaces do Agente de DOMÍNIO do sistema AMPLIA, apresentando as funcionalidades existentes e uma proposta de alteração que será desenvolvida no projeto de re-engenharia. Algumas considerações a respeito do trabalho até aqui realizado e do trabalho que ainda será desenvolvido serão abordadas.

1 BANCO DE DADOS DO AGENTE DE DOMINIO

Neste capítulo serão apresentados alguns detalhes do agente de DOMÍNIO, como a estrutura do banco de dados utilizado no projeto de re-engenharia. Serão apontados e abordados os requisitos identificados, a partir da interface original do agente de DOMÍNIO, no qual se fez uma engenharia reversa para extrair os campos e a definição da base de dados a ser utilizada. O banco de dados dará suporte tanto ao agente de DOMÍNIO quanto aos demais agentes, em virtude de concentrar todas as informações da rede bayesiana e os casos reais utilizados para avaliação qualitativa da rede no ambiente AMPLIA. Esta estrutura foca a eliminação da limitação da tecnologia original, que faz uso apenas de uma rede local (LAN), em função das tecnologias aplicadas. Com isto, aumentam-se a abrangência e o alcance, possibilitando expandir o conhecimento às demais universidades e encurtando distâncias, consentindo a criação de uma grande rede de repositórios com diagnósticos e casos clínicos à disposição de acadêmicos da área da saúde.

1.1 MODELAGEM DO BANCO DE DADOS DO AGENTE DE DOMÍNIO

A modelagem do banco de dados, a seguir apresentada, foi dividida em duas etapas: modelo lógico e modelo físico.

Inicialmente é apresentado um modelo Lógico (figura 1.1) do protótipo, que serviu como base para validação e desenvolvimento do modelo físico do banco de dados.

O modelo lógico foi elaborado com auxílio da ferramenta BrModelo, uma ferramenta *Freeware*, voltada para elaboração e desenvolvimento de modelagem em banco de dados objeto relacional (BrModelos, 2009).

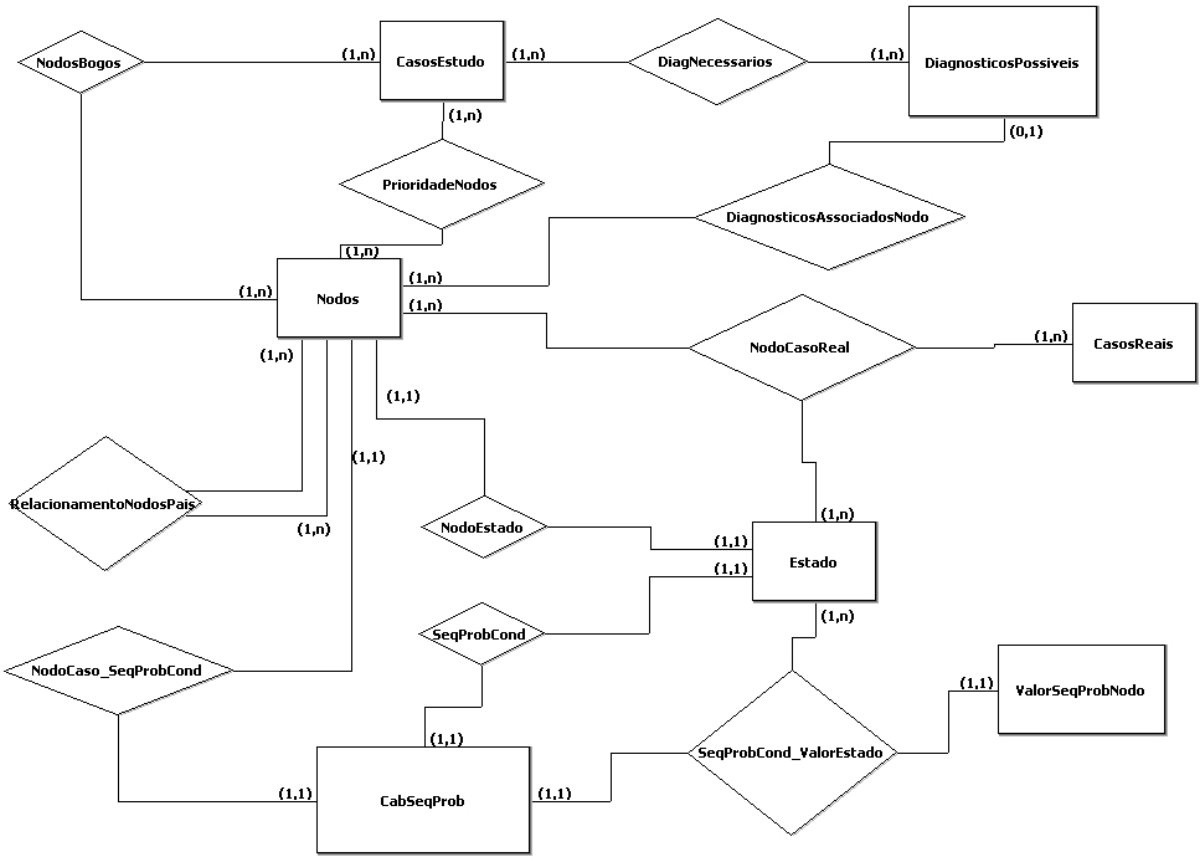

Figura 1.1 – Modelo lógico do banco de dados

Na arquitetura apresentada há algumas entidades que serão detalhadas no decorrer do capítulo, porém, salienta-se alguns detalhes iniciais os quais servirão para se ter uma idéia do modelo apresentado.

CasoEstudo é uma entidade que deve servir para armazenar as principais informações e detalhes do caso a ser modelado. É através desta entidade que boa parte do banco de dados será relacionado.

A entidade **Nodos** é usada a fim de armazenar os nodos da rede Bayesiana e suas devidas propriedades, e estes nodos possuem muitas particularidades, entre elas, o estado do nodo, os diagnósticos associados aos nodos, NodosBogos (nodos que não interferem no resultado final da rede), PrioridadeNodos, ou seja, a prioridade que cada nodo deve assumir no contexto do caso, como Essencial, Excludente, Trigger, Complementar e Desnecessário (Tabela 1).

Outras entidades que precisam de uma breve explanação são **CabSeqProb** e **ValorProbNodo**. Estas entidades são responsáveis por armazenar os valores probabilísticos dos nodos no contexto da rede, de cada caso clínico. Portanto, é através dela que serão

armazenados e alterados os pesos (valores) que cada nodo tem sobre os demais nodos que ele se relaciona diretamente.

A principal função desta base de dados é o armazenamento das definições do especialista para correta avaliação do agente de DOMÍNIO. Com isso é possível avaliar o modelo bayesiano construído pelos alunos quanto à sua viabilidade (se o modelo corresponde a uma rede bayesiana), corretude (se não existem nodos desnecessários ou excludentes), completude (se não faltam nodos ou arcos no modelo) e eficácia (qual é o seu comportamento diagnóstico diante de uma base de dados de casos médicos reais). (FLORES, 2005).

A partir do modelo lógico gerado e fazendo uso das telas do agente original, realizou-se um processo de engenharia reversa e foram extraídos os demais campos e características para que sejam alimentadas ao novo agente.

Em decorrência das telas e modelo lógico, foi gerado o modelo físico (figura 1.2), contendo todas as regras e definições do novo agente de DOMÍNIO.

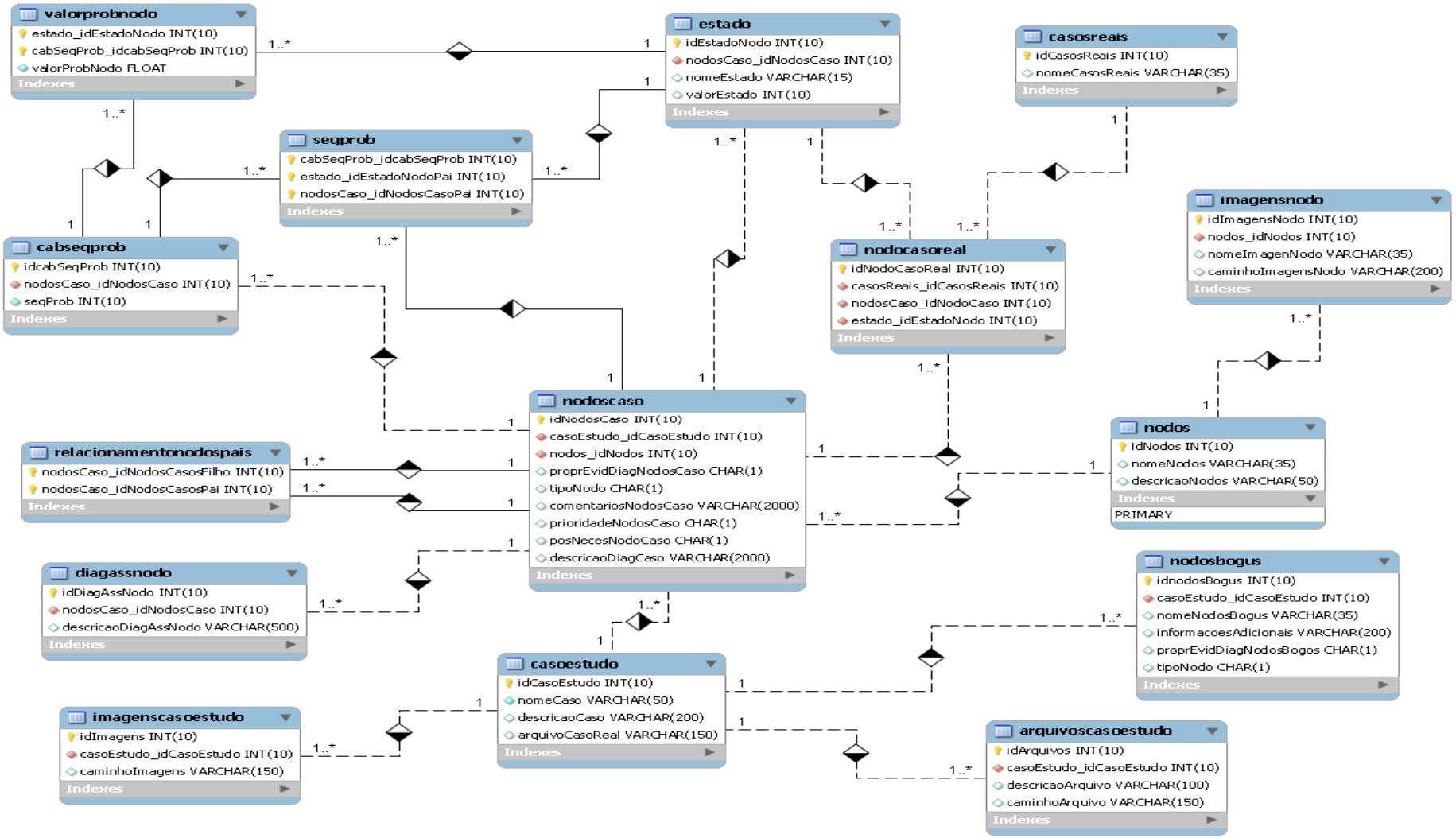

Figura 1.2 – Modelo Físico do banco de dados

Através de uma API, conhecida como UnBBayes, que será utilizada para manipular graficamente a rede bayesiana e sua inferência, obtém-se um arquivo do tipo XML (*Extensible Markup Language*) (W3C, 2009) e, a partir deste, é possível criar uma classe para alimentar o banco de dados.

Neste arquivo XML gravam-se os dados e atributos gerados em função da inferência dos nodos da rede. Uma vez que estas informações tenham sido armazenadas no banco de dados, os demais agentes podem consultar e alterar as informações de uma forma única, e respeitando as regras definidas no agente de DOMÍNIO.

As tabelas do modelo apresentado são:

- casoestudo;
- arquivoscasoestudo;
- nodos;
- nodoscaso;
- cabseqprob;
- casosreais;
- diagassnodo;
- estado;
- imagenscasoestudo;
- imagensnodo;
- nodocasoreal;
- nodosbogus;
- relacionamentonodospais;
- seqprob;
- valorprobnodo;

A seguir são apresentadas as principais características de cada tabela e sua principal função dentro do escopo geral. O script SQL para gerar o banco de dados encontra-se no Apêndice A.

A tabela 1 apresenta a definição da tabela **casoEstudo**, usada para armazenar as informações do caso a ser apresentado, como o nome do caso, uma breve descrição que será informada pelo especialista da área e o caminho do arquivo XML resultante da rede bayesiana gerada. Nesta tabela o campo **idCasoEstudo**, servirá para relacionar-se com as demais tabelas.

Tabela 1 – Definição da tabela casoEstudo gerada pelo DbDesigner

casoEstudo

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
idCasoEstudo	**INTEGER**	PK	NN	UNSIGNED			AI
nomeCaso	VARCHAR(50)		NN				
descricaoCaso	VARCHAR(200)						
arquivoCasoReal	VARCHAR(150)						

A tabela 2 apresenta a definição da tabela **casosReais**, que serve para armazenar as informações reais, baseadas em históricos e experiências relatadas pelo especialista. Estas informações não interferem nos dados estatísticos da rede, servindo apenas para posteriormente, em conjunto com a tabela **nodoCasoReal**, validar qualitativamente a rede gerada pelo agente APRENDIZ.

Tabela 2 – Definição da tabela casosReais gerada pelo DbDesigner

casosReais

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
idCasosReais	**INTEGER**	PK	NN	UNSIGNED			AI
nomeCasosReais	VARCHAR(35)						

A tabela 3 apresenta a definição da tabela **diagAssNodo**, usada para armazenar os diagnósticos pertinentes ao nodo e ao caso de estudo que se relaciona, obrigatoriamente, devem pertencer a rede gerada pelo aluno.

Tabela 3 – Definição da tabela diagAssNodo gerada pelo DbDesigner

diagAssNodo

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
idDiagAssNodo	**INTEGER**	PK	NN	UNSIGNED			AI
nodosCaso_idNodosCaso	**INTEGER**	PK	NN	UNSIGNED			
descricaoDiagAssNodo	VARCHAR(500)						

A tabela 4 apresenta a definição da tabela **Nodos**, usada para armazenar informações exclusivas dos nodos, não pertinentes ao caso estudado, ou seja, dados exclusivos e não vinculados a casos de estudo. Nesta tabela são armazenados o nome dos nodos e um breve descritivo do nodo. Para esta tabela foi adicionada uma coluna **idNodos**, que será utilizada para seu relacionamento com outras tabelas.

Tabela 4 – Definição da tabela nodos gerada pelo DbDesigner

nodos

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
idNodos	**INTEGER**	PK	NN	UNSIGNED			AI
nomeNodos	VARCHAR(35)						
descricaoNodos	VARCHAR(50)						

A tabela 5 apresenta a definição da tabela **arquivoscasoestudo**, usada para armazenar o caminho de arquivos adicionais que servirão de apoio ao aluno (Agente APRENDIZ), para resolução e apresentação do caso clínico. Esta tabela também poderá servir futuramente para guardar um objeto adicional para o caso de estudo (XML, binários, etc). Atualmente, possui uma coluna com a descrição do arquivo, uma coluna com o caminho do arquivo, uma coluna com o vínculo com o caso de estudo em questão. Como identificação única, esta tabela possui uma coluna **idArquivos**, que pode servir futuramente para relacionamento com outras tabelas.

Tabela 5 – Definição da tabela arquivoscasoestudo gerada pelo DbDesigner

arquivosCasoEstudo

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
idArquivos	**INTEGER**	PK	NN	UNSIGNED			AI
casoEstudo_idCasoEstudo	INTEGER		NN	UNSIGNED			
descricaoArquivo	VARCHAR(100)						
caminhoArquivo	VARCHAR(150)						

A tabela 6 apresenta a definição da tabela **estado**. Esta tabela é usada para armazenar todos os estados possíveis que cada nodo poderá assumir no contexto do caso. É importante salientar que cada nodo poderá assumir vários estados, prévios para cada Caso de Estudo. Esta tabela possui uma coluna que conterá o nome do estado, uma coluna que vincula ao nodo e ao caso em questão. Esta tabela também possui uma coluna de identificação única que servirá para vincular o estado ao valor que cada estado assumirá na rede.

Tabela 6 – Definição da tabela estado gerada pelo DbDesigner

estado

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
idEstadoNodo	**INTEGER**	PK	NN	UNSIGNED			AI
nodosCaso_idNodosCaso	INTEGER		NN	UNSIGNED			
nomeEstado	VARCHAR(15)						

A tabela 7 apresenta a definição da tabela **imagensCasoEstudo**. Esta tabela serve para armazenar o caminho de imagens que servirão de apoio como informação adicional ao aluno (Agente APRENDIZ), para resolução e apresentação do caso clínico. Para elaboração desta tabela, trabalhou-se com a idéia de ter um servidor de imagens médicas, trabalho de pesquisa que está sendo desenvolvido em paralelo por outros pesquisadores. Portanto, apenas será armazenado o caminho web deste servidor e a respectiva imagem.

Tabela 7 – Definição da tabela imagensCasoEstudo gerada pelo DbDesigner

imagensCasoEstudo

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
idImagens	**INTEGER**	PK	NN	UNSIGNED			AI
casoEstudo_idCasoEstudo	INTEGER		NN	UNSIGNED			
caminhoImagens	VARCHAR(150)						

A tabela 8 apresenta a definição da tabela **imagensNodo**. Esta tabela serve para armazenará o caminho de imagens de cada nodo, estas imagens podem auxiliar como informação adicional, assim como as imagens do caso de estudo, como apoio ao aluno (Agente APRENDIZ), para resolução do caso clínico em questão. Esta tabela possui uma coluna com o vínculo ao nodo a que se refere esta imagem, uma coluna com o nome da imagem, o caminho web do servidor de imagens.

Tabela 8 – Definição da tabela imagensNodo gerada pelo DbDesigner

imagensNodo

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
idImagensNodo	**INTEGER**	PK	NN	UNSIGNED			AI
nodos_idNodos	**INTEGER**	PK	NN	UNSIGNED			
nomeImagenNodo	VARCHAR(35)						
caminhoImagensNodo	VARCHAR(200)						

A tabela 9 apresenta a definição da tabela **nodoCasoReal**, usada para armazenar o vínculo entre a tabela de casosReais e a tabela Nodo. Esta tabela possui uma coluna que identifica o caso real, o nodo e o estado a que se refere. A coluna de identificação do estado, (que vincula nodoCasoReal ao Estado), foi definida para permitir valores nulos (null), possibilitando, com isto, uma identificação, de ser um estado para o caso, ainda não definido pelo especialista.

Tabela 9 – Definição da tabela nodoCasoReal gerada pelo DbDesigner

nodoCasoReal

estado_idEstado**odo**

estado presentes na tabela de estado ou Indefinido, não vinculado a estados.

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
idNodoCasoReal	**INTEGER**	PK	NN	UNSIGNED			AI
casosReais_idCasosReais	**INTEGER**	PK	NN	UNSIGNED			
nodosCaso_idNodoCaso	INTEGER		NN	UNSIGNED			
estado_idEstadoNodo	INTEGER			UNSIGNED			

A tabela 10 apresenta a definição da tabela **nodosBogus**. Esta tabela tem a função de armazenar os nodos que poderão ou não estar presentes na rede do aluno (Agente APRENDIZ), na resolução e apresentação do caso clínico. Esta possui uma coluna com o

nome do nodo, informações adicionais sobre este nodo, o tipo de nodo (Booleano, Simbólico, Decisão, Utilidade ou Numérico), uma coluna com a informação se ele é um nodo de evidencia ou diagnostico e uma coluna de auto-incremento, para identificação única deste nodo.

Tabela 10 – Definição da tabela nodosBogus gerada pelo DbDesigner

nodosBogus

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
idnodosBogus	**INTEGER**	PK	NN	UNSIGNED			AI
casoEstudo_idCasoEstudo	**INTEGER**	PK	NN	UNSIGNED			
nomeNodosBogus	VARCHAR(35)						
informacoesAdicionais	VARCHAR(50)						
proprEvidDiagNodosBogos	CHAR(1)						
tipoNodo	CHAR(1)						

A tabela 11 apresenta a definição da tabela **relacionamentoNodosPais**. Esta tabela é de auto relacionamento, servindo para definir quais são os nodos pais de um respectivo nodo. Esta tabela possui apenas duas colunas, que servem para armazenar a coluna de identificação única de cada nodo.

Tabela 11 – Definição da tabela relacionamentoNodosPais gerada pelo DbDesigner

relacionamentoNodosPais

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
nodosCaso_idNodosCasosFilho	**INTEGER**	PK	NN	UNSIGNED			
nodosCaso_idNodosCasosPai	**INTEGER**	PK	NN	UNSIGNED			

A tabela 12 apresenta a definição da tabela **seqProb**. Esta tabela serve para vincular a seqüência da probabilidade. Possui apenas colunas que fazem referência às correspondentes tabelas de estado e nodo, referentes ao caso de estudo.

Tabela 12 – Definição da tabela seqProb gerada pelo DbDesigner

seqProb

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
cabSeqProb_idcabSeqProb	**INTEGER**	PK	NN	UNSIGNED			
estado_idEstadoNodoPai	**INTEGER**	PK	NN	UNSIGNED			
nodosCaso_idNodosCasoPai	**INTEGER**	PK	NN	UNSIGNED			

A tabela 13 apresenta a definição da tabela **valorProbNodo**. Esta tabela serve para armazenar os valores das probabilidades que serão geradas em função da inferência dos nodos na rede. Além da coluna com o valor da probabilidade, esta tabela também possui duas outras colunas, que fazem referência ao estado e ao cabeçalho de probabilidade.

Tabela 13 – Definição da tabela valorProbNodo gerada pelo DbDesigner

valorProbNodo

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
estado_idEstadoNodo	**INTEGER**	PK	NN	UNSIGNED			
cabSeqProb_idcabSeqProb	**INTEGER**	PK	NN	UNSIGNED			
valorProbNodo	FLOAT(1.2)		NN				

A tabela 14 apresenta a definição da tabela **cabSeqProb**. Esta tabela é usada para armazenar a seqüência de probabilidades que serão geradas em função da inferência dos nodos na rede. Esta tabela possui uma coluna de identificação única que servirá para referenciar a tabela de **valorProbNodo** e o vínculo da probabilidade como caso de estudo.

Tabela 14 – Definição da tabela cabSeqProb gerada pelo DbDesigner

cabSeqProb

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
idcabSeqProb	**INTEGER**	PK	NN	UNSIGNED			AI
nodosCaso_idNodosCaso	INTEGER		NN	UNSIGNED			
seqProb	INTEGER		NN	UNSIGNED			

A tabela 15 apresenta a definição da tabela **nodosCaso**. Esta tabela pode ser definida como a tabela principal do banco de dados, em virtude de armazenar as principais informações e ligações entre as demais tabelas. Esta tabela possui uma coluna que referencia o respectivo nodo ao caso de estudo, as propriedades do nodo, prioridade do nodo, se ele é possível ou necessário na rede e os comentários. Assim como as demais tabelas, também possui uma coluna de identificação única que servirá para relacionar-se com as demais tabelas.

Tabela 15 – Definição da tabela nodosCaso gerada pelo DbDesigner

nodosCaso

tipoNodo: Simbolico "S", Numerico "N", Utilidade "U", Decisão "D" e Booleano "B"
Propriedade: Diagnostico "D" e Evidencia "E"
Prioridade possíveis: Complementar, Essencial, Excludente e Triger
PosNeces: Possivel "P" e Necessario "N"

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
idNodosCaso	**INTEGER**	PK	NN	UNSIGNED			AI
casoEstudo_idCasoEstudo	INTEGER		NN	UNSIGNED			
nodos_idNodos	INTEGER		NN	UNSIGNED			
proprEvidDiagNodosCaso	CHAR(1)						
tipoNodo	CHAR(1)						
comentariosNodosCaso	VARCHAR(2000)						
prioridadeNodosCaso	CHAR(1)						
posNecesNodoCaso	CHAR(1)						
descricaoDiagCaso	VARCHAR(2000)						

1.2 IMPORTAÇÃO DOS DADOS

Com a elaboração e desenvolvimento de um banco de dados completamente novo para o agente de DOMÍNIO, conforme a modelagem apresentada, houve a necessidade de alimentar o novo banco de dados, com os dados atualmente em uso no sistema.

O Agente de DOMÍNIO atual possui funcionalidades de exportação e importação de dados em arquivos no formato XML em sua interface, conforme apêndice C.

Em virtude do agente de DOMÍNIO atual ser uma aplicação estruturada apenas para se trabalhar em uma rede local, a sua estrutura de dados foi definida para armazenar as informações em arquivos independentes para cada rede gerada, deste modo, os dados são completamente específicos da rede, não possuindo um normalização que comporte diversas redes em uma mesma base de dados. Como conseqüência disso, a sua exportação ocorre da mesma maneira, dificultando a sua importação em uma nova estrutura do banco de dados.

Durante processo de estudo e pesquisa, para a re-engenharia foi identificada a necessidade de criar um processo independente de importação das informações do AMPLIA original para o novo projeto e, com isto, facilitar o aproveitamento das redes geradas e validadas até o momento.

Baseado no arquivo XML gerado foi possível identificar e mapear as informações necessárias para alimentar a nova estrutura, no entanto, como estão organizadas conforme a lógica do AMPLIA (Atual), seria necessário criar uma classe de importação (como a identificação dos dados) e adaptar para armazenamento e complementação das demais informações necessárias para o novo banco de dados, criando um “de / para” do sistema antigo para o atual.

Como a solução oferecida demandaria muito tempo, optou-se por desmembrar esta etapa em dois processos, ao qual seria criado um grupo que desenvolveria um módulo posteriormente à parte, e para agilizar o procedimento, optou-se por importar os dados manualmente da menor rede existente.

Para esta importação de dados, em função de ser um processo manual, foi necessário criar um dicionário de dados e um mapa de dados, possibilitando a criação de *scripts* SQL,

conforme exemplo no apêndice D, específicos para inserção de cada uma das tabelas do banco de dados.

Uma vez definida a estrutura do banco de dados e sua composição tendo sido validada com as informações de uma rede, no próximo capítulo será apresentado um estudo detalhado do *Framework* Hibernate, utilizado para persistência e comunicação com o banco de dados.

2 FRAMEWORK DE COMUNICAÇÃO COM BANCO DE DADOS

Um das características desejáveis para o novo agente de DOMÍNIO refere-se ao fato de ser uma aplicação independente do banco de dados, justamente por não se saber qual banco será utilizado ao final do projeto. Desta forma não poderá haver regras do negócio aplicadas ao banco, fato que restringiria o ambiente de forma expressiva; uma das soluções encontradas seria trabalhar na camada de persistência com um *Framework* que facilitasse e abstraísse facilmente a camada da aplicação da camada do banco de dados.

Neste capítulo serão apresentadas algumas características, do *Framework* Hibernate, solução proposta pelo grupo de pesquisa.

2.1 HIBERNATE

Atualmente existem algumas opções quanto a Framework de persistência em banco de dados, quando se trata de linguagem Java. Segundo Tate (2009), os principais Frameworks são, Hibernate, JDO, EJB e IBatis. Cada um destes Frameworks possui algumas peculiaridades e particularidades, bem como algumas vantagens e desvantagens. Como no grupo de pesquisa deste projeto, havia integrantes com conhecimento em Hibernate, optou-se por explorar este conhecimento.

O projeto Hibernate é ambicioso e pretende ser uma solução completa para o problema de gerenciamento de dados persistentes com Java. Ele cria uma interação do aplicativo com um banco de dados relacional, deixando o desenvolvedor livre para se concentrar nas regras de negócio (Bauer e King, 2005).

O Mapeamento Objeto/Relacional (ORM) é uma solução automatizada para aproximar os dois contextos: orientado a objetos e o relacional. O Hibernate é um framework de desenvolvimento da camada de persistência totalmente implementado em Java que adota a abordagem ORM. Tem como objetivo proporcionar uma elaboração de software, orientado a objetos para uma camada do software considerada crítica a qualquer sistema que necessite de uma persistência (FERREIRA, 2006)

O principal diferencial de uma aplicação que utiliza os métodos tradicionais de comunicação através de uma API JDBC (JDBC, 2009) e com o uso do Framework Hibernate,

diz respeito à produtividade, manutenção e à independência do banco de dados, pois o Hibernate atua diretamente entre a aplicação e o banco de dados (Figura 2.1).

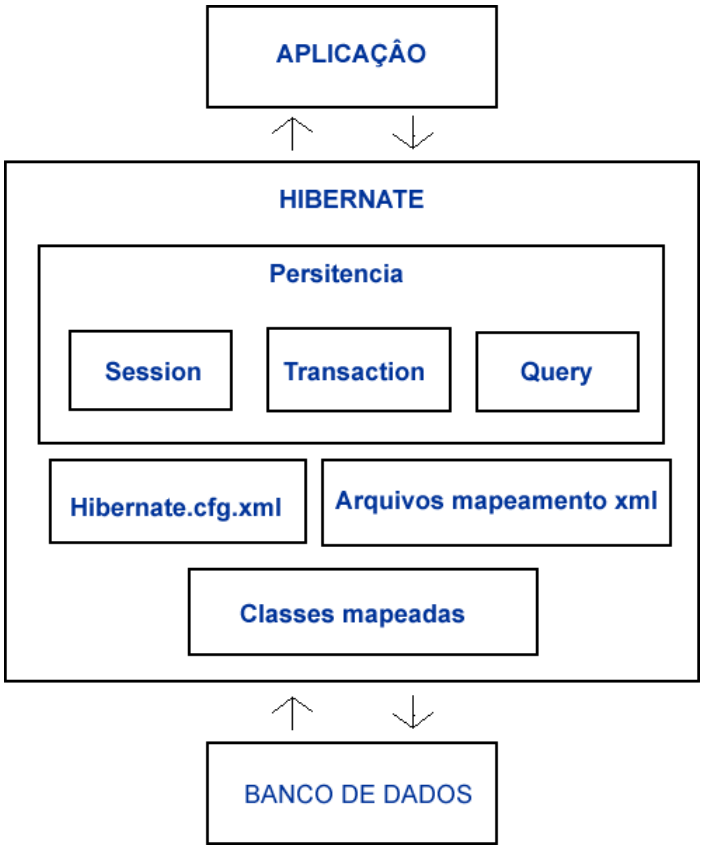

Figura 2.1 –Estrutura básica do funcionamento do Hibernate.

O uso do Hibernate garante uma maior produtividade e menor manutenção, pois possibilita menor número de códigos escritos em função de não ter uma preocupação do programador quanto à persistência destes dados. Por conseqüência, sua manutenção é facilitada e a possibilidade de erros reduzida.

De acordo com Silva

> Com menos código, é enfatizado a lógica de negócio promovendo um maior entendimento do sistema. Quando o sistema faz uso da persistência manual há sempre uma tensão, pois se o modelo de objeto sofrer alguma mudança, o modelo relacional será impactado ou vice-versa. (SILVA, 2007)

Com o uso desta tecnologia, o programador pode trabalhar com o banco de dados com os conceitos de orientação a objetos(Herança e Polimorfismo), mesmo sendo uma banco de dados puramente relacional.

O principio básico de funcionamento de um Framework de persistência é (SILVA 2007, apud LOZANO 2009):

- Tabelas de um banco de dados relacional com uma descrição do mapeamento entre o modelo de objetos em memória e seu correspondente em disco;
- Uma API para armazenamento e recuperação de objetos persistidos;
- Uma linguagem de consultas que permite determinar o subconjunto dos objetos persistentes que devem ser trazidos para a memória.

Como as demais Framework, as propriedades de cada uma das tabelas do banco de dados são armazenadas em uma classe e o seu relacionamento em um arquivo do tipo XML, conforme apresentado na tabela 16.

Tabela 16 – Exemplo tabela Mapeada pelo Hibernate

Descrição	**Correspondente**
Tabela Banco	Pessoa
Classe	Pessoa.java
Relacionamento/Herança	Pessoa.hbm.xml

Os mapeamentos e as correspondentes informações e parâmetros para conexão com o banco de dados em questão são armazenados em um arquivo chamado Hibernate.cfg.xml. Sempre que a API necessitar fazer uma conexão com o banco, será deste arquivo que serão extraídos os dados necessários para a consulta em questão. O Hibernate ainda trabalha com uma classe auxiliar que armazenará a sessão a qual estão sendo referenciados os acessos ao banco de dados, desta forma os dados são persistidos através de cada objeto de sessão criado para o acesso.

Para realizar a consulta aos dados, o Hibernate oferece como recurso uma linguagem própria para consulta o HQL, Hibernate Query Language. O HQL é uma linguagem de consulta muito semelhante ao SQL, porém, quando se faz uma consulta HQL está sendo realizada uma pesquisa na classe correspondente à tabela a ser pesquisada, e desta maneira o Hibernate se encarrega de traduzir e alimentar a classe com os dados pesquisados.

O uso do HQL possibilita as seguintes funcionalidades (Bauer e King, 2005, p.142)

- A funcionalidades de aplicar restrições a propriedades de objetos associados, relacionados por referencia, ou contidos nas coleções(para navegar usando o objeto gráfico);
- A funcionalidades de obter somente propriedades de uma entidade ou entidades, sem o custo indireto de carregar a própria entidade em um escopo transacional;
- A funcionalidades de ordenar os resultados da consulta;
- A funcionalidades de paginar os resultados;

- Agregação com *group by*, *having* e funções agregadas como *sum, min, max*;
- Associações externas ao obter múltiplos objetos por linha;
- A funcionalidades de chamar funções SQL definidas pelo usuário;
- Subconsultas (consultas aninhadas).

Através do uso do HQL, o Hibernate possibilita o uso de quatro tipos de consulta (Bauer e King, 2005, p.144).

- *Immediate Fetching* - O objeto associado é obtido imediatamente, utilizando um banco de dados de leitura seqüencial (ou pesquisa de *cache*)
- *Lazy Fetching* - O objeto ou coleção associada é buscado "preguiçosamente", quando é acessado pela primeira vez. Isso resulta em um novo pedido para o banco de dados (a menos que o objeto associado esteja em *cache*).
- *Eager Fetching* - O objeto ou coleção associada é buscado em conjunto com possuidor do objeto, nenhuma requisição adicional ao banco de dados é exigida.
- *Batch Fetching* - Esta abordagem pode ser usada para melhorar o desempenho do *Lazy Fetching* recuperando um lote de objetos ou coleções, quando uma associação *Lazy Fetching* é solicitada. (A busca em lote também pode ser usada para melhorar o desempenho do *Immediate Fetching*.)

A quebra de paradigma talvez seja o ponto culminante para o uso da tecnologia em questão, pois o programador tem que alterar a forma tradicional de raciocínio, quando se trata de acesso a dados em um banco de dados. Desta maneira, não acessa mais diretamente as tabelas, o ingresso ocorre apenas a Classe, e por conseguinte, o Hibernate se encarrega de gerenciar e acessar a respectiva tabelas do Banco de dados.

O uso do HQL se torna uma poderosa ferramenta para consulta e manipulação de dados, porém não elimina por completo o uso do SQL, através da API JDBC. Em algumas consultas mais elaboradas ou que necessite uma melhor performance não é recomendado o uso do HQL ,e sim, do SQL nativo.

3 IDE PARA O DESENVOLVIMENTO

Partindo do princípio do qual a linguagem para desenvolvimento, JAVA, já havia sido definida pelo grupo de pesquisa no qual este trabalho esta sendo baseado, houve a necessidade de escolher uma IDE de desenvolvimento. Esta escolha foi baseada nas facilidades e no perfeito casamento entre a linguagem e os *Frameworks* que serão utilizados.

Neste capítulo serão apresentadas algumas características, como pontos positivos e negativos no uso das duas IDE'S mais populares para desenvolvimento na plataforma JAVA: Eclipse (ECLIPSE, ,2009) e Netbeans (NETBEANS, 2009).

3.1 ECLIPSE

O Eclipse é uma das ferramentas de desenvolvimento de software mais polulares atualmente para o desenvolvimento em plataforma Java, sendo considerada uma das ferramentas chave em se tratando de iniciativas open-source (código aberto). Como IDE, possui facilidades como visualização de todos os arquivos contidos no projeto de forma clara, ferramentas de gerenciamento de trabalho coletivo, compilação em tempo real, geração automática de código, dentre outras (LIMA, 2009).

Seu projeto de desenvolvimento iniciou-se em 2001, sendo os principais membros do projeto na época a Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft e Webgain. Posteriormente, muitas empresas se juntaram ao projeto, tais como Oracle e JBos. Atualmente, o responsável pela continuidade do desenvolvimento da ferramenta é o Consórcio Eclipse.org (ECLIPSE, 2009), criado pela IBM, empresa responsável pelo desenvolvimento da ferramenta em sua fase inicial (LIMA, 2009).

Sua plataforma de desenvolvimento possui certa neutralidade quanto à linguagem a ser utilizada, devido aos seus sub-projetos, suportando facilmente o seu uso para desenvolvimento em C, C++, JAVA, entre outras.

O que faz desta ferramenta um diferencial é a flexibilidade proporcionada ao desenvolvedor. Ele continuamente trabalha em um *workbench*, isto é, um ambiente que pode

ser configurado conforme suas necessidades com uso de perspectivas (CARVALHO; BATISTA; ULBRICH, 2007 apud LIMA, 2009).

O Eclipse possui como forte característica a facilidade de escrita de códigos, fazendo uso de um poderoso componente chamado “Method Completion”, que sugere e completa os códigos na medida em que está sendo digitado. Outro ponto forte da plataforma é a interface de desenvolvimento, que é muito rica e de fácil utilização (figura 3.1).

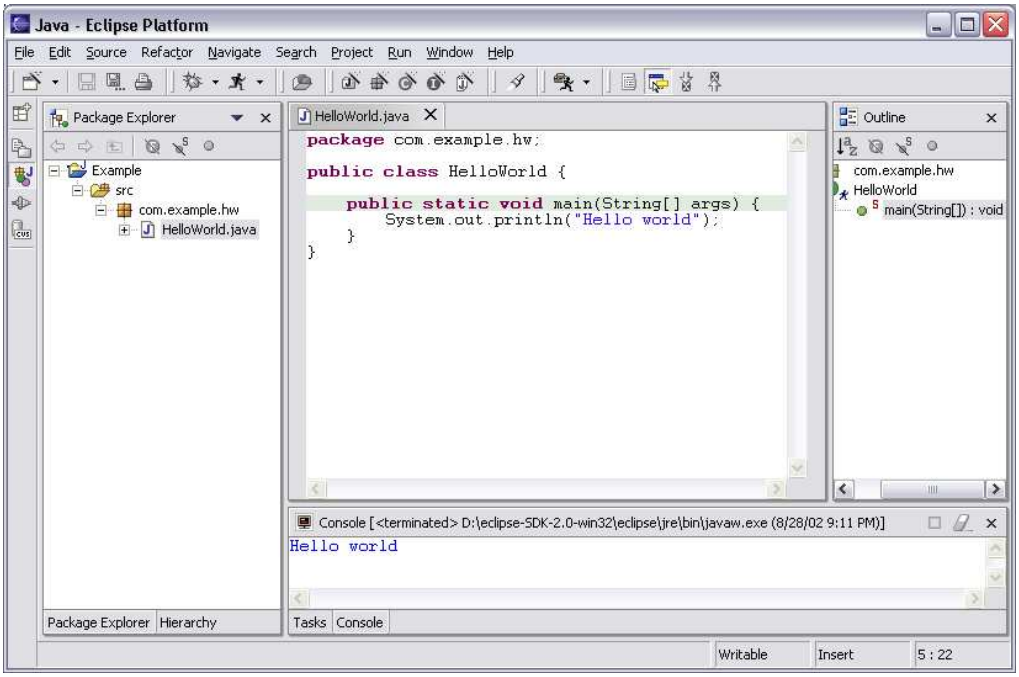

Figura 3.1 – Interface de desenvolvimento Eclipse.

Outra característica importante é a integração com servidores CVS. O eclipse já vem com plug-ins para conexão com servidor CVS, desta forma, facilitando a atualização do fonte e organização do desenvolvimento. No caso deste estudo, é muito importante este recurso, visto que o projeto é grande e o seu desenvolvimento se dá em grupos separados.

Os *plug-ins* são basicamente softwares, com funcionalidades específicas não contempladas na ferramenta em questão, que são facilmente “plugados, encaixados”, agregando funcionalidades a ferramenta (figura 3.2).

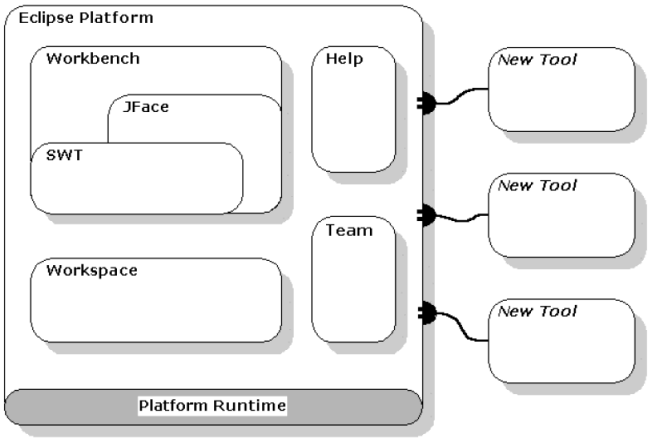

Figura 3.2 – Arquitetura da plataforma Eclipse (CARVALHO; BATISTA; ULBRICH, 2007)

3.2 NETBEANS

O NetBeans é um ambiente de desenvolvimento em Java criado pela Sun, empresa responsável pelo linguagem Java, que se mantém como seu principal apoiador. A ferramenta é muito semelhante ao Eclipse, possui seu código fonte aberto, registrado de forma livre, porém, seu principal diferencial refere-se à proposta. O NetBeans propõe-se a ser referência no desenvolvimento em Java, ao contrário do Eclipse, onde a proposta é muito mais ampla, devido aos seus subprojetos.

Seu grupo de empresas apoiadoras é vasto, contando com grandes empresas como JBoss, EBAY, Sony Ericson, Ricoh, entre outras (NETBEANS, 2009).

O NetBens possui suporte para execução em ambiente Windows, Linux e MacOS. Sua configuração é extremamente simples, sendo facilmente instalado e executado. Possui uma ampla documentação, em diversos idiomas, inclusive com suporte direto.

O ambiente de desenvolvimento NetBeans oferece suporte a uso de plug-ins, agregando, com isto, diversas funcionalidades não inclusas em sua versão original. Alguns destes plug-ins, devido a sua procura, são oferecidos previamente incorporados à ferramenta como módulos específicos, pacotes personalizados para determinados fins, como o suporte a desenvolvimento as outras linguagens, como C/C++, PHP entre outras (NETBEANS, 2009).

Como destaque em seus recursos para desenvolvimento, faz-se importante citar seu excelente editor para desenvolvimento de aplicações visuais - Swing ou AWT, como também para aplicações WEB (JSP, Servlets, etc.). Conta com uma interface rica em recursos e de fácil utilização (figura 3.3), oferecendo componentes de interface, que podem ser “arrastados

para o Form”, posteriormente configurados com uma interface gráfica e utilitários, que facilitam e agilizam o desenvolvimento, seguindo o mesmo padrão para uso de ferramentas pagas muito populares, como o Visual Estúdio da Microsoft.

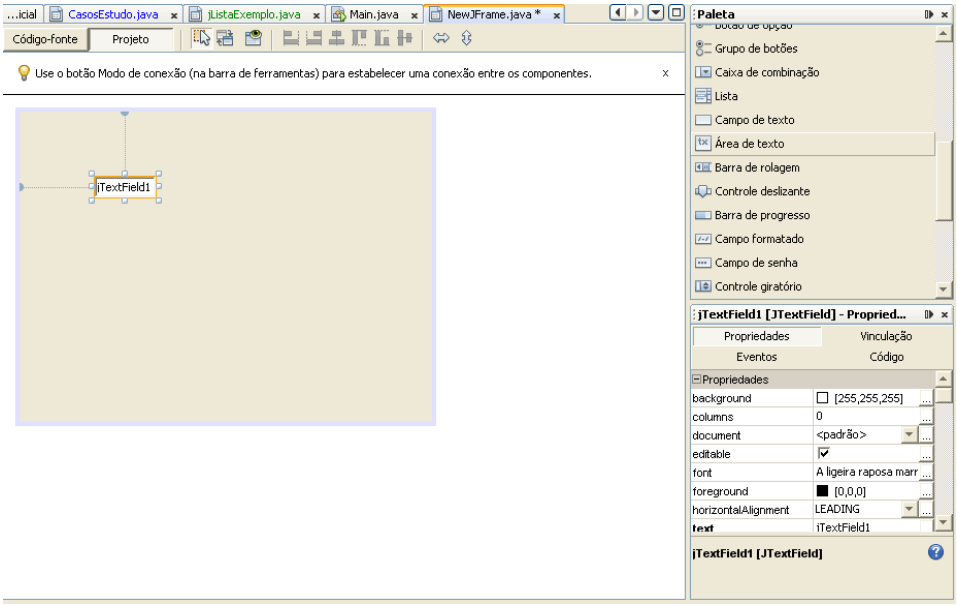

Figura 3.3 – Editor gráfico de desenvolvimento

Outro recurso oferecido como diferencial pela ferramenta, é o assistente “Wizards”, um recurso que faz o gerenciamento e criação de programas, aplicativos baseados em “templates” (figura 3.4).

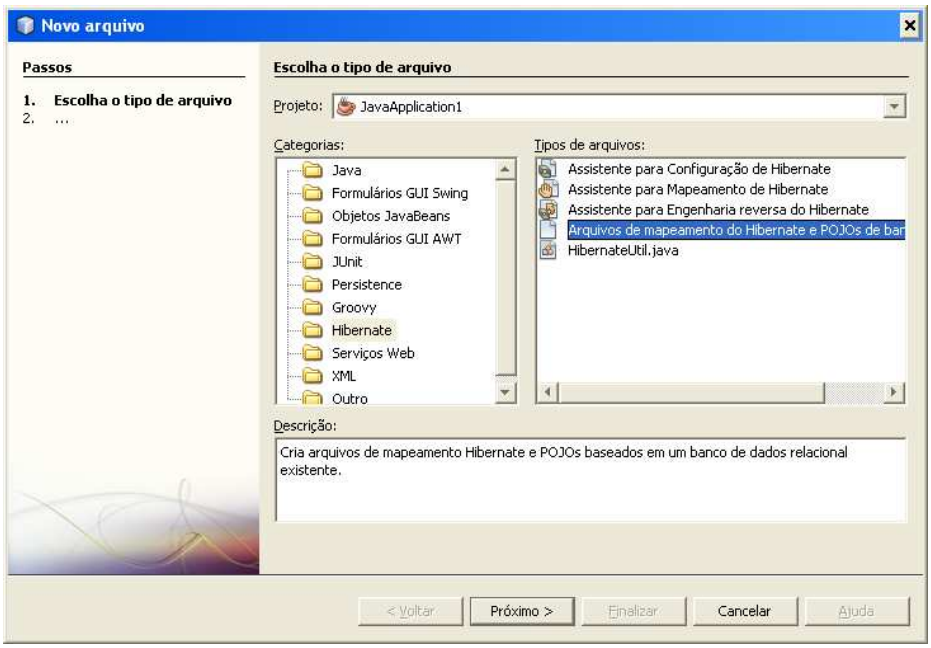

Figura 3.4 – Assistente Wizards

A ferramenta ainda oferece um recurso interessante, que vale ser destacado, o gerenciador de módulos plug-ins. Um recurso que permite o desenvolvedor realizar a

atualização, alteração ou exclusão dos módulos integrados ao ambiente de desenvolvimento de forma dinâmica (figura 3.5).

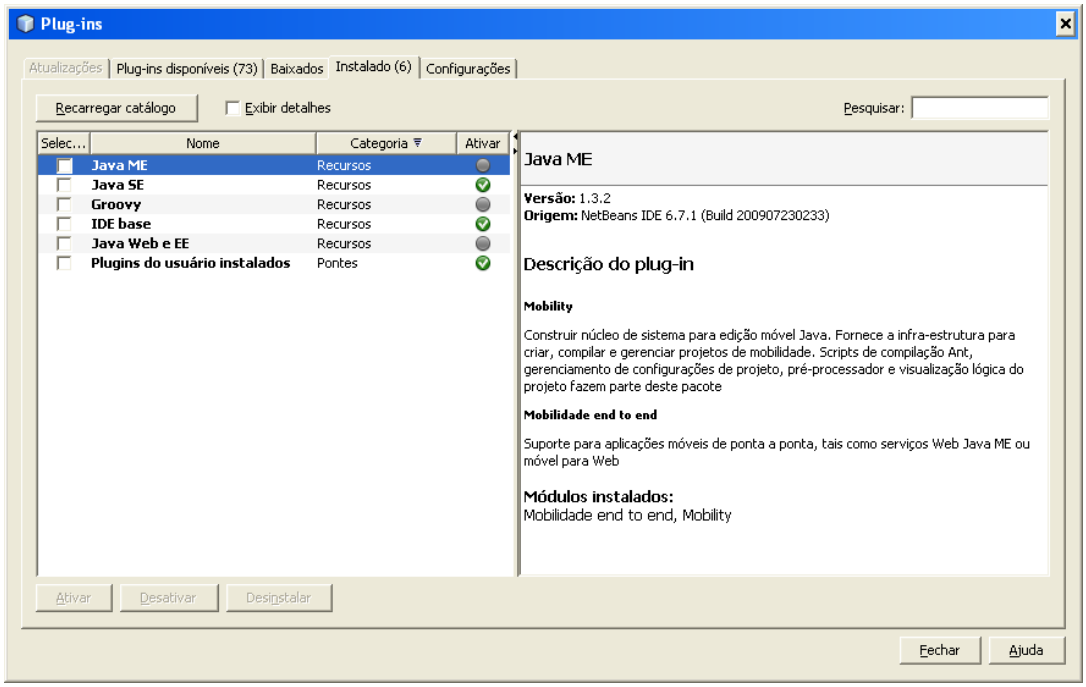

Figura 3.5 – Gerenciador de módulos.

Através do gerenciador de módulos, a ferramenta realiza uma conexão com o centro de atualizações do grupo NetBeans.org e executa uma verificação da existência de atualização em algum módulo previamente instalado na ferramenta do usuário. Caso alguma alteração de versão seja detectada ou um novo módulo seja inserido no repositório, o desenvolvedor é alertado e deve marcar a opção de atualização ou não. Dessa forma, caso selecionada a opção de atualização, a ferramenta se encarrega de atualizar a versão do módulo automaticamente para o usuário, realizando o *download* e instalação do mesmo (CARVALHO; BATISTA; ULBRICH, 2007 apud SEVERO, 2009).

3.3 INTERFACE DE DESENVOLVIMENTO UTILIZADA

As duas ferramentas estudadas são consideradas, em muitos fóruns e artigos publicados, ferramentas muito procuradas dentre os evangelistas da linguagem Java, não sendo as únicas, mas sim as principais.

Dentre as ferramentas escolhidas existem algumas particularidades diferenciais que podem facilitar e até mesmo definir o sucesso do desenvolvimento. Para sua escolha, levaram-

se em consideração alguns aspectos que seriam de vital importância para que o desenvolvimento fosse executado da melhor forma possível.

A IDE de desenvolvimento Eclipse foi a ferramenta usada inicialmente. Um dos principais fatores determinante para isto foi que durante a graduação se teve muito contato com esta IDE, sendo encontrada na maior parte dos laboratórios da FEEVALE, somando-se ao fato de dispor de inúmeras promessas de recursos disponíveis através de plug-ins.

Para escrita de códigos o Eclipse IDE se mostrou muito eficiente pela forma que seu ambiente é disponibilizado. Através do seu *Workbench*, janela principal para uso do desenvolvedor, pode-se facilmente identificar os recursos, bem como navegar entre eles, criando um ambiente facilmente organizado.

O Eclipse IDE apresenta inúmeras qualidades, dentre elas a grande quantidade de recursos obtidos através do uso de plug-ins. Porém, encontrou-se certa dificuldade com o uso de alguns destes componentes. Faz-se importante salientar que estes recursos, em muitos casos, não são extensíveis a qualquer versão IDE, e a falta de documentação acaba dificultando o seu uso.

Alguns dos componentes utilizados dependem de outros componentes e, por conseqüência, estes também têm suas dependências, bem como algumas incompatibilidades a versões do próprio JAVA. Em algumas situações acaba-se ficando preso a determinado conjunto de componentes aplicados a IDE. Nota-se claramente uma carência no controle dos recursos aplicados, e um próprio controle de versões destes componentes, dificultando muito seu uso.

Em algumas situações há a nítida sensação de que se está utilizando um "*Frankenstein*" de desenvolvimento; e a própria configuração acaba se tornando tão específica, que o desenvolvedor se sente completamente desprovido de documentação.

Esta dificuldade é tão corriqueira no mundo Eclipse, que algumas empresas identificaram esta carência e passaram a vender serviços de configuração e personalização da IDE em forma de pacotes e serviços de consultoria ao seu uso.

Uma das empresas que se teve contato foi a MyEclipse (MYECLIPSE, 2009). Esta empresa oferece uma versão da IDE e alguns componentes como forma de pacotes, previamente configurados, conforme a necessidade do desenvolvedor. Oferece juntamente a consultoria e documentação para o uso destes componentes. Esta alternativa acabou se tornando inviável devido ao seu elevado custo.

Como conseqüência, buscou-se alternativas para estas dificuldades. Após algumas pesquisas e consultas a fóruns, deu-se inicio ao estudo e leituras sobre o uso do NetBeans.

O NetBeans IDE, possui um recurso muito interessante para o problema do gerenciamento dos plug-ins. A IDE possui um componente nativo, que se encarrega do gerenciamento e controle dos recursos e compatibilidades dos demais componentes de forma automática; e através da WEB, possibilita a atualização dos demais componentes e, até mesmo da própria IDE. A disponibilidade dos componentes é controlada conforme as versões utilizadas de cada recurso, facilitando seu uso e promovendo um controle de forma eficiente e centralizado.

A interface com o usuário do NetBeans é muito semelhante ao do Eclipse, porém, nativamente ele oferece uma gama muito maior de recursos para o desenvolvimento em JAVA, como conectividade e manipulação de servidores WEB e de Data Bases.

O NetBeans possui um excelente editor gráfico - como plug-in nativo - para desenvolvimento de interfaces gráficas, recursos também encontrado no Eclipse, porém fornecidos pela comunidade em geral. Como principal diferencial neste ponto, destaca-se o editor gráfico do NetBeans, pois além de ser nativo separa completamente o código gerado pelo editor e o código gerado pelo programador; separando com marcações e bloqueios para que não seja alterado manualmente por engano e criando falhas indesejáveis no aplicativo, ou que empeçam o seu reconhecimento pelo interpretador.

O uso de um bom editor gráfico para desenvolvimento de interfaces facilita e agiliza o desenvolvimento, eliminando perda de tempo com trabalho extremamente braçal e mecânico.

Um outro ponto positivo foi a facilidade de integração e uso do plug-in do Hibernate, completando com o recurso de Wizard, oferecendo uma excelente ferramenta de configuração e de engenharia reversa para extração das tabelas e definições do banco de dados.

Um detalhe interessante é gama de documentações em diversos idiomas encontradas no próprio site do NetBeans, que se estendem de artigos e teses a simples tutoriais “passo a passo” para configuração e utilização dos recursos oferecidos pela IDE. Isto facilita bastante seu uso, em virtude de não se ter experiência nem conhecimento de nenhum dos recursos que estavam sendo utilizados para o desenvolvimento.

Em virtude do aporte teórico, dos resultados encontrados, bem como da experiência adquirida com o uso das duas IDE’s citadas, optou-s por usar o NetBeans como ferramenta para desenvolvimento do protótipo do Agente de DOMÍNIO

No próximo capítulo será apresentada a interface atual do sistema, ou seja, as telas disponíveis para cada função a ser executada.

4 INTERFACE DO AGENTE DE DOMINIO

Neste capítulo serão apresentadas algumas características da interface implementada no novo agente de DOMÍNIO.

4.1 CASO DE ESTUDO

Conforme já mencionado em capítulos anteriores, é através do agente de DOMÍNIO que será oferecida manutenção de informações que serão utilizadas pela rede bayesiana e pelos agentes. Um dos módulos principais do agente de DOMÍNIO é o módulo de manutenção do caso de estudo (figura 4.1). É através deste módulo que serão manipuladas e definidas algumas características de cada caso, através de algumas telas que serão apresentadas a seguir:

A tela de **diagnósticosPossíveis**: apresenta os diagnósticos que estarão disponíveis para modelagem da rede pelo modelo APRENDIZ.

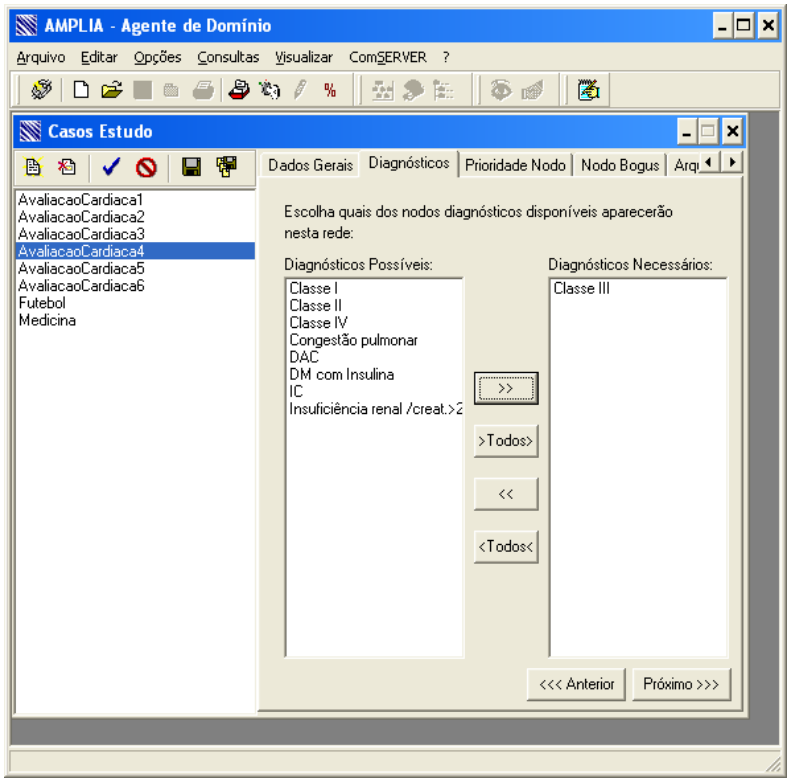

Figura 4.1 – Interface Diagnósticos

A tela de Dados Gerais (Figura 4.2), apresenta as características descritivas do caso, bem como o local onde se encontra a rede gerada.

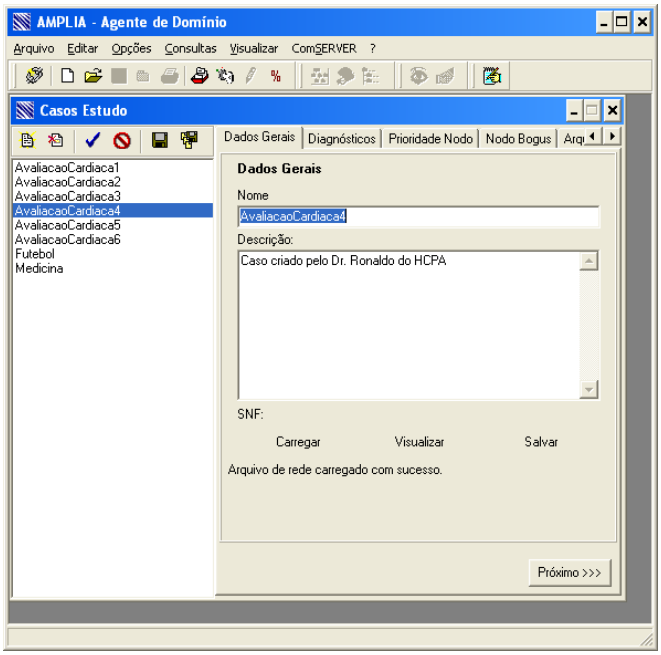

Figura 4.2 – Interface Dados Gerais

Através da interface Prioridade Nodo (Figura 4.3), serão definidas as propriedades que assumirá o nodo na rede, como complementar, essencial, excludente ou *triger*, conforme a tabela 1.

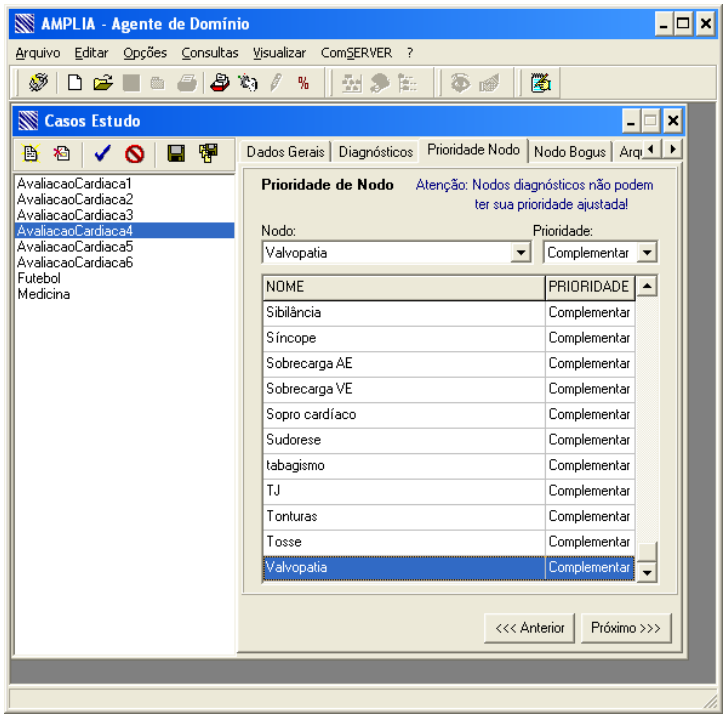

Figura 4.3 – Interface Prioridade Nodo

A tela Nodos Bogus (Figura 4.4), permite que sejam incluídos os nodos que poderão fazer parte da rede, porém não implicarão no resultado final da avaliação.

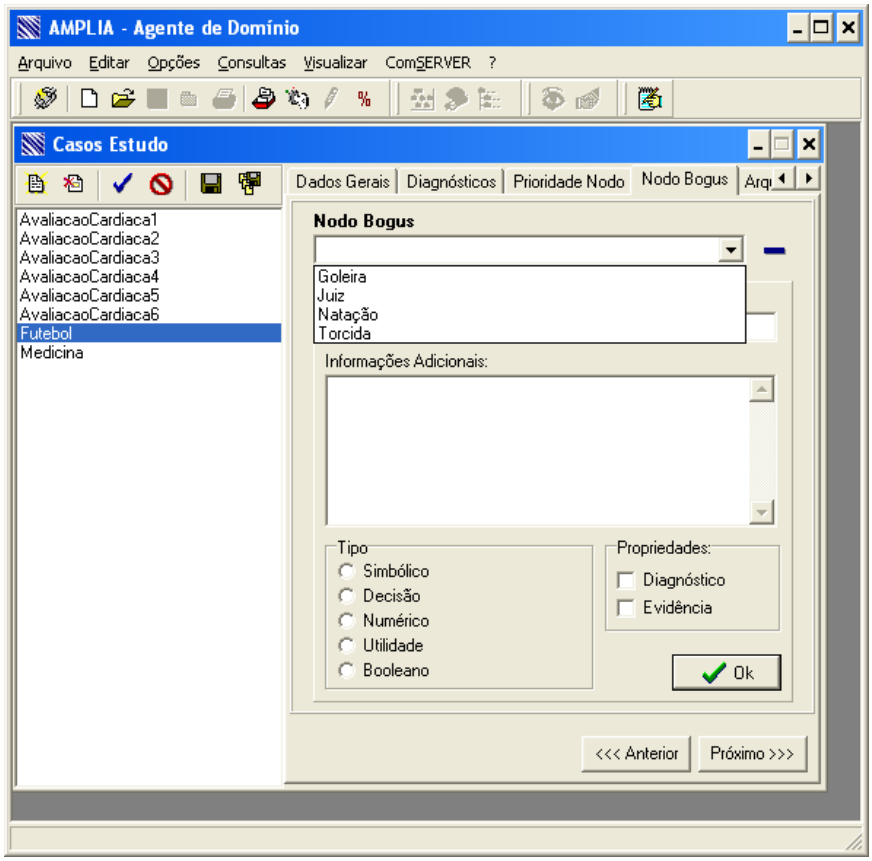

Figura 4.4 – Interface Nodos Bogos

Para estes nodos devem ser definidas algumas características como:Tipo do Nodo (Simbólico, Decisão, Numérico, Utilidade e Booleano) (FLORES 2005, p.23), e a propriedade (Diagnósticos e Evidencia), conforme tabela 17.

Tabela 17 – Possíveis classificações da propriedade do nodo

Descrição	**Tipo**
Diagnostica	Levantadas a partir de estudo
Evidencia	Necessárias para que tome uma decisão

É através da interface Arquivos (Figura 4.5), que o especialista pode inserir informações ou arquivos adicionais para orientação do aluno na construção do modelo bayesiano.

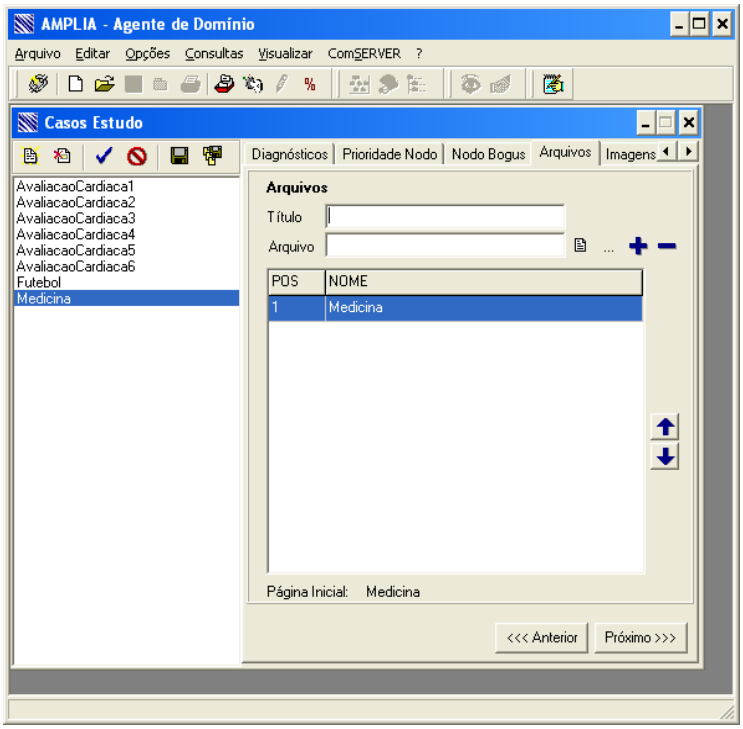

Figura 4.5 – Interface Arquivos

A interface Imagens (Figura 4.6), permite que o especialista insira imagens para orientação ou exemplificação, para servir como guia ao aluno na construção do modelo bayesiano.

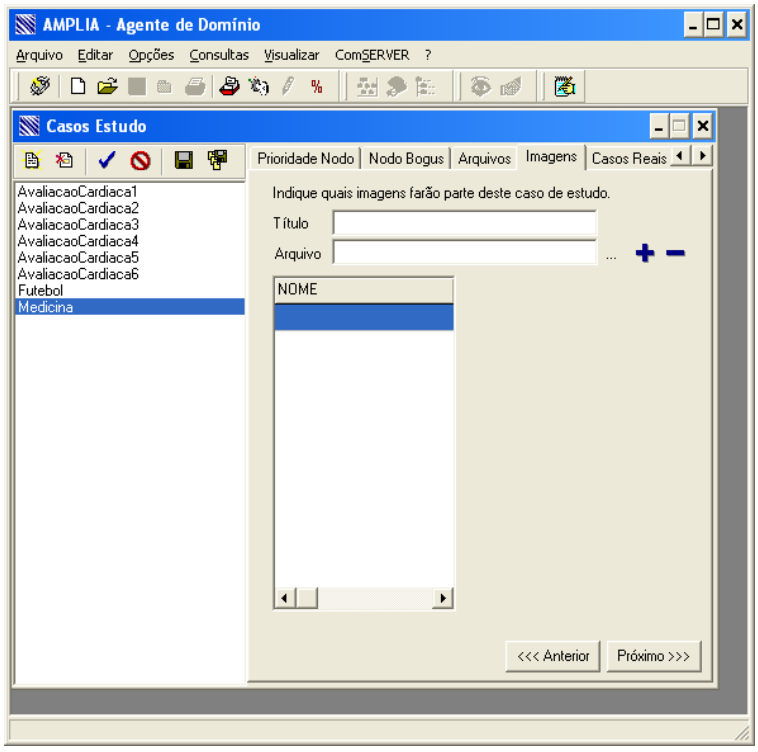

Figura 4.6 – Interface Imagens

Em Casos Reais (Figura 4.7), o especialista pode inserir informações sobre casos reais para servirem de base para a avaliação qualitativa da rede pelo agente de DOMÍNIO.

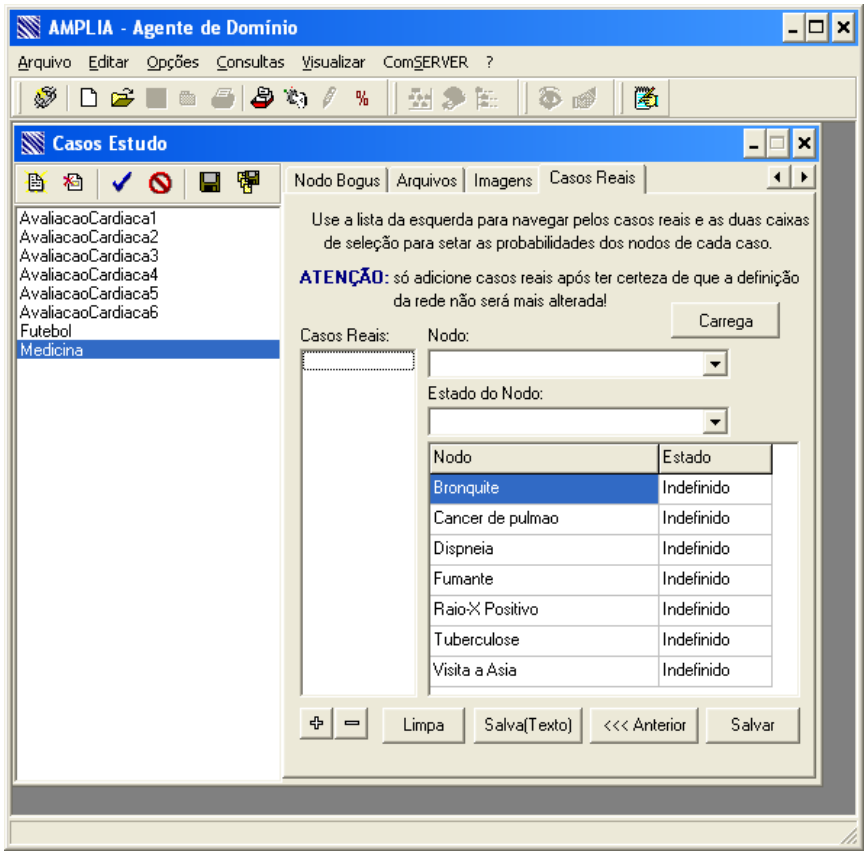

Figura 4.7 – Interface Casos Reais

4.2 INFORMAÇÕES SOBRE OS NODOS

O módulo de manutenção das informações sobre os nodos é relativamente simples, contendo a interface de Propriedades Básicas, Probabilidades Condicionais, Informações Adicionais, Diagnósticos Associados e Imagens (Figura 4.8).

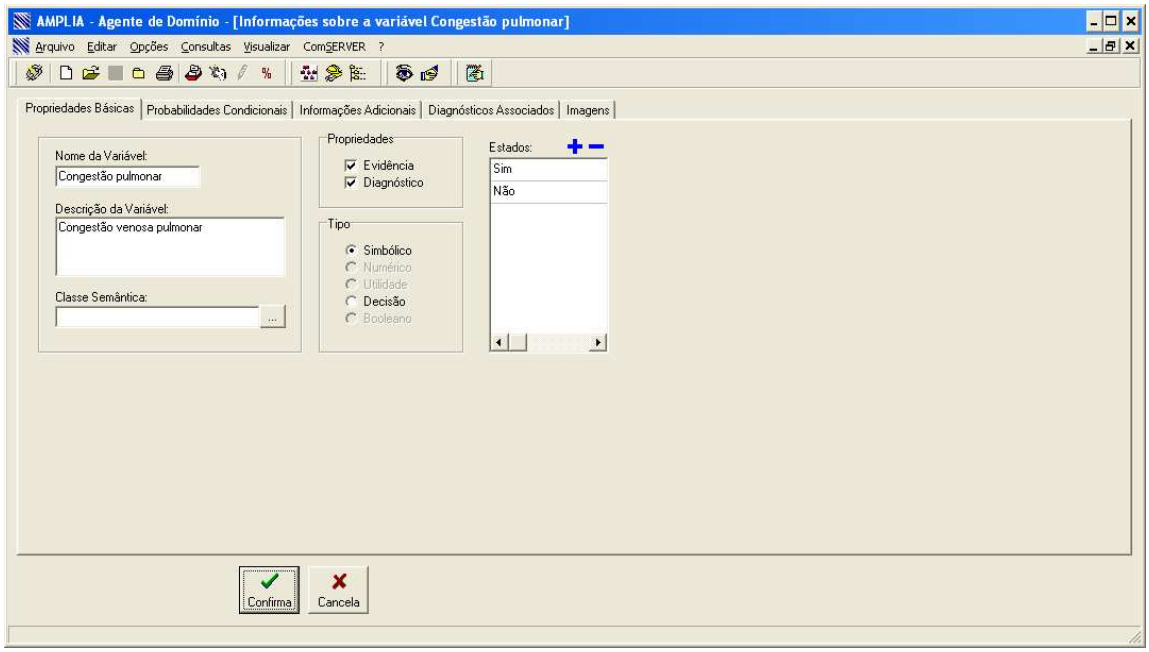

Figura 4.8 – Interface do módulo de manutenção das propriedades do Nodo

Através da interface de Propriedades Básicas será possível visualizar o nome do nodo e sua descrição. Estes campos são puramente descritivos. Assim sendo, servirão apenas como informativo para o especialista e para o aluno. Já os campos propriedades (Evidencia e Diagnóstico) (Tabela 17) e os tipos possuem dados que servirão para validar e definir os nodos na rede. O campo Estado serve para definir os estados possíveis (definidos pelo especialista) que o nodo deve assumir perante o contexto geral. E através dele que serão geradas as probabilidades na rede.

Uma segunda interface do módulo, conhecida como probabilidades condicionais (Figura 4.9), é responsável pela apresentação gráfica da relação entre as variáveis pais do nodo, ou seja, quais nodos são condicionais à existência do outro, bem como a sua probabilidade gerada pela inferência na rede bayesiana, em função dos estados (item da interface propriedade básica).

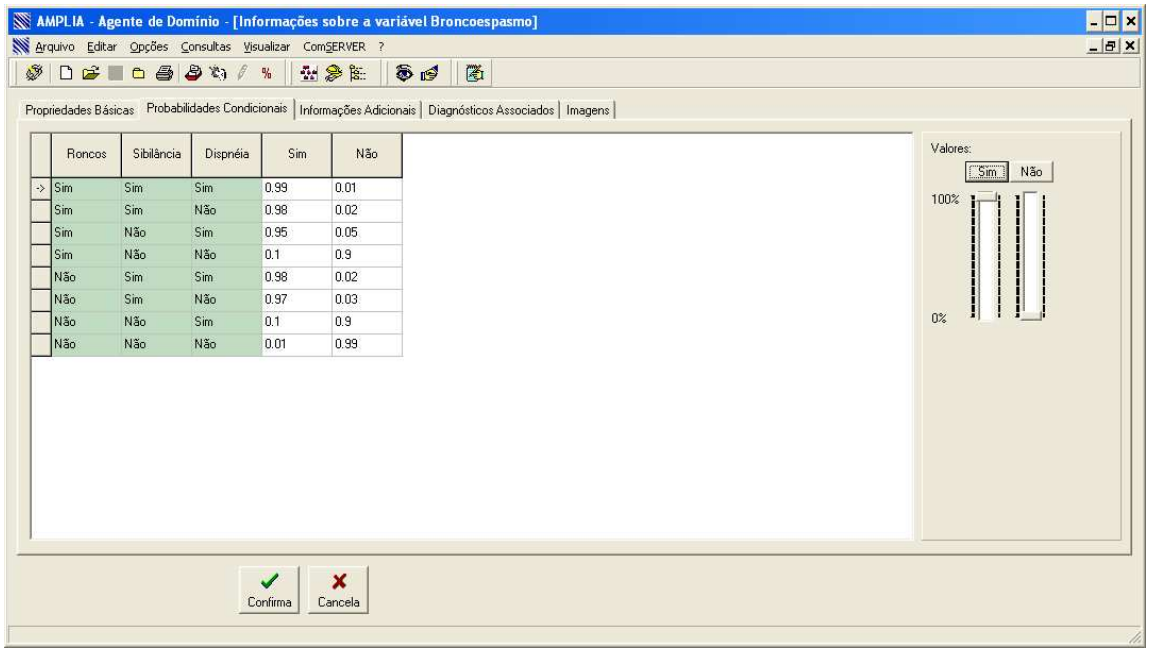

Figura 4.9 – Interfaces de Probabilidades Condicionais

Na terceira interface do módulo, Informações Adicionais (Figura 4.10), pode ser adicionado um breve descritivo referente à variável. Através desta interface o especialista pode adicionar informações complementares sobre o conteúdo desta variável, bem como os sintomas apresentados.

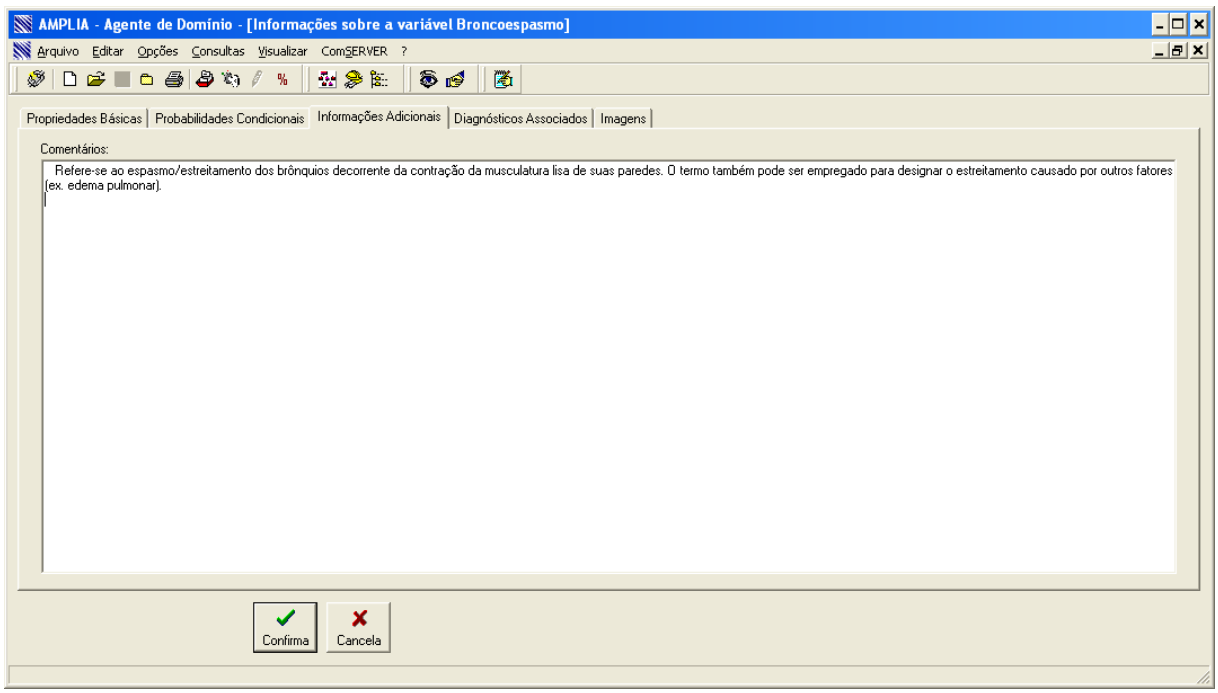

Figura 4.10 – Interface de Informações Adicionais do Nodo.

A quarta interface, Diagnósticos Associados (Figura 4.11), permite as manipulações de diagnósticos associados a variável, ou seja, outros diagnósticos que podem apresentar esta variável, como os demais casos onde serão encontrados esta variável.

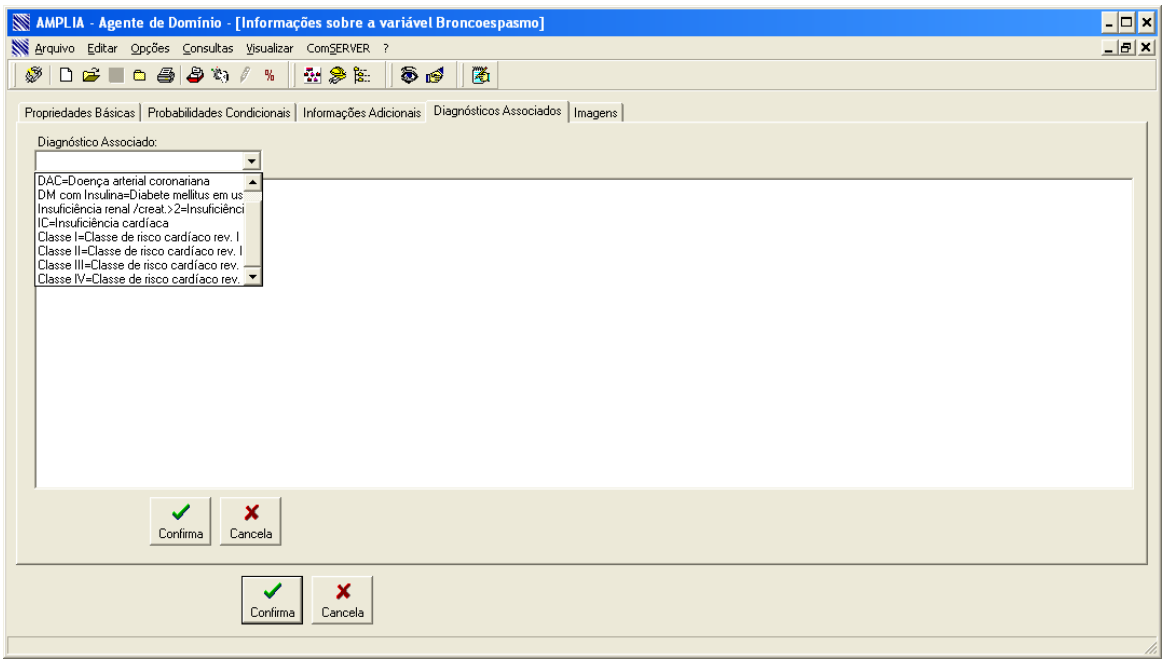

Figura 4.11 – Interface de Diagnósticos Associados ao Nodo.

A quinta e última interface, Imagens, permite armazenar imagens que se referem a variável. Estas imagens podem servir para que o aluno (Agente APRENDIZ) possa visualizar melhor o que representa o nodo.

CONSIDERAÇÕES FINAIS

Inicialmente foi buscado aporte teórico nos dados sobre o projeto AMPLIA, ao qual este trabalho esta sendo baseado, e deverá fazer parte do novo projeto, sendo destacados pontos importantes sobre o mesmo, como: seu funcionamento, sua estrutura e os princípios educacionais envolvidos na sua construção.

No primeiro capítulo, apresentou-se o protótipo do banco de dados, bem como o modelo lógico e o modelo físico; e se descreveu cada tabela e suas interações no contexto geral do ambiente. Ainda no capitulo dois, tratou-se também do processo de carga de dados de uma rede validada no ambiente original, que serviu como base para validação da base de dados proposta.

Após estudo e definição no que se refere ao banco de dados, o capítulo dois tratou da forma de acesso e persistência dos dados, e como solução foi apresentado o Framework Hibernate, um recurso que facilita a manipulação na camada de persistência do banco de dados com uso da tecnologia JAVA, promovendo a interação do aplicativo com o banco de dados relacional, agilizando os demais processos de desenvolvimento e liberando o programador para se ater apenas às regras de negócio.

No capítulo seguinte foi apresentado o Eclipse e o seu concorrente NetBeans, conhecidas como as principais IDE'S de desenvolvimento. Foram exemplificados pontos positivos e os principais recursos oferecidos pelas ferramentas em questão, bem como o motivo da escolha do Hibernate, destacando-se pelos principais fatores decisivos e benefícios com o seu uso para o desenvolvimento do projeto AMPLIA.

Uma vez preparada toda a estrutura e ambiente para desenvolvimento, no quarto capítulo foi apresentado as interfaces do Agente de DOMÍNIO, suas funcionalidades atuais e suas deficiências. Por conseguinte, foram propostas algumas alterações para melhorar a interação do uso especialista e, com isto, facilitar sua utilização.

A interface atual apresentou algumas deficiências, relatadas pelos especialistas que fizeram uso do agente de DOMÍNIO no sistema AMPLIA. Entre elas, destaca-se a dificuldade dos especialistas em lidar com grafos (nodos da rede) para representação de sintomas ou características de um determinado caso clínico.

Uma dificuldade descrita faz referência à interface de manutenção de probabilidades condicionais onde o especialista deve utilizar uma escala numérica - uma linha vertical ou

horizontal com extremos demarcando de 0% e 100% de chance, e algumas divisões numéricas entremeadas para representar dada característica.

Para a interface gráfica de manipulação e representação dos nodos, ainda não se encontrou uma melhor forma de representação gráfica, contudo, para a interface de manutenção de probabilidades condicionais, propõe-se o uso de escala de Van der Gaag. Esta escala apresentaria algumas poucas marcações (0%, 25%, 50%, 75% e 100%), com enumeração de termos mais conhecidos (improvável, pouco provável e muito provável) (SILVESTRE, 2003).

Estudo realizado por Van der Gaag e associados comprovaram que:

- Enunciados verbais facilitavam a compreensão pelo especialista do que se estava sendo solicitado. Muitos especialistas não se sentiam confortáveis com notações matemáticas de probabilidade;
- Quanto mais incertos os especialistas estivessem com relação a uma determinada probabilidade, mais propensos estariam a raciocinar em termos de palavras. As marcações verbais na escala então os auxiliavam a determinar a probabilidade que tinham em mente;
- Outra idéia nova do método é a de se agrupar as perguntas relativas a uma mesma distribuição de probabilidade para que sejam consideradas pelo especialista de forma simultânea. Os especialistas são, então, encorajados a responder as probabilidades que tenham mais certeza, geralmente os extremos e probabilidades com influências causais únicas, e então interpolar as probabilidades restantes;

A partir do trabalho exposto foi possível observar até o presente momento que ainda existe um amplo caminho a ser percorrido para se finalizar o Agente de DOMÍNIO. Até o presente momento, não foi possível implementar o uso da API UnBBayes e utilização da biblioteca JADE. A sua implementação prática, bem como as devidas alterações práticas na interface, ficará para projetos futuros.

O agente de DOMÍNIO deverá interagir diretamente com o editor de redes gráficas Bayesianas, que está sendo desenvolvido pelo aluno Felipe Scuciatto dos Santos, como trabalho de conclusão de curso.

O editor tem o intuito de viabilizar a manipulação de uma forma gráfica de representação das relações entre variáveis e suas probabilidades dentro de um escopo. Elas são representadas por grafos acíclicos, nos quais cada nó é uma variável aleatória (CHARNIAK, 1991). Cada variável deve possuir um conjunto limitado de valores (estados) e

a cada nó raiz da rede, deve ser atribuída uma probabilidade. As probabilidades dos nós não raiz devem levar em conta as probabilidades dos pais.

Por ser este um projeto de pesquisa de grandes proporções, é dividido em módulos, que serão integrados posteriormente, levando-se em consideração as particularidades de cada módulo. Esta é uma característica importante do projeto, que propiciou uma oportunidade de integração com vários profissionais de diversas áreas, possibilitando uma experiência rica durante o desenvolvimento do trabalho de conclusão.

REFERÊNCIAS BIBLIOGRÁFICAS

BAUER, C.; KING, G. ***Hibernate in Action.*** ed. MANNING 2005.

BRMODELOS. **Ferramenta para ensino de modelagem em banco de dados relacional.** Disponível em: < http://sis4.com/brModelo/Default.aspx> Acesso em: agosto. 2009.

CARVALHO, Lucas Simões de; BATISTA, Marcel Cunha;ULBRICH,Vinicius . **ANÁLISE DE FERRAMENTAS PARA DESENVOLVIMENTO DE APLICAÇÕES PARA SISTEMA OPERACIONAL SYMBIAN.** 2007. - Universidade do Estado de Santa Catarina/Centro de Ciências Tecnológicas – UDESC/CCT , Santa Catarina.

CHARNIAK, E. **Bayesian networks without tears**. AI Magazine, 1991.

CNRM. Comissão Nacional de Residência Médica – **Resolução CNRM Nº 004/2003** Disponível em: <http://portal.mec.gov.br/sesu/arquivos/pdf/cnrm_042003.pdf>. Acesso em: 17 jun. 2008.

ECLIPSE. - ***Eclipse IDE***. Disponivel em: < http://www.eclipse.org/>. Acesso em: jun. 2009.

FERREIRA, Paulo André. **DESENVOLVIMENTO DE SISTEMA DE INFORMAÇÃO WEB PARA O CONTROLE INTERNO DE PROTOCOLOS DA ESCOLA POLITÉCNICA DE PERNAMBUCO** - ESCOLA POLITÉCNICA DE PERNAMBUCO. Disponivel em: <http://dsc.upe.br/~tcc/20061/PauloAndreFerreira.pdf> Acesso em: jun. 2009

FLORES, C. D. **Fundamentos dos Sistemas Especialistas Organizado por: Dante Augusto Couto Barone Sociedades Artificiais: A Nova Fronteira da Inteligência nas Máquinas:**ed. 1 ed., Porto Alegre:, Bookman (ArtMed), 2002, v. 1, p. 127-154.

FLORES, Cecília D. **Negociação Pedagógica Aplicada a um Ambiente Multiagente de Aprendizagem Colaborativa**. 2005. 121p. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, PPGC / UFRGS, Porto Alegre.

GLUZ, J. C. **A Biblioteca FACIL (FIPA-ACL Interface Library)**: Uma Avaliação das Plataformas de Comunicação FIPA e Especificação de uma Interface de Programação FIPA Independente de Linguagem de Programação. Porto Alegre: PPGC - Instituto de Informática - UFRGS, 2002 (Trabalho Individual em Nível de Doutorado).

GLUZ, J. C. **Formalização de Comunicação de Conhecimentos Probabilísticos em Sistemas Multiagentes: Uma Abordagem Baseada em Lógica Probabilística**. 2005. 237f.

Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

GLUZ, J. C. **Linguagens de Comunicação entre Agentes: Fundamentos e Propostas de Padronização.** 2002. Trabalho Individual (Doutorado em Ciência da Computação) –Instituto de Informática, UFRGS, Porto Alegre.

GLUZ, J. C.; FLORES, Cecilia D.; VICCARI, Rosa M. **Formal Aspects of Pedagogical Negotiation in AMPLIA System**. In Nadia Nedja; Luiza M. Mourelle; Nival N. de Almeida; Mario N. Borges. (Org.). Intelligent Educational Machines. New York: Springer. v. 44, p. 117-146, 2006.

JADE. ***Java Agent Development Framework***. Disponível em: < http://jade.tilab.com//> Acesso em: mar. 2009.

JDBC – **API JDBC** - Disponível em: <http://java.sun.com/products/jdbc/download.html> Acesso em: out. 2009.

LIMA, Lucas Albertins de. *Eclipse tools* - **Ferramenta para auxilio à composição dinâmica de software**. - Universidade Federal de Campina grande. Disponível em: <http://www.dsc.ufcg.edu.br/~pet/Artigos/ARTIGO_ECLIPSETOOLS.pdf>. Acesso em: 20/09/2009.

LOZANO F**. Persistência com Hibernate.** Java Magazine, Ed. 28, p. 18-28, 2006.

MACEDO, Arthur R. de. **Resolução CNE/CES 4/2001.** Diário Oficial da União, Brasília, 9 de novembro de 2001. Seção 1, p.38

MYECLIPSE. - ***Java EE / J2EE IDE for the open source Eclipse platform*** -Disponível em: <http://www.myeclipseide.com/> Acesso em: out. 2009.

NETBEANS. - ***NetBeans open-source and free IDE***. Disponivel em: < http://www.netbeans.org/ >. Acesso em: jul. 2009.

PRODANOV, Cleber Cristiano. Manual de Metodologia Científica. 3.ed. Novo Hamburgo: Feevale, 2006. 77p.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial.** Rio de Janeiro: Campus, 2004. 1021p.

SEAMED. **Sistemas especialistas para a área médica.** Disponível em: <http://www.inf.ufrgs.br/~dflores/seamed/default.htm//> Acesso em: mar. 2009.

SILVA, Carolina Fernanda. - **ANÁLISE E AVALIAÇÃO DO FRAMEWORK HIBERNATE EM UMA APLICAÇÃO CLIENTE/SERVIDOR.** Disponível em : <http://bibdig.poliseducacional.com.br/document/?down=8 em 15/10/2009> Acesso em: set. 2009.

SILVESTRE A. M. **Raciocínio Probabilístico Aplicado ao Diagnóstico de Insuficiência Cardíaca Congestiva** (ICC). 2003. 89 p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

SUN Microsystems. **Java**. Disponível em: <HTTP://java.sun.com/>. Acesso em: 09 nov. 2008.

TATE, Bruce. ***Persistence strategies.*** *IBM Technical library*. Disponível em: http://www.ibm.com/developerworks/java/library/os-lightweight6/index.html?ca Acesso em: outubro. 2009.

UnBBayes. **Open source software for modeling, learning and reasoning upon probabilistic networks.** Disponível em: < http://unbbayes.sourceforge.net/index.html /> Acesso em: mar. 2009.

VICCARI, R. M. et all. ***A Multi-Agent Intelligent Environment for Medical Knowledge.*** Artificial Intelligence in Medicine. Elsevier Science B. V., v. 27, p. 335-366, 2003.

W3C. ***Extensible Markup Language*** (XML). Disponivel em: <http://www.w3.org/XML/>. Acesso em: 22 mai. 2009.

APÊNDICES

APÊNDICE A – AMBIENTE MULTIAGENTE PROBABILÍSTICO INTELIGENTE DE APRENDIZAGEM – AMPLIA

Neste apêndice será descrito o projeto AMPLIA, apresentando uma visão geral sobre o sistema, seus princípios norteadores e sua estrutura de funcionamento. Os assuntos mais específicos, citados neste capítulo, e que não interferem diretamente no desenvolvimento deste trabalho, não serão abordados de forma aprofundada. Para maiores informações acerca dos mesmos devem ser consultados (FLORES, 2005), (VICCARI, 2003), (GLUZ, 2006) e (SILVESTRE, 2003).

O grupo de pesquisa atualmente investigando o sistema AMPLIA e que participou do desenvolvimento deste documento foi Dra. Rosa Maria Vicari (UFRGS), Dra. Cecília Dias Flores (UFCSPA), Michele Silva (UFRGS), Julia Marques (UFRGS), Marta Rosecler Bez (FEEVALE), Regis Leandro Sebastiani (FEEVALE), Paulo Ricardo Muniz Barros (FEEVALE), Felipe Scuciatto dos Santos (FEEVALE).

1. Contextualização

O projeto AMPLIA – Ambiente Multiagente Probabilístico Inteligente de Aprendizagem – é resultado do trabalho desenvolvido pelo Grupo de Inteligência Artificial do Instituto de Informática da UFRGS. Consiste em um *software* que propõem um ambiente de ensino de medicina onde não ocorra simplesmente a transferência de conhecimento do especialista (no caso o professor) para o aluno; e sim uma negociação, com a possibilidade de diálogo, argumentação ou colaboração para a construção do conhecimento de ambos os lados (FLORES, 2005).

Parte-se do princípio que o conhecimento do professor não é o único correto acerca de um determinado assunto, tendo como base o fato deste também ter sido construído em algum momento. Sendo assim, é uma das possíveis soluções para a resolução de um problema. Leva-se em consideração, também, que o aluno pode apresentar argumentos relevantes não observados até então para encontrar a solução de um problema. Perante este cenário o software apresenta um mediador que tem como principal objetivo conseguir que as expectativas de um em relação ao outro (aluno e professor) se confirmem e que um grau de confiança elevado entre as partes seja estabelecido.

O AMPLIA também tem o objetivo de auxiliar as escolas de medicina brasileiras a adaptarem seus currículos às mudanças de diretrizes curriculares para os cursos de graduação em Medicina, aprovadas pelo Conselho Nacional de Educação, que vem sendo implantadas desde 2003 através do Programa de Incentivo às Mudanças Curriculares nas Escolas Médicas, o PROMED. Segundo estas diretrizes, é esperada "a utilização de metodologias de ensino que favoreçam a participação mais ativa do aluno na construção do conhecimento" (MACEDO, 2001), ou seja, aproximar as práticas de medicina do ambiente de aprendizado. O aluno, durante sua formação, deve poder construir modelos diagnósticos de enfermidades, incluindo as prováveis causas, sintomas associados e avaliar a aplicação do modelo construído. Desta forma, o aluno consegue desenvolver o seu raciocínio diagnóstico enquanto elabora e aplica estratégias de ação sobre determinada situação.

Conforme a resolução 04/2003 do CNRM, são consideradas atividades teórico-complementares, entre outras, a discussão de artigos científicos, cursos, palestras e seminários pertinentes ao assunto (CNRM, 2008). Entretanto, torna-se cada vez mais comum a utilização de sistemas informatizados, como tutores inteligentes, no apoio ao ensino e aprendizagem (FLORES, 2005). O AMPLIA pode ser considerado um exemplo destes sistemas.

Em suma, o *software* conta com duas frentes de ação. Na primeira, um especialista insere no sistema um caso clínico que servirá de estudo e que, na maioria das vezes, é real, contendo dados e características de um determinado paciente (como história clínica, sintomas e dados laboratoriais). Com base neste caso clínico, o especialista monta o seu diagnóstico estruturado em uma rede bayesiana, o qual é armazenado no sistema. Terminado este processo, os alunos têm a oportunidade de analisar o caso clínico em estudo e, com base nos seus conhecimentos previamente adquiridos e no princípio da negociação pedagógica, criar o seu diagnóstico, apresentando-o ao sistema para ser comparado com o do professor.

O foco da negociação pedagógica do AMPLIA é o processo de ensino e aprendizagem. O modelo empregado no sistema é o de negociação baseada em argumentação aplicada à aprendizagem colaborativa, sendo que a argumentação contempla aspectos cognitivos como crenças, ações e níveis de confiança entre aprendiz e mestre. No projeto, a negociação é empregada para alcançar um ponto de equilíbrio entre as expectativas dos alunos e do professor. Isto se obtém mediante uso intenso da argumentação nas estratégias pedagógicas empregadas no projeto. Este processo ocorre em ciclos, onde cada ciclo é iniciado quando o aluno submete seu modelo de diagnóstico para avaliação.

Desta forma, observa-se que as partes envolvidas na negociação são o aluno e o especialista, ambos representados por seus agentes de software, e um terceiro agente,

considerado o mediador e responsável por conduzir os outros dois a um entendimento comum.

Segundo Flores (2005), "*o principal objetivo a ser alcançado pela negociação pedagógica proposta no projeto AMPLIA é o estabelecimento e a confirmação de um alto grau de confiança entre os participantes do processo*".

Um exemplo de uso do AMPLIA pode ser visto na Figura 1. Para Flores (2005), "*as diferenças entre as redes são tratadas por meio de estratégias pedagógicas baseadas na interação e na negociação entre os agentes inteligentes do sistema e o aluno*".

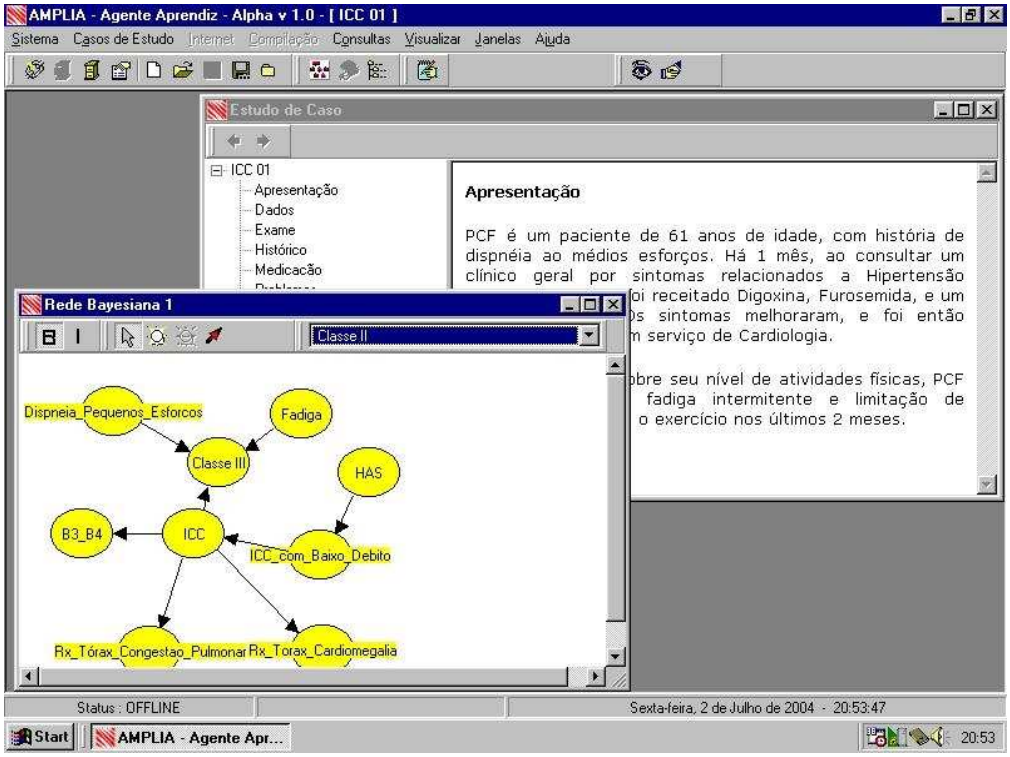

Figura 1 – Interface do AMPLIA apresentando um Estudo de caso e a rede do aluno

Fonte: (FLORES, 2005, p. 78)

Vale destacar que ambos os modelos são validados em uma base de dados com casos reais, e o objetivo principal é fazer com que o aluno desenvolva a capacidade de raciocínio diagnóstico, apresentando, ao final do processo, um diagnóstico o mais semelhante possível ou até melhor do que aquele elaborado pelo especialista.

Conforme mencionado anteriormente, para a elaboração do diagnóstico do especialista e do raciocínio diagnóstico do aluno, são utilizadas as redes bayesianas, que são grafos acíclicos orientados, compostos por nós e arcos, sendo consideradas uma das maiores tendências da área de Inteligência Artificial (IA). Os nós representam as variáveis envolvidas no cenário, atribuindo-lhes um valor de probabilidade condicional (um peso de decisão) e os arcos, que unem estes nós, representam a dependência probabilística entre as variáveis

associadas. Estas redes também podem ser chamadas redes probabilísticas, pois através delas consegue-se trabalhar com incertezas (SILVESTRE, 2003).

Russel (2004) afirma que as pessoas enfrentam situações que envolvem incerteza freqüentemente. Exemplos são os diagnósticos médicos, que, em grande parte das vezes, são elaborados através da análise da probabilidade das variáveis associadas aos casos médicos. O mesmo autor ainda afirma que uma rede bayesiana pode ser vista como: "*uma representação da distribuição de probabilidade conjunta*" e "*como uma codificação de uma coleção de declarações de independência condicional*".

Silvestre (2003) também destaca que a representação gráfica das redes bayesianas é uma ótima ferramenta a ser empregada na aquisição de conhecimentos e em processos de verificação.

2 Arquitetura multi-agente

O software foi construído como um sistema multiagente. Segundo Russell (2004), agente é qualquer entidade que consiga coletar informações a respeito do que está acontecendo no meio em que se encontra através de sensores, processar estas informações e promover alguma ação sobre o meio através de atuadores, mantendo uma racionalidade, ou seja, tendo como objetivo maximizar o seu desempenho. Um agente humano, por exemplo, tem os cinco sentidos como sensores e, através do processamento das informações captadas, provoca uma reação qualquer como mover as pernas, braços ou simplesmente expressar um sentimento. Da mesma forma, um agente de software pode, por exemplo, receber, mediante seus sensores, dados digitados ou movimentos do mouse, processar estas entradas, decidir o que fazer e exibir algo na tela ou realizar qualquer outra tarefa como criar um arquivo ou mandar uma mensagem através da rede.

Os sistemas multiagentes são compostos por mais de um agente que apresentam características independentes, ou seja, tem um comportamento próprio em relação aos outros agentes do sistema, mas que, obrigatoriamente, conseguem se comunicar com os demais utilizando protocolos de comunicação com pelo menos uma das características a seguir: coordenação, cooperação, competição e/ou negociação. Como o foco deste trabalho é apenas investigar o funcionamento do software AMPLIA, os conceitos de IA não serão abordados profundamente. Para maiores informações sobre o assunto, pode ser consultado (RUSSELL, 2004).

O AMPLIA apresenta em sua estrutura básica a existência dos três agentes listados a seguir: Agente APRENDIZ; Agente de DOMÍNIO; Agente MEDIADOR. A estrutura pode ser vista na figura 2. Cada agente destes tem sua estrutura dividida em três níveis distintos: Nível de Interação; Nível Operacional; Nível de Decisão. No nível de Interação, nível mais básico de todos os agentes, encontram-se todas as ações e atividades que o agente precisa realizar para se comunicar com o meio no qual se encontra. Já no nível Operacional, estão disponíveis as ações e atividades condizentes com a separação e classificação das informações que vem do nível de Interação, repasse das mesmas para o nível de Decisão e planejamento e execução das instruções recebidas do nível de Decisão. O nível de Decisão, conforme já dito, é responsável por tomar as decisões do que será feito, ou seja, neste nível é feita uma avaliação do cenário através dos dados recebidos dos outros níveis do agente e é retornado aos níveis inferiores quais atitudes tomar com relação a um determinado estado.

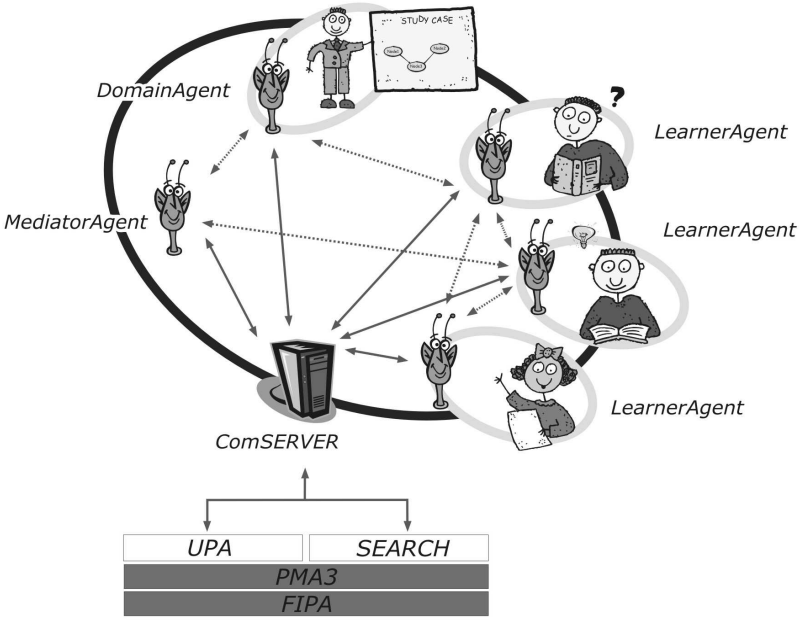

Figura 2 – Esboço da estrutura multiagente do AMPLIA

Fonte: (FLORES, 2005)

O agente APRENDIZ representa o aluno durante a realização das atividades propostas no nível virtual. Nele estão contidas todas as informações quanto ao estado do seu processo de aprendizagem, bem como, o grau de autoconfiança declarado pelo aluno. Este agente também procura deduzir o nível de autonomia que o aluno apresenta. Esta autonomia é verificada pela análise de um *log* que é gerado por uma interface gráfica colaborativa a respeito da sua utilização por parte do aluno. Este *log* (apresentado na figura 3) guarda informações de como o aluno escolheu os elementos da rede, de como as ligações destes

elementos foram realizadas, quantas vezes estas foram refeitas, entre outros elementos. A autoconfiança pode ser baixa, média ou alta. O nível de autonomia atribuído pelo sistema também se enquadra nestas três faixas.

```
AMPLIA LOG - Sessao iniciada em 15/07/2004 - 18:21:53
INSERÇÃO NODO SIMBÓLICO (15/07/2004 - 18:43:30) : B3_B4
INSERÇÃO NODO SIMBÓLICO (15/07/2004 - 18:43:57) : Rx_Tórax_Congestao_Pulmonar
INSERÇÃO NODO SIMBÓLICO (15/07/2004 - 18:44:11) : Rx_Torax_Cardiomegalia
INSERÇÃO NODO SIMBÓLICO (15/07/2004 - 18:45:26) : Dispneia_Pequenos_Esforcos
INSERÇÃO NODO SIMBÓLICO (15/07/2004 - 18:46:11) : Fadiga
INSERÇÃO NODO SIMBÓLICO (15/07/2004 - 18:47:26) : Fadiga
EXCLUSÃO NODO SIMBÓLICO (15/07/2004 - 18:47:30) : Fadiga
INSERÇÃO NODO SIMBÓLICO (15/07/2004 - 18:47:47) : ICC
INSERÇÃO SETA (15/07/2004 - 18:48:07) DE Fadiga ATE ICC
INSERÇÃO SETA (15/07/2004 - 18:48:09) DE Dispneia_Pequenos_Esforcos ATE ICC
INSERÇÃO SETA (15/07/2004 - 18:48:11) DE B3_B4 ATE ICC
INSERÇÃO SETA (15/07/2004 - 18:48:12) DE Rx_Tórax_Congestao_Pulmonar ATE ICC
INSERÇÃO SETA (15/07/2004 - 18:48:14) DE Rx_Torax_Cardiomegalia ATE ICC
EXCLUSÃO SETA (15/07/2004 - 18:49:05) DE Fadiga ATE ICC
EXCLUSÃO SETA (15/07/2004 - 18:49:26) DE B3_B4 ATE ICC
EXCLUSÃO SETA (15/07/2004 - 18:49:47) DE Dispneia_Pequenos_Esforcos ATE ICC
EXCLUSÃO SETA (15/07/2004 - 18:50:17) DE Rx_Torax_Cardiomegalia ATE ICC
EXCLUSÃO SETA (15/07/2004 - 18:50:23) DE Rx_Tórax_Congestao_Pulmonar ATE ICC
INSERÇÃO SETA (15/07/2004 - 18:51:11) DE ICC ATE Rx_Tórax_Congestao_Pulmonar
INSERÇÃO SETA (15/07/2004 - 18:51:13) DE ICC ATE B3_B4
INSERÇÃO SETA (15/07/2004 - 18:51:15) DE ICC ATE Rx_Torax_Cardiomegalia
INSERÇÃO NODO SIMBÓLICO (15/07/2004 - 18:52:34) : Classe III
INSERÇÃO SETA (15/07/2004 - 18:52:43) DE Fadiga ATE Classe III
INSERÇÃO SETA (15/07/2004 - 18:52:45) DE Dispneia_Pequenos_Esforcos ATE Classe III
INSERÇÃO SETA (15/07/2004 - 18:52:53) DE ICC ATE Classe III
AMPLIA LOG - Sessao finalizada em 15/07/2004 - 18:53:32
```

Figura 3 – Exemplo de log de alunos do AMPLIA

Fonte: (FLORES, 2005, p. 60)

A interatividade dos agentes, entre si e com o aluno, segue um protocolo de interação/conversação pré-definido. O primeiro passo é o repasse de um estudo de caso do agente de DOMÍNIO para o agente APRENDIZ. Este, por sua vez, guarda os dados necessários sobre aquele caso de estudo e apresenta o mesmo ao aluno. Após a análise do caso recebido, o aluno monta a sua hipótese diagnóstica que é enviada ao agente de DOMÍNIO através do agente APRENDIZ a fim de ser avaliada (o processo de avaliação será descrito nos próximos parágrafos). O resultado deste ajuizamento, mais especificamente os pontos divergentes da hipótese diagnóstica do aluno para a do especialista, são passados para o agente MEDIADOR. Este agente, com base na avaliação do agente de DOMÍNIO, do nível de autoconfiança do aluno e no nível de autonomia inferido pelo agente APRENDIZ acerca do aluno, decide qual estratégia pedagógica melhor se enquadra naquela situação e a aplica. O aluno recebe uma mensagem do agente MEDIADOR com base na estratégia escolhida e opta em argumentar, alterando sua hipótese diagnóstica e a enviando novamente para avaliação, ou abandona o processo de aprendizagem.

É importante destacar que quando um caso clínico que será estudado é apresentado ao aluno, algumas restrições são definidas em função do alto nível de complexidade

envolvido na verificação dos resultados. Uma destas restrições é a limitação das variáveis que podem ser utilizadas em cada caso.

O agente de DOMÍNIO e o agente MEDIADOR dividem a tarefa do tutor. O agente de DOMÍNIO apresenta o nível de conhecimento mais amplo a respeito do caso clínico tratado. Nele estão associados a rede bayesiana desenvolvida pelo especialista (ver exemplo na figura 4) e a base de dados com casos reais.

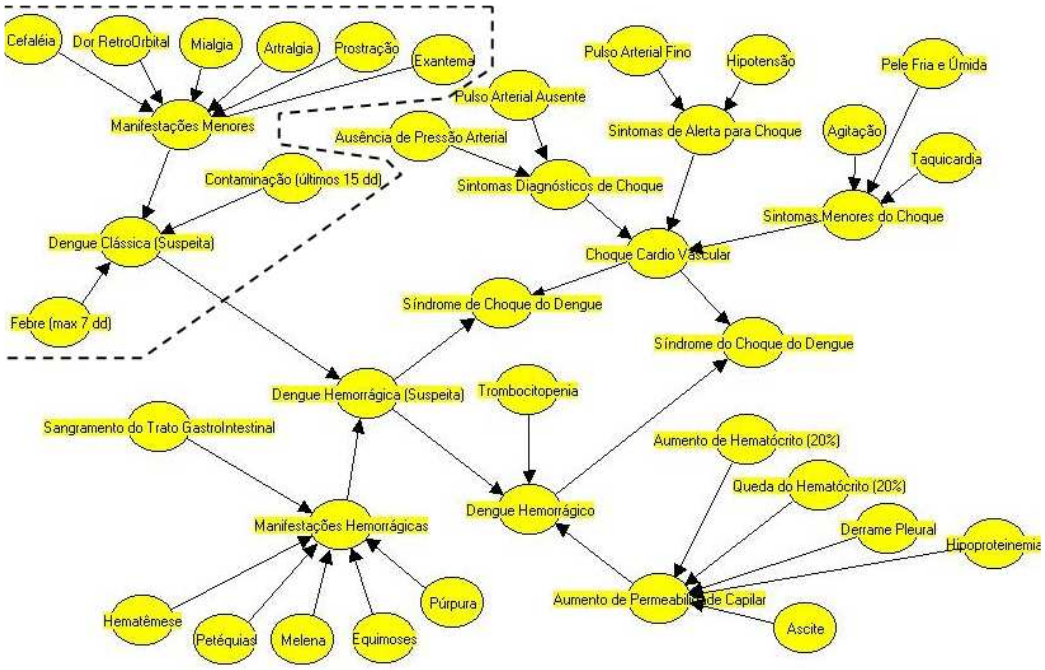

Figura 4 – Exemplo de modelo bayesiano construído pelo especialista e uma simplificação

Fonte: (FLORES, 2005, p. 64)

Desta forma, este agente é responsável pela avaliação da rede bayesiana desenvolvida pelo aluno. Esta avaliação é feita através de verificações qualitativas e quantitativas.

A avaliação qualitativa realizada pelo AMPLIA apresenta os seguintes passos: *simplificação do modelo bayesiano do especialista* – são eliminados todos os nodos inseridos pelo especialista em um modelo de DOMÍNIO geral que não se aplicam a um determinado caso clínico; *análise das relações* – os possíveis relacionamentos errados, invertidos ou faltantes no modelo bayesiano do aluno são conferidos; *análise dos nodos* – identifica os nodos que estão faltando ou sobrando na rede do aluno em relação à rede do especialista e classifica-os conforme a tabela 1.

Tabela 1 – Classificações possíveis para os nodos da rede

Diagnóstico	**Deve estar sempre presente**
Trigger	Quando presente, seleciona o diagnóstico como solução potencial

Essencial	Deve estar presente para assegurar a identificação do diagnóstico
Complementar	Sua presença aumenta a probabilidade do diagnóstico
Excludente	Sua presença diminui a probabilidade de confirmação do diagnóstico
Desnecessário	Não é necessário para a confirmação do diagnóstico

Fonte: FLORES (2005, p. 63)

Caso o agente de DOMÍNIO indique uma avaliação qualitativa satisfatória, é realizada a avaliação quantitativa que submete a rede do aluno a uma base de casos reais para avaliar sua performance. Neste caso, é avaliada, além da estrutura da rede em si, a distribuição de probabilidades entre as variáveis, ou seja, é verificado se a rede do aluno legitimamente pode fornecer como resposta o diagnóstico esperado pelo especialista.

De modo geral, nas avaliações é levado em consideração se o modelo formulado pelo aluno realmente é uma rede bayesiana, se não faltam ou sobram informações/etapas na rede e qual o resultado do diagnóstico diante de uma base de dados de casos reais. As redes dos alunos podem ser classificadas conforme a tabela 2.

Tabela 2 – Possíveis classificações da rede bayesiana do aluno

Rede	**Parâmetros**
Inviável	Rede apresenta ciclos ou nós não orientados
Incorreta	Sem diagnóstico, diagnóstico justifica as causas, presença de nós excludentes
Potencial	Ausência de alguns nós importantes e presença de nós desnecessários
Satisfatória	Diferente do modelo especialista, mas satisfaz o estudo do caso proposto
Completa	Rede topologicamente idêntica ao modelo construído pelo especialista

Fonte: FLORES (2005, p. 71)

Já o agente MEDIADOR detém as estratégias e táticas pedagógicas necessárias caso o aluno precise de apoio durante o processo de aprendizagem. Cabe a ele também decidir quando e de que forma utilizar estas táticas. Para decidir qual ação pedagógica tomar, leva-se em consideração a avaliação da rede bayesiana do aluno, a autoconfiança declarada pelo aluno e a credibilidade inferida pelo agente APRENDIZ e, ainda, a incidência de problemas encontrados na modelagem da rede.

As táticas pedagógicas utilizadas pelo agente MEDIADOR podem ser: correção; indicação; sugestão; experimentação; busca; reflexão; exemplos; problematização; discussão; demonstração e hipóteses. Estas táticas são distribuídas entre as estratégias que seguem:

- **Orientação**: abordagem direta, utilizada quando são identificados problemas graves na rede do aluno, objetivando indicar estes erros para correção e aumentar a confiança do aluno no agente de DOMÍNIO.
- **Contestação**: indica uma reavaliação da rede criada pelo aluno, indicando a existência de inconsistências. Nesta estratégia é levada em consideração a autoconfiança declarada pelo aluno e a credibilidade aferida pelo agente APRENDIZ, para avaliar qual o tipo de abordagem que será utilizada.

- **Apoio**: é utilizado nos casos em que a rede do aluno está categorizada como potencial e sua autoconfiança for baixa. Os casos clínicos são reapresentados apontando o que já está correto, para fazer com que o aluno pense sobre o que está faltando.
- **Ampliação**: utilizada com redes potenciais ou satisfatórias e autoconfiança dos alunos alta. No primeiro caso, sugere a elaboração de novas hipóteses. Já, no segundo caso, é apresentado ao aluno que a performance da sua rede pode melhorar, sugerindo revisões.
- **Comprovação**: utilizado para aumentar a confiança dos alunos que chegaram a um resultado satisfatório, mas que, mesmo assim, não se sentem seguros sobre o resultado atingido. Para tanto, pode ser apresentado o modelo do especialista e comparações da rede do aluno com o banco de dados de casos reais.

A tabela 3 apresenta uma relação entre os resultados da avaliação da rede do aluno, seu nível de autoconfiança e estratégias de ensino a serem utilizadas em cada situação.

Tabela 3 – Resumo das estratégias empregadas no AMPLIA

	Nível de autoconfiança do aluno		
Avaliação da rede	**Alta**	**Média**	**Baixa**
Inviável	Orientação	Orientação	Orientação
Incompleta	Contestação	Contestação	Contestação
Potencial	Ampliação	Ampliação ou apoio	Apoio
Satisfatória	Ampliação	Comprovação ou apoio	Comprovação
Completa	Comprovação	Comprovação	Comprovação

Fonte: FLORES (2005, p. 73)

A seguir são apresentadas algumas situações para ilustrar o funcionamento do AMPLIA em determinadas situações, quanto à tomada de decisão de qual estratégia utilizar (FLORES, 2005).

Situação 1: O modelo do aluno apresenta problemas sérios de estrutura, como evidências sem ligação com as demais, ou evidencias ligadas em ciclo.

Estratégia: Orientação sobre o problema existente na rede.

Exemplo: Há ciclos (nodos desconexos) em sua rede que não estão de acordo com o conceito de rede bayesiana. / Seu modelo não corresponde ao modelo de rede bayesiana. / Sua rede não está de acordo com o conceito de rede bayesiana.

Situação 2: O modelo do aluno contém praticamente todas as evidências, entretanto, as relações entre elas não estão corretas ou estão faltando relações.

Estratégia: Comprovação de que estão próximos da confirmação do diagnóstico (para alunos que declararam média ou baixa autoconfiança) e incentivo para ampliar o modelo (para alunos de elevada autoconfiança).

Exemplo: Comprovação: Experimente executar sua rede e veja se é possível identificar o diagnóstico, pois ainda estão faltando algumas relações entre os nodos ou estas reações estão incorretas. No material em anexo, há informações sobre algumas destas relações.

3 A interação no ambiente

O ambiente do projeto AMPLIA se desenvolveu em duas fases. Inicialmente os trabalhos foram focados na construção dos agentes e suas estruturas de comunicação. A concepção foi desenvolvida em Delphi 6.0, em função do seu precedente (o editor de redes bayesianas SEAMED) ter sido desenvolvido nesta plataforma. A fase inicial demandou um tempo grande de desenvolvimento em função da agentificação dos módulos do sistema.

As especificações iniciais da comunicação dos agentes foram baseadas na linguagem de comunicação FIPA-SL. Para a interface desta linguagem foi desenvolvida uma biblioteca chamada FÁCIL. Através desta biblioteca foi possível utilizar o padrão FIPA.

A segunda fase do projeto se deu com a integração do AMPLIA com agentes de controle de perfil de usuário (UPA) e pesquisa na internet baseada em consultas semânticas (SEARCH), ambos desenvolvidos em Java pelo grupo de inteligência artificial da UFRGS. A integração também tomou bastante tempo de implementação.

Atualmente, o AMPLIA roda em ambiente de rede local, não sendo possível o acesso ao sistema via Internet. Uma das propostas (que será mais bem detalhada nos próximos capítulos) é a adaptação do sistema para que o mesmo consiga fazer uso dos recursos da WEB através da substituição da biblioteca FACIL que não é portável para esta plataforma JAVA, pela API JADE.

No próximo capítulo será apresentado de forma delineada o banco de dados utilizados para o desenvolvimento de um protótipo a ser criado para substituir o Agente de DOMÍNIO original.

APÊNDICE B – ARQUIVO SQL PARA GERAR BANCO DE DADOS

```
 CREATE TABLE TipoNodo (
  idTipoNodo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  tipoNodo VARCHAR(25) NULL,
  PRIMARY KEY(idTipoNodo)
);

CREATE TABLE DiagnosticosPossiveis (
  idDiagnosticos INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  descricaoDiag VARCHAR(50) NULL,
  PRIMARY KEY(idDiagnosticos)
);

CREATE TABLE CasosReais (
  idCasosReais INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  nomeCasosReais INTEGER UNSIGNED NULL,
  PRIMARY KEY(idCasosReais)
);

CREATE TABLE CasoEstudo (
  idCasoEstudo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  nomeCaso VARCHAR(50) NOT NULL,
  descricaoCaso VARCHAR(200) NULL,
  arquivoCasoReal VARCHAR(150) NULL,
  PRIMARY KEY(idCasoEstudo)
);

CREATE TABLE Imagens (
  idImagens INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  CasoEstudo_idCasoEstudo INTEGER UNSIGNED NOT NULL,
  caminhoImagens VARCHAR(150) NULL,
  PRIMARY KEY(idImagens),
  INDEX Imagens_FKIndex1(CasoEstudo_idCasoEstudo),
  FOREIGN KEY(CasoEstudo_idCasoEstudo)
    REFERENCES CasoEstudo(idCasoEstudo)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE Nodos (
  idNodos INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  TipoNodo_idTipoNodo INTEGER UNSIGNED NOT NULL,
  descricaoNodo VARCHAR(50) NULL,
  nomeNodo VARCHAR(35) NULL,
  evidenDiagNodo VARCHAR(100) NULL,
  PRIMARY KEY(idNodos),
  INDEX Nodos_FKIndex1(TipoNodo_idTipoNodo),
  FOREIGN KEY(TipoNodo_idTipoNodo)
    REFERENCES TipoNodo(idTipoNodo)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE estado (
  idEstadoNodo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
```

```
  Nodos_idNodos INTEGER UNSIGNED NOT NULL,
  nomeEstado VARCHAR(15) NULL,
  valorEstado INTEGER UNSIGNED NULL,
  PRIMARY KEY(idEstadoNodo),
  INDEX estado_FKIndex1(Nodos_idNodos),
  FOREIGN KEY(Nodos_idNodos)
    REFERENCES Nodos(idNodos)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE ComentariosNodo (
  idComentariosNodo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Nodos_idNodos INTEGER UNSIGNED NOT NULL,
  textoComentario VARCHAR(400) NULL,
  PRIMARY KEY(idComentariosNodo),
  INDEX ComentariosNodo_FKIndex1(Nodos_idNodos),
  FOREIGN KEY(Nodos_idNodos)
    REFERENCES Nodos(idNodos)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE Arquivos (
  idArquivos INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  CasoEstudo_idCasoEstudo INTEGER UNSIGNED NOT NULL,
  descricaoArquivo VARCHAR(100) NULL,
  caminhoArquivo VARCHAR(150) NULL,
  PRIMARY KEY(idArquivos),
  INDEX Arquivos_FKIndex1(CasoEstudo_idCasoEstudo),
  FOREIGN KEY(CasoEstudo_idCasoEstudo)
    REFERENCES CasoEstudo(idCasoEstudo)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE PrioridadeNodo (
  idPropriedadeNodo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Nodos_idNodos INTEGER UNSIGNED NOT NULL,
  CasoEstudo_idCasoEstudo INTEGER UNSIGNED NOT NULL,
  prioridadeNodo VARCHAR(50) NULL,
  PRIMARY KEY(idPropriedadeNodo),
  INDEX PropriedadeNodo_FKIndex1(CasoEstudo_idCasoEstudo),
  INDEX PrioridadeNodo_FKIndex2(Nodos_idNodos),
  FOREIGN KEY(CasoEstudo_idCasoEstudo)
    REFERENCES CasoEstudo(idCasoEstudo)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(Nodos_idNodos)
    REFERENCES Nodos(idNodos)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE RelacionamentoNodos (
  idNodoPai INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  idNodoFilho INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(idNodoPai, idNodoFilho),
  INDEX RelacionamentoNodos_FKIndex1(idNodoPai),
  INDEX RelacionamentoNodos_FKIndex2(idNodoFilho),
  FOREIGN KEY(idNodoPai)
```

```
    REFERENCES Nodos(idNodos)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(idNodoFilho)
    REFERENCES Nodos(idNodos)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE SeqProbCondicional (
  seqProbabilidades INTEGER UNSIGNED NOT NULL,
  idEstadoNodo INTEGER UNSIGNED NOT NULL,
  Nodos_idNodos INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(seqProbabilidades, idEstadoNodo),
  INDEX SeqProbCondicional_FKIndex2(idEstadoNodo),
  INDEX SeqProbCondicional_FKIndex2(Nodos_idNodos),
  FOREIGN KEY(idEstadoNodo)
    REFERENCES estado(idEstadoNodo)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(Nodos_idNodos)
    REFERENCES Nodos(idNodos)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE DiagnosticosAssociadosNodo (
  idDiagnosticosAssociadosNodo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  DiagnosticosPossiveis_idDiagnosticos INTEGER UNSIGNED NOT NULL,
  Nodos_idNodos INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(idDiagnosticosAssociadosNodo),
  INDEX DiagnosticosAssociadosNodo_FKIndex1(Nodos_idNodos),
  INDEX
DiagnosticosAssociadosNodo_FKIndex2(DiagnosticosPossiveis_idDiagnosticos),
  FOREIGN KEY(Nodos_idNodos)
    REFERENCES Nodos(idNodos)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(DiagnosticosPossiveis_idDiagnosticos)
    REFERENCES DiagnosticosPossiveis(idDiagnosticos)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE ImagensNodo (
  idImagensNodo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Nodos_idNodos INTEGER UNSIGNED NOT NULL,
  Imagens_idImagens INTEGER UNSIGNED NOT NULL,
  nomeImagenNodo VARCHAR(35) NULL,
  PRIMARY KEY(idImagensNodo),
  INDEX ImagensNodo_FKIndex1(Imagens_idImagens),
  INDEX ImagensNodo_FKIndex2(Nodos_idNodos),
  FOREIGN KEY(Imagens_idImagens)
    REFERENCES Imagens(idImagens)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(Nodos_idNodos)
    REFERENCES Nodos(idNodos)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);
```

```
CREATE TABLE nodosBogus (
  idnodosBogus INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  CasoEstudo_idCasoEstudo INTEGER UNSIGNED NOT NULL,
  Nodos_idNodos INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(idnodosBogus),
  INDEX nodosBogus_FKIndex1(Nodos_idNodos),
  INDEX nodosBogus_FKIndex2(CasoEstudo_idCasoEstudo),
  FOREIGN KEY(Nodos_idNodos)
    REFERENCES Nodos(idNodos)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(CasoEstudo_idCasoEstudo)
    REFERENCES CasoEstudo(idCasoEstudo)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE DiagNecessarios (
  CasoEstudo_idCasoEstudo INTEGER UNSIGNED NOT NULL,
  DiagnosticosPossiveis_idDiagnosticos INTEGER UNSIGNED NOT NULL,
  INDEX DiagNecessarios_FKIndex1(DiagnosticosPossiveis_idDiagnosticos),
  INDEX DiagNecessarios_FKIndex2(CasoEstudo_idCasoEstudo),
  FOREIGN KEY(DiagnosticosPossiveis_idDiagnosticos)
    REFERENCES DiagnosticosPossiveis(idDiagnosticos)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(CasoEstudo_idCasoEstudo)
    REFERENCES CasoEstudo(idCasoEstudo)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE nodoCasoReal (
  estado_idEstadoNodo INTEGER UNSIGNED NOT NULL,
  CasosReais_idCasosReais INTEGER UNSIGNED NOT NULL,
  Nodos_idNodos INTEGER UNSIGNED NOT NULL,
  INDEX nodoCasoReal_FKIndex1(Nodos_idNodos),
  INDEX nodoCasoReal_FKIndex2(CasosReais_idCasosReais),
  INDEX nodoCasoReal_FKIndex3(estado_idEstadoNodo),
  FOREIGN KEY(Nodos_idNodos)
    REFERENCES Nodos(idNodos)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(CasosReais_idCasosReais)
    REFERENCES CasosReais(idCasosReais)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(estado_idEstadoNodo)
    REFERENCES estado(idEstadoNodo)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE ValorSeqProbNodo (
  idValorSeqProbNodo INTEGER UNSIGNED NOT NULL,
  SeqProbCondicional_idEstadoNodo INTEGER UNSIGNED NOT NULL,
  SeqProbCondicional_seqProbabilidades INTEGER UNSIGNED NOT NULL,
  valorEstadoNodo INTEGER UNSIGNED NULL,
  PRIMARY KEY(idValorSeqProbNodo),
```

```
  INDEX ValorSeqProbNodo_FKIndex2(SeqProbCondicional_seqProbabilidades,
SeqProbCondicional_idEstadoNodo),
  FOREIGN KEY(SeqProbCondicional_seqProbabilidades,
SeqProbCondicional_idEstadoNodo)
    REFERENCES SeqProbCondicional(seqProbabilidades, idEstadoNodo)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);
```

APÊNDICE C – ARQUIVO COM REDE EXPORTADA


```
<?xml version="1.0" ?>
- <XML_NET_FILE>
<CASO_ESTUDO NOME="Futebol" />
- <REDE>
- <ARVORE_SEMANTICA TAMANHO="1">
<NODO PAI="-1" NOME="" />
</ARVORE_SEMANTICA>
- <NODOS TAMANHO="7">
- <NODO NOME="Vontade">
<TITULO Valor="Vontade" />
- <DESCRICAO>
- <![CDATA[
Tenho vontade de jogar bola?
]]>
</DESCRICAO>
<X Valor="69" />
<Y Valor="137" />
<DIAGNOSTICO Valor="False" />
<EVIDENCIA Valor="True" />
<EXCLUDENTE Valor="False" />
<TIPO_DE_DADO Valor="dtSimbolic" />
<SEMANTICA Valor="0" />
<INDICE Valor="0" />
<ORDEM Valor="0" />
<REDE_ORIGINAL Valor="-1" />
<CLASSIFICACAO Valor="Essencial" />
- <COMENTARIO>
- <![CDATA[
Desejo, anseio , disposição de espirito para participar de uma partida de futebol.
]]>
</COMENTARIO>
- <PAIS TAMANHO="1">
<NODO>Tempo</NODO>
</PAIS>
- <FUNCAO TIPO="0" UNIVERSORESTRITO="True">
<VALOR NOME="LIMMIN">0.00000</VALOR>
<VALOR NOME="PONTMIN">0.00000</VALOR>
<VALOR NOME="CENTRO">50.00000</VALOR>
<VALOR NOME="PONTMAX">100.00000</VALOR>
<VALOR NOME="LIMMAX">100.00000</VALOR>
<FUNCAO_PADRAO>50.00000</FUNCAO_PADRAO>
</FUNCAO>
- <ESTADOS TAMANHO="2">
<NOME PRIORIDADE="0">Sim</NOME>
<NOME PRIORIDADE="0">Não</NOME>
</ESTADOS>
<MIDIAS TAMANHO="0" />
<DIAGNOSTICO_ASSOCIADO COUNT="0" />
- <TABELA TAMANHO="4">
<VALOR>0.30000</VALOR>
<VALOR>0.99000</VALOR>
<VALOR>0.70000</VALOR>
<VALOR>0.01000</VALOR>
</TABELA>
<TABELA_UTIL TAMANHO="0" />
</NODO>
```

```
- <NODO NOME="Tempo">
  <TITULO Valor="Tempo" />
   - <DESCRICAO>
   - <![CDATA[
   Como está o tempo?
     ]]>
     </DESCRICAO>
  <X Valor="68" />
  <Y Valor="252" />
  <DIAGNOSTICO Valor="False" />
  <EVIDENCIA Valor="True" />
  <EXCLUDENTE Valor="False" />
  <TIPO_DE_DADO Valor="dtSimbolic" />
  <SEMANTICA Valor="0" />
  <INDICE Valor="1" />
  <ORDEM Valor="0" />
  <REDE_ORIGINAL Valor="-1" />
  <CLASSIFICACAO Valor="Complementar" />
   - <COMENTARIO>
   - <![CDATA[
   O clima compreende-se por um conjunto de estados como tempo, vento e umidade. Cada região do
      planeta tem um determinado clima, e isso influencia na maneira das pessoas viverem e nas
      atividades de lazer da região.
     ]]>
     </COMENTARIO>
  <PAIS TAMANHO="0" />
- <FUNCAO TIPO="0" UNIVERSORESTRITO="True">
  <VALOR NOME="LIMMIN">0.00000</VALOR>
  <VALOR NOME="PONTMIN">0.00000</VALOR>
  <VALOR NOME="CENTRO">50.00000</VALOR>
  <VALOR NOME="PONTMAX">100.00000</VALOR>
  <VALOR NOME="LIMMAX">100.00000</VALOR>
  <FUNCAO_PADRAO>50.00000</FUNCAO_PADRAO>
     </FUNCAO>
- <ESTADOS TAMANHO="2">
  <NOME PRIORIDADE="0">Chuvoso</NOME>
  <NOME PRIORIDADE="0">Ensolarado</NOME>
     </ESTADOS>
  <MIDIAS TAMANHO="0" />
  <DIAGNOSTICO_ASSOCIADO COUNT="0" />
- <TABELA TAMANHO="2">
  <VALOR>0.50000</VALOR>
  <VALOR>0.50000</VALOR>
     </TABELA>
  <TABELA_UTIL TAMANHO="0" />
     </NODO>
- <NODO NOME="Local">
  <TITULO Valor="Local" />
   - <DESCRICAO>
   - <![CDATA[
   Tipo de Local
     ]]>
     </DESCRICAO>
  <X Valor="184" />
  <Y Valor="252" />
  <DIAGNOSTICO Valor="False" />
  <EVIDENCIA Valor="True" />
  <EXCLUDENTE Valor="False" />
  <TIPO_DE_DADO Valor="dtSimbolic" />
  <SEMANTICA Valor="0" />
  <INDICE Valor="2" />
  <ORDEM Valor="0" />
  <REDE_ORIGINAL Valor="-1" />
  <CLASSIFICACAO Valor="Essencial" />
   - <COMENTARIO>
   - <![CDATA[
   Quadras de parque , campos, parques ou qualquer lugar aberto com um bom espaço podem servir como
      lugar para se realizar uma partida de futebol.
```

```
    ]]>
    </COMENTARIO>
- <PAIS TAMANHO="2">
  <NODO>Tempo</NODO>
  <NODO>Dinheiro</NODO>
    </PAIS>
- <FUNCAO TIPO="0" UNIVERSORESTRITO="True">
  <VALOR NOME="LIMMIN">0.00000</VALOR>
  <VALOR NOME="PONTMIN">0.00000</VALOR>
  <VALOR NOME="CENTRO">50.00000</VALOR>
  <VALOR NOME="PONTMAX">100.00000</VALOR>
  <VALOR NOME="LIMMAX">100.00000</VALOR>
  <FUNCAO_PADRAO>50.00000</FUNCAO_PADRAO>
    </FUNCAO>
- <ESTADOS TAMANHO="2">
  <NOME PRIORIDADE="0">Coberto</NOME>
  <NOME PRIORIDADE="0">Ar Livre</NOME>
    </ESTADOS>
  <MIDIAS TAMANHO="0" />
  <DIAGNOSTICO_ASSOCIADO COUNT="0" />
- <TABELA TAMANHO="8">
  <VALOR>0.99000</VALOR>
  <VALOR>0.30000</VALOR>
  <VALOR>0.80000</VALOR>
  <VALOR>0.01000</VALOR>
  <VALOR>0.01000</VALOR>
  <VALOR>0.70000</VALOR>
  <VALOR>0.20000</VALOR>
  <VALOR>0.99000</VALOR>
    </TABELA>
  <TABELA_UTIL TAMANHO="0" />
    </NODO>
- <NODO NOME="Dinheiro">
  <TITULO Valor="Dinheiro" />
   - <DESCRICAO>
   - <![CDATA[
   Tenho dinheiro?
     ]]>
     </DESCRICAO>
  <X Valor="309" />
  <Y Valor="252" />
  <DIAGNOSTICO Valor="False" />
  <EVIDENCIA Valor="True" />
  <EXCLUDENTE Valor="False" />
  <TIPO_DE_DADO Valor="dtSimbolic" />
  <SEMANTICA Valor="0" />
  <INDICE Valor="3" />
  <ORDEM Valor="0" />
  <REDE_ORIGINAL Valor="-1" />
  <CLASSIFICACAO Valor="Complementar" />
   - <COMENTARIO>
   - <![CDATA[
   O Dinheiro é um artefato usado na troca de bens, podendo ter presença "física", como no caso de
      moedas ou papel moeda, ou virtual, como no caso do dinheiro depositado em bancos e aplicado na
      bolsa.
     ]]>
     </COMENTARIO>
  <PAIS TAMANHO="0" />
- <FUNCAO TIPO="0" UNIVERSORESTRITO="True">
  <VALOR NOME="LIMMIN">0.00000</VALOR>
  <VALOR NOME="PONTMIN">0.00000</VALOR>
  <VALOR NOME="CENTRO">50.00000</VALOR>
  <VALOR NOME="PONTMAX">100.00000</VALOR>
  <VALOR NOME="LIMMAX">100.00000</VALOR>
  <FUNCAO_PADRAO>50.00000</FUNCAO_PADRAO>
    </FUNCAO>
- <ESTADOS TAMANHO="2">
```

```
<NOME PRIORIDADE="0">Sim</NOME>
<NOME PRIORIDADE="0">Não</NOME>
</ESTADOS>
<MIDIAS TAMANHO="0" />
<DIAGNOSTICO_ASSOCIADO COUNT="0" />
- <TABELA TAMANHO="2">
<VALOR>0.50000</VALOR>
<VALOR>0.50000</VALOR>
</TABELA>
<TABELA_UTIL TAMANHO="0" />
</NODO>
- <NODO NOME="Bola">
<TITULO Valor="Bola" />
- <DESCRICAO>
- <![CDATA[
Tenho bola?
]]>
</DESCRICAO>
<X Valor="309" />
<Y Valor="138" />
<DIAGNOSTICO Valor="False" />
<EVIDENCIA Valor="True" />
<EXCLUDENTE Valor="False" />
<TIPO_DE_DADO Valor="dtSimbolic" />
<SEMANTICA Valor="0" />
<INDICE Valor="4" />
<ORDEM Valor="0" />
<REDE_ORIGINAL Valor="-1" />
<CLASSIFICACAO Valor="Trigger" />
- <COMENTARIO>
- <![CDATA[
Objeto esferico cuja finalidade principal no jogo de futebol é coloca-la para dentro do goleira
do adversario, marcando assim um gol (designação do ponto no futebol).
]]>
</COMENTARIO>
- <PAIS TAMANHO="1">
<NODO>Dinheiro</NODO>
</PAIS>
- <FUNCAO TIPO="0" UNIVERSORESTRITO="True">
<VALOR NOME="LIMMIN">0.00000</VALOR>
<VALOR NOME="PONTMIN">0.00000</VALOR>
<VALOR NOME="CENTRO">50.00000</VALOR>
<VALOR NOME="PONTMAX">100.00000</VALOR>
<VALOR NOME="LIMMAX">100.00000</VALOR>
<FUNCAO_PADRAO>50.00000</FUNCAO_PADRAO>
</FUNCAO>
- <ESTADOS TAMANHO="2">
<NOME PRIORIDADE="0">Sim</NOME>
<NOME PRIORIDADE="0">Não</NOME>
</ESTADOS>
<MIDIAS TAMANHO="0" />
<DIAGNOSTICO_ASSOCIADO COUNT="0" />
- <TABELA TAMANHO="4">
<VALOR>0.99000</VALOR>
<VALOR>0.30000</VALOR>
<VALOR>0.01000</VALOR>
<VALOR>0.70000</VALOR>
</TABELA>
<TABELA_UTIL TAMANHO="0" />
</NODO>
- <NODO NOME="Futebol">
<TITULO Valor="Futebol" />
- <DESCRICAO>
- <![CDATA[
Vou jogar bola?
]]>
</DESCRICAO>
```

```
<X Valor="184" />
<Y Valor="138" />
<DIAGNOSTICO Valor="True" />
<EVIDENCIA Valor="False" />
<EXCLUDENTE Valor="False" />
<TIPO_DE_DADO Valor="dtSimbolic" />
<SEMANTICA Valor="0" />
<INDICE Valor="5" />
<ORDEM Valor="0" />
<REDE_ORIGINAL Valor="-1" />
<CLASSIFICACAO Valor="Complementar" />
- <COMENTARIO>
- <![CDATA[
O futebol é o esporte mais praticado do mundo. É disputado por dois times, ou equipas, de 11
   jogadores que têm por objetivo colocar a bola na meta, ou baliza, adversária. O gol (meta ou
   baliza) é um retângulo formado por duas traves ou postes verticais, perpendiculares ao solo, e
   uma trave ou travessão paralela ao solo. Ali fica postado o goleiro, ou guarda-redes, que é o
   único jogador com permissão para colocar as mãos na bola (apenas dentro da sua área),
   defendendo o gol. O jogo é vencido pelo time que marcar um maior número de gols.
  ]]>
  </COMENTARIO>
- <PAIS TAMANHO="4">
<NODO>Local</NODO>
<NODO>Vontade</NODO>
<NODO>Bola</NODO>
<NODO>Jogadores</NODO>
  </PAIS>
- <FUNCAO TIPO="0" UNIVERSORESTRITO="True">
<VALOR NOME="LIMMIN">0.00000</VALOR>
<VALOR NOME="PONTMIN">0.00000</VALOR>
<VALOR NOME="CENTRO">50.00000</VALOR>
<VALOR NOME="PONTMAX">100.00000</VALOR>
<VALOR NOME="LIMMAX">100.00000</VALOR>
<FUNCAO_PADRAO>50.00000</FUNCAO_PADRAO>
  </FUNCAO>
- <ESTADOS TAMANHO="2">
<NOME PRIORIDADE="0">Sim</NOME>
<NOME PRIORIDADE="0">Não</NOME>
  </ESTADOS>
<MIDIAS TAMANHO="0" />
<DIAGNOSTICO_ASSOCIADO COUNT="0" />
- <TABELA TAMANHO="32">
<VALOR>0.99000</VALOR>
<VALOR>0.40000</VALOR>
<VALOR>0.30000</VALOR>
<VALOR>0.10000</VALOR>
<VALOR>0.80000</VALOR>
<VALOR>0.30000</VALOR>
<VALOR>0.30000</VALOR>
<VALOR>0.10000</VALOR>
<VALOR>0.90000</VALOR>
<VALOR>0.30000</VALOR>
<VALOR>0.20000</VALOR>
<VALOR>0.10000</VALOR>
<VALOR>0.70000</VALOR>
<VALOR>0.20000</VALOR>
<VALOR>0.20000</VALOR>
<VALOR>0.10000</VALOR>
<VALOR>0.01000</VALOR>
<VALOR>0.60000</VALOR>
<VALOR>0.70000</VALOR>
<VALOR>0.90000</VALOR>
<VALOR>0.20000</VALOR>
<VALOR>0.70000</VALOR>
<VALOR>0.70000</VALOR>
<VALOR>0.90000</VALOR>
<VALOR>0.10000</VALOR>
```

```
<VALOR>0.70000</VALOR>
<VALOR>0.80000</VALOR>
<VALOR>0.90000</VALOR>
<VALOR>0.30000</VALOR>
<VALOR>0.80000</VALOR>
<VALOR>0.80000</VALOR>
<VALOR>0.90000</VALOR>
  </TABELA>
<TABELA_UTIL TAMANHO="0" />
  </NODO>
- <NODO NOME="Jogadores">
<TITULO Valor="Jogadores" />
 - <DESCRICAO>
 - <![CDATA[
 Tenho outros jogadores?
  ]]>
  </DESCRICAO>
<X Valor="183" />
<Y Valor="41" />
<DIAGNOSTICO Valor="False" />
<EVIDENCIA Valor="True" />
<EXCLUDENTE Valor="False" />
<TIPO_DE_DADO Valor="dtSimbolic" />
<SEMANTICA Valor="0" />
<INDICE Valor="6" />
<ORDEM Valor="0" />
<REDE_ORIGINAL Valor="-1" />
<CLASSIFICACAO Valor="Trigger" />
 - <COMENTARIO>
 - <![CDATA[
 Jogadores são um grupo de pessoas que praticam um esporte por habito ou profissão.
  ]]>
  </COMENTARIO>
<PAIS TAMANHO="0" />
- <FUNCAO TIPO="0" UNIVERSORESTRITO="True">
<VALOR NOME="LIMMIN">0.00000</VALOR>
<VALOR NOME="PONTMIN">0.00000</VALOR>
<VALOR NOME="CENTRO">50.00000</VALOR>
<VALOR NOME="PONTMAX">100.00000</VALOR>
<VALOR NOME="LIMMAX">100.00000</VALOR>
<FUNCAO_PADRAO>50.00000</FUNCAO_PADRAO>
  </FUNCAO>
- <ESTADOS TAMANHO="2">
<NOME PRIORIDADE="0">Sim</NOME>
<NOME PRIORIDADE="0">Não</NOME>
  </ESTADOS>
<MIDIAS TAMANHO="0" />
<DIAGNOSTICO_ASSOCIADO COUNT="0" />
- <TABELA TAMANHO="2">
<VALOR>0.50000</VALOR>
<VALOR>0.50000</VALOR>
  </TABELA>
<TABELA_UTIL TAMANHO="0" />
  </NODO>
  </NODOS>
  </REDE>
  </XML_NET_FILE>
```

APÊNDICE D – EXEMPLO SCRIPT SQL

```
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

SET AUTOCOMMIT=0;
START TRANSACTION;

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

-- Banco de Dados: `ampliajava`

-- INSERT TABELA`arquivoscasoestudo`
--
INSERT INTO `arquivoscasoestudo` (`idArquivos`, `casoEstudo_idCasoEstudo`, `descricaoArquivo`, `caminhoArquivo`) VALUES
(1, 1, 'C:\\Arquivo\\futeebol.txt', '1');

-- INSERT TABELA`cabseqprob`

INSERT INTO `cabseqprob` (`idcabSeqProb`, `nodosCaso_idNodosCaso`, `seqProb`) VALUES
(1, 1, 1),
(2, 5, 1),
(5, 7, 1),
(6, 4, 1),
(7, 4, 2),
(8, 3, 1),
(9, 3, 2),
(10, 6, 1),
(11, 6, 2),
(12, 6, 3),
(13, 6, 4),
(14, 2, 1),
(15, 2, 2),
(16, 2, 3),
(17, 2, 4),
(18, 2, 5),
(19, 2, 6),
(20, 2, 7),
(21, 2, 8),
(22, 2, 9),
(23, 2, 10),
(24, 2, 11),
(25, 2, 12),
(26, 2, 13),
```

```
(27, 2, 14),
(28, 2, 15),
(29, 2, 16);

-- INSERT TABELA`casoestudo`

INSERT INTO `casoestudo` (`idCasoEstudo`, `nomeCaso`, `descricaoCaso`, `arquivoCasoReal`) VALUES
(1, 'Futebol', 'Vamos jogar futebol', 'c:\\teste');

-- INSERT TABELA`casosreais`

INSERT INTO `casosreais` (`idCasosReais`, `nomeCasosReais`) VALUES
(1, 'Caso1');

-- INSERT TABELA`diagassnodo`

-- INSERT TABELA`estado`

INSERT INTO `estado` (`idEstadoNodo`, `nodosCaso_idNodosCaso`, `nomeEstado`, `valorEstado`) VALUES
(1, 1, 'SIM', 0),
(2, 1, 'NÃO', 0),
(3, 2, 'SIM', 0),
(4, 2, 'NÃO', 0),
(5, 3, 'SIM', 0),
(6, 3, 'NÃO', 0),
(7, 4, 'NÃO', 0),
(8, 4, 'SIM', 0),
(9, 5, 'NÃO', 0),
(10, 5, 'SIM', 0),
(11, 6, 'COBERTO', 0),
(12, 6, 'AR LIVRE', 0),
(13, 7, 'CHUVOSO', 0),
(14, 7, 'ENSOLARADO', 0);

-- INSERT TABELA`nodocasoreal`

INSERT INTO `nodocasoreal` (`idNodoCasoReal`, `casosReais_idCasosReais`, `nodosCaso_idNodoCaso`,
`estado_idEstadoNodo`) VALUES
(1, 1, 4, 7),
(2, 1, 5, 9),
(3, 1, 2, 0),
(4, 1, 1, 2),
(5, 1, 6, 12),
(6, 1, 7, 14),
(7, 1, 3, 5);

-- INSERT TABELA`nodos`
```

```
INSERT INTO `nodos` (`idNodos`, `nomeNodos`, `descricaoNodos`) VALUES
(1, 'Jogadores', 'Tenho Outros Jogadores'),
(2, 'Vontade', 'Tenho vontade de jogar bola?'),
(3, 'Futebol', 'Vou jogar bola?'),
(4, 'Bola', 'Tenho bola?'),
(5, 'Dinheiro', 'Tenho dinheiro?'),
(6, 'Local', 'Tipo Local'),
(7, 'Tempo', 'Como está o tempo?'),
(8, 'Visita a Asia', 'O paciente visitou a Asia nos ultimos meses'),
(9, 'Dispineia', 'O paciente apresenta dificuldades para respirar'),
(10, 'Tuberculose', 'O paciente tem tuberculose'),
(11, 'Cancer de Pulmão', 'O paciente apresenta cancer no pulmão'),
(12, 'Bronquite', 'O paciente tem bronquite '),
(13, 'Raio-X Positivo', 'Raio-X Positivo'),
(14, 'Fumante', 'O paciente fuma ');

-- INSERT TABELA`nodosbogus`

INSERT INTO `nodosbogus` (`idnodosBogus`, `casoEstudo_idCasoEstudo`, `nomeNodosBogus`, `informacoesAdicionais`, `proprEvidDiagNodosBogos`, `tipoNodo`) VALUES
(1, 1, 'Goleira', 'Estrutura que sustenta a rede localizada no fundo da linha de gol. ', 'E', 'S'),
(2, 1, 'Juiz', 'Árbitro. Aquele que tem o poder de julgar ', 'E', 'S'),
(3, 1, 'Natação', 'Ato ou exercício de nadar. Sistema de locomoção na água. ', 'E', 'S'),
(4, 1, 'Torcida', 'Grupo de pessoas q assiste a uma apresentação ou espetáculo e que manifesta incentivo ou repúdio. Grupo de torcedores.', 'E', 'S');

-- INSERT TABELA`nodoscaso`

INSERT INTO `nodoscaso` (`idNodosCaso`, `casoEstudo_idCasoEstudo`, `nodos_idNodos`, `proprEvidDiagNodosCaso`, `tipoNodo`, `comentariosNodosCaso`, `prioridadeNodosCaso`, `posNecesNodoCaso`, `descricaoDiagCaso`) VALUES
(1, 1, 1, 'E', 'S', 'Jogadores são um grupo de pessoas que praticam um esporte por habito ou profissão.', 'T', NULL, 'descricao Disgnostico'),
(2, 1, 3, 'D', 'S', 'O futebol é o esporte mais praticado do mundo. É disputado por dois times, ou equipas, de 11 jogadores que têm por objetivo colocar a bola na meta, ou baliza, adversária. O gol (meta ou baliza) é um retângulo formado por duas traves ou postes verticais, perpendiculares ao solo, e uma trave ou travessão paralela ao solo. Ali fica postado o goleiro, ou guarda-redes, que é o único jogador com permissão para colocar as mãos na bola (apenas dentro da sua área), defendendo o gol. O jogo é vencido pelo time que marcar um maior número de gols. ', '', 'N', 'descricaoDisgnostico'),
(3, 1, 2, 'E', 'S', 'Desejo, anseio , disposição de espirito para participar de uma partida de futebol.', 'E', NULL, 'Descriçao'),
(4, 1, 4, 'E', 'S', 'Objeto esferico cuja finalidade principal no jogo de futebol é coloca-la para dentro do goleira do adversario, marcando assim um gol (designação do ponto no futebol). ', 'T', NULL, 'Descriçao'),
(5, 1, 5, 'E', 'S', 'O Dinheiro é um artefato usado na troca de bens, podendo ter presença "física", como no caso de moedas ou papel moeda, ou virtual, como no caso do dinheiro depositado em bancos e aplicado na bolsa. ', 'C', NULL, 'Descriçao'),
(6, 1, 6, 'E', 'S', 'Quadras de parque , campos, parques ou qualquer lugar aberto com um bom espaço podem servir como lugar para se realizar uma partida de futebol. ', 'E', NULL, 'Descriçao'),
```

```
(7, 1, 7, 'E', 'S', 'O clima compreende-se por um conjunto de estados como tempo, vento e umidade. Cada região do planeta tem um determinado clima, e isso influencia na maneira das pessoas viverem e nas atividades de lazer da região.', 'C', NULL, 'Descriçao'),
(8, 2, 8, 'E', 'S', 'Comentario nodo Caso', 'C', '', 'Descricao diagnostico Caso'),
(9, 2, 9, 'E', 'S', 'Comentario nodo Caso', 'E', NULL, 'Descricao diagnostico Caso'),
(10, 2, 10, 'D', 'S', 'Comentario nodo Caso', '', 'N', 'Descricao diagnostico Caso'),
(11, 2, 11, 'D', 'S', 'Comentario nodo Caso', '', 'N', 'Descricao diagnostico Caso'),
(12, 2, 12, 'D', 'S', 'Comentario nodo Caso', '', 'N', 'Descricao diagnostico Caso'),
(13, 2, 13, 'E', 'S', 'Comentario nodo Caso', 'T', NULL, 'Descricao diagnostico Caso'),
(14, 2, 14, 'E', 'S', 'Comentario nodo Caso', 'E', NULL, 'Descricao diagnostico Caso');

-- INSERT TABELA`relacionamentonodospais`

INSERT INTO `relacionamentonodospais` (`nodosCaso_idNodosCasosFilho`, `nodosCaso_idNodosCasosPai`) VALUES
(2, 1),
(2, 3),
(2, 4),
(2, 6),
(3, 7),
(4, 5),
(6, 5),
(6, 7);

-- INSERT TABELA`seqprob`

INSERT INTO `seqprob` (`cabSeqProb_idcabSeqProb`, `estado_idEstadoNodoPai`, `nodosCaso_idNodosCasoPai`) VALUES
(6, 10, 5),
(7, 9, 5),
(8, 13, 7),
(9, 14, 7),
(10, 9, 5),
(10, 10, 5),
(10, 13, 7),
(11, 13, 7),
(12, 10, 5),
(12, 14, 7),
(13, 9, 5),
(13, 14, 7),
(14, 1, 1),
(14, 5, 3),
(14, 8, 4),
(14, 11, 6),
(15, 2, 1),
(15, 5, 3),
(15, 8, 4),
(15, 11, 6),
```

(16, 1, 1),
(16, 5, 3),
(16, 7, 4),
(16, 11, 6),
(17, 2, 1),
(17, 5, 3),
(17, 7, 4),
(17, 11, 6),
(18, 1, 1),
(18, 6, 3),
(18, 8, 4),
(18, 11, 6),
(19, 2, 1),
(19, 6, 3),
(19, 8, 4),
(19, 11, 6),
(20, 1, 1),
(20, 6, 3),
(20, 7, 4),
(20, 11, 6),
(21, 2, 1),
(21, 6, 3),
(21, 7, 4),
(21, 11, 6),
(22, 1, 1),
(22, 5, 3),
(22, 8, 4),
(22, 12, 6),
(23, 2, 1),
(23, 5, 3),
(23, 8, 4),
(23, 12, 6),
(24, 1, 1),
(24, 5, 3),
(24, 7, 4),
(24, 12, 6),
(25, 2, 1),
(25, 5, 3),
(25, 7, 4),
(25, 12, 6),
(26, 1, 1),
(26, 6, 3),
(26, 8, 4),
(26, 12, 6),
(27, 2, 1),
(27, 6, 3),
(27, 8, 4),
(27, 12, 6),

```
(28, 1, 1),
(28, 6, 3),
(28, 7, 4),
(28, 12, 6),
(29, 2, 1),
(29, 6, 3),
(29, 7, 4),
(29, 12, 6);

-- INSERT TABELA`valorprobnodo`

INSERT INTO `valorprobnodo` (`estado_idEstadoNodo`, `cabSeqProb_idcabSeqProb`, `valorProbNodo`)
VALUES
(1, 1, 0.5),
(2, 1, 0.5),
(3, 14, 0.99),
(3, 15, 0.4),
(3, 16, 0.3),
(3, 17, 0.1),
(3, 18, 0.8),
(3, 19, 0.3),
(3, 20, 0.3),
(3, 21, 0.1),
(3, 22, 0.9),
(3, 23, 0.3),
(3, 24, 0.2),
(3, 25, 0.1),
(3, 26, 0.7),
(3, 27, 0.2),
(3, 28, 0.2),
(3, 29, 0.1),
(4, 14, 0.1),
(4, 15, 0.6),
(4, 16, 0.7),
(4, 17, 0.9),
(4, 18, 0.2),
(4, 19, 0.7),
(4, 20, 0.7),
(4, 21, 0.9),
(4, 22, 0.1),
(4, 23, 0.7),
(4, 24, 0.8),
(4, 25, 0.9),
(4, 26, 0.3),
(4, 27, 0.8),
(4, 28, 0.8),
(4, 29, 0.9),
(5, 8, 0.3),
```

```
(5, 9, 0.99),
(6, 8, 0.7),
(6, 9, 0.01),
(7, 6, 0.01),
(7, 7, 0.7),
(8, 6, 0.99),
(8, 7, 0.3),
(9, 2, 0.5),
(10, 2, 0.5),
(11, 10, 0.99),
(11, 11, 0.3),
(11, 12, 0.8),
(11, 13, 0.01),
(12, 10, 0.01),
(12, 11, 0.7),
(12, 12, 0.2),
(12, 13, 0.99),
(13, 5, 0.5),
(14, 5, 0.5);
COMMIT;
```