UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

VANESSA FELISBERTO BILÉSIMO

MÉTODO PARA INTEGRAÇÃO DE UMA BASE DE CONHECIMENTOS DE UM SISTEMA ESPECIALISTA PROBABILÍSTICO UTILIZANDO A NETICA JAVA API

CRICIUMA, DEZEMBRO DE 2007

VANESSA FELISBERTO BILÉSIMO

MÉTODO PARA INTEGRAÇÃO DE UMA BASE DE CONHECIMENTOS DE UM SISTEMA ESPECIALISTA PROBABILÍSTICO UTILIZANDO A NETICA JAVA API

Trabalho de Conclusão de Curso apresentado para a obtenção do grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientadora: Profª MSc. Priscyla Waleska Targino de Azevedo Simões.

Co-orientador: Prof. Fabrício Giordani.

CRICIÚMA, DEZEMBRO DE 2007

VANESSA FELISBERTO BILÉSIMO

Método para a Integração de uma Base de Conhecimentos de um Sistema Especialista Probabilístico Utilizando a Netica Java API

Submetido ao corpo docente do Curso de Ciência da Computação na Unidade Acadêmica de Ciências, Engenharias e Tecnologias da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

_____

Profa. MSc. Ana Claudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

_____

Prof. MSc. Priscyla Waleska Targino de Azevedo Simões (UNESC)
Orientadora

_____

Prof. Fabrício Giordani (UNESC)
Co-orientador

_____

Prof. MSc. Merisandra Côrtes de Mattos (UNESC)

_____

Prof. MSc. Eduardo Menna da Silva (UNESC)

Dedico este trabalho aos meus pais Deonésio e Luzia e ao meu namorado John, por todo apoio e carinho durante esta caminhada.

AGRADECIMENTOS

Agradeço primeiramente a Deus por me dar o dom da vida, por me dar saúde, luz e sabedoria para chegar até aqui. Por ter me guiado sempre pelo melhor caminho a seguir iluminando-o com pessoas maravilhosas que sempre me deram apoio e coragem para alcançar mais um sonho.

Aos meus queridos pais, que são o porto seguro de minha vida, a razão do meu existir. Pelo apoio na hora certa e incentivo a continuar quando eu queria desistir nas horas de maior desespero ao final de cada semestre. Pela palavra amiga, amor e paciência. Obrigada pelo apoio incondicional.

Ao meu namorado John, que depois dos meus pais é meu maior incentivador, é meu grande companheiro de todas as horas, mesmo longe é meu refúgio nas horas de desespero e aflição. Obrigada por tudo, Te amo.

A minha irmã Tays pelo apoio concedido durante esta caminhada.

A minha Orientadora Priscyla, que com sua sabedoria e paciência soube me guiar indicando o próximo passo a seguir, ora fosse chamar minha atenção, ora fosse um elogio. Obrigada pela confiança e compreensão deste momento.

Ao meu Co-orientador Fabrício Giordani, pela sua disposição e dedicação de sempre me ajudar nas horas que precisei sanar dúvidas, as vezes até prejudicando seus próprios horários. Muito obrigada pela compreensão.

Ao colega de curso Edroaldo Lumertz da Rocha, pela sua disposição e sabedoria em me ajudar sempre que precisei sanar dúvidas de programação e contribuindo assim para a conclusão deste trabalho. Muito obrigada pela atenção.

Ao colega de curso Rafael Lazzari, pela sua disposição e sabedoria em me ajudar na hora que precisei de sua ajuda para concluir uma das etapa desta pesquisa.

A todos os professores e colaboradores do departamento, que cruzaram meu caminho contribuindo para que eu pudesse alcançar meus objetivos. Obrigada pela amizade e por compartilharem seus conhecimentos.

A todos os integrantes do Grupo de Informática médica e Telemedicina Kiron, que direta ou indiretamente ajudaram nos processos para a conclusão desta pesquisa.

A minhas amigas Rosangela e Patrícia por toda disposição de ajudar, de saber ouvir, de companheirismo e amizade ao longo desta jornada que chega ao fim.

Ao meu gerente administrativo Luiz Carlos Boeing que com toda compreensão soube entender e me apoiar nas horas que precisei de sua atenção, com todo carinho sempre me liberou em horário de trabalho. Sou muito grata a você.

A todos que estiveram a meu lado durante esta caminhada: amigos, colegas de trabalho, de faculdade, colegas no ônibus e todos os que me incentivaram de alguma forma para que hoje eu estivesse aqui. Gostaria de citar nomes, mas são muitos, e seria injusto esquecer alguém. Tenho um enorme carinho por todos vocês. Obrigada!

Nas grandes batalhas da vida, o primeiro passo para a vitória é o desejo de vencer.
Mahatma Gandhi

RESUMO

Em alguns domínios de aplicação, o raciocínio para a solução de um determinado problema não é exato, sendo acompanhado por certo grau de incerteza, como acontece em certos diagnósticos médicos e, para tratar desta questão, entre outras técnicas surgiram os Sistemas Especialistas Probabilísticos, cuja Base de Conhecimento é representada por meio de Redes Bayesianas. Neste contexto, o objetivo desta pesquisa é aplicar um método para integração de uma Base de Conhecimento de um Sistema Especialista Probabilístico utilizando a Netica Java API. A metodologia de desenvolvimento iniciou com o estudo das classes e métodos disponibilizados pela Netica Java API, seguindo-se com a escolha de algumas Redes Bayesianas para a utilização do estudo de caso, finalizando-se com a implementação e documentação do sistema desenvolvido. Esta pesquisa resultou em novas versões do Sistema Especialista para o Apoio ao Diagnóstico de Doenças Exantemáticas Maculopapulosas com Erupção Obrigatória, Sistema Especialista Probabiístico para Prognóstico de Doenças Bucais, Biowoman – Base de Conhecimento Dinâmica para Sistemas Especialistas Probabilístico desenvolvidos anteiormente na Universidade do Extremo Sul Catarinense. Estas aplicações foram implementadas no ambiente de programação NetBeans IDE 5.5 por meio da utilização da Netica Java API. Cada interface ofereceu recursos de inferência, módulo de explicação, de ajuda, e de análise gráfica das hipótese diagnósticas. Pode-se concluir por meio desta pesquisa que os resultados obtidos foram satisfatórios, considerando-se que o método de integração baseado na Netica Java API e as applets desenvolvidas atenderam as expectativas dos especialistas do domínio de aplicação.

Palavras chaves: Redes Bayesianas, *Shell* Netica, Netica Java API, Inteligência Artificial na saúde, Programação Orientada á Objetos.

ABSTRACT

In some areas of application, the reasoning for the solution of a specific problem is not exact, and is accompanied by some degree of uncertainty, as in some medical diagnoses, and to address this issue, among other techniques emerged the Systems Specialists Probabilistics, whose Knowledge Base is represented through networks Bayesians. In this context, the goal of this research is to implement a method to integrate a Knowledge Base System of a Specialist Probabilistic using the Netica Java API. The methodology of development began with the study of the classes and methods provided by Netica Java API, followed by the choice of some networks Bayesians for the use of case study, ending with the implementation and documentation of the system developed. This search resulted in new versions of System Support Specialist for the Diagnosis of Diseases Exantematicas Maculopapulosas with Compulsory Eruption, System Specialist Probabilistic for Prognosis of Oral Diseases, Biowoman-Dynamic Knowledge Base Systems for Experts Probabilistic developed previously at the University of the Far South Catarinense. These applications have been implemented in the programming environment NetBeans IDE 5.5 through the use of Netica Java API. Each interface offered resources of inference, module of explanation, aid, and graphical analysis of diagnostic hypothesis. It can be concluded through this research that the results were satisfactory, considering that the method of integration based on Netica Java API and applets developed met the expectations of experts of the field of application

Keywords: Bayesians Networks, Netica Shell, Netica Java API, health Artificial Intelligence, Objects Oriented Programming

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

AC	Aquisição de Conhecimento
API	Application Programmer Interfaces
BC	Base de Conhecimento
BIOWOMAN	Base de Conhecimento Dinâmica para Sistemas Especialistas Probabilísticos
IA	Inteligência Artificial
MI	Motor de Inferência
PC	Probabilidade Condicional
PROBUCAL	Sistema Especialista Probabilístico Para Prognóstico De Doenças Bucais
RB	Rede Bayesiana
SEDDEM	Sistema Especialista para o Apoio ao Diagnóstico de Doenças Exantemáticas Maculopapulosas com Erupção Obrigatória
SARA	Sistema Inteligente de Auxílio à Reparação na Injeção Eletrônica Automotiva
SEP	Sistema Especialista Probabilístico
UNESC	Universidade do Extremo Sul Catarinense
UTI	Unidade de Terapia Intensiva

SUMÁRIO

1 INTRODUÇÃO

Os Sistemas Especialistas Probabilísticos (SEP) hoje em dia são implementados e desenvolvidos com a finalidade de atuar em diversas áreas, desde a resolução de problemas até situações que envolvem maior complexidade como o diagnóstico de uma doença.

Nesses sistemas, o raciocínio probabilístico, fundamentado na teoria da probabilidade, é utilizado para modelagem de problemas que apresentam incerteza por aleatoriedade, ou seja, raciocínio cujo resultado mesmo em condições normais de experimentação, varia de uma observação para outra, dificultando desta maneira a previsão de um resultado futuro (LUNA, 2004). Destaca-se nesse contexto, o raciocínio médico, em que, por exemplo, podem existir dois pacientes com sinais e sintomas similares, porém com diagnósticos diferentes.

Considerando este aspecto, Nassar (2006) descreve que pela teoria da probabilidade não se pode prever com toda certeza o que acontecerá num novo caso, mesmo diante do conhecimento de casos anteriores do domínio de aplicação.

Sendo assim, a arquitetura de um sistema especialista probabilístico apresenta a Base de Conhecimento, a Aquisição de Conhecimento, Motor de Inferência, Especialista do Conhecimento, Engenheiro do Conhecimento, Usuário, a Interface com o Usuário (IU) e o módulo de explicação, sendo que a IU e o módulo de explicação costumam ser desenvolvidos em um ambiente de programação como o Delphi, C++ Builder, entre outros; já a Base de Conhecimento (BC) costuma ser implementada em ferramentas como a *shell* Netica (NORSYS, 2006), e o Hugin (HUGIN, 2006), entre outras.

A BC e o motor de inferência necessitam se comunicar com os demais módulos do SEP, para isso utiliza-se uma linguagem de programação escolhida para o desenvolvimento do sistema, chamadas a Application Programmer Interfaces (API) de uma s*hell* de Redes Bayesianas (RB) que, entre seus recursos, oferece o motor de inferência, que por meio do teorema de Bayes propaga e gera as probabilidades *a posteriori*, a partir das probabilidades *a priori* do domínio de conhecimento representado (LUNA, 2004).

A primeira API dessa *Shell* foi desenvolvida para ser utilizada na linguagem C, e, considerando o processo de modelagem de uma base de conhecimento por meio da Aquisição de Conhecimento (AC) não automatizada, oferecia recursos para gerenciamento: do ambiente Netica, de nós, de probabilidades condicionais, das relações de dependência entre os nós (*links*), de inferência, além de outras funcionalidades para aquisição de conhecimento automatizada.

Com o passar dos anos, os recursos dessa API foram atualizados, e seu fabricante desenvolveu uma nova biblioteca de funções, denominada Netica Java API, que apresenta as funções originais, e novos recursos que visam facilitar sua utilização, além de ser desenvolvida conforme a filosofia e metodologia do Java (NORSYS, 2006).

Nesse sentido, busca-se por meio dessa pesquisa, estudar e aplicar a metodologia de utilização da Netica Java API, no desenvolvimento de um sistema especialista probabilístico possibilitando uma contribuição na área de Inteligência Artificial (IA).

1.1 OBJETIVO GERAL

Utilizar a Netica Java API como interface de comunicação entre a base de conhecimento e a interface com o usuário no desenvolvimento de um protótipo de um Sistema Especialista Probabilístico.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho são:

a) compreender o processo de desenvolvimento de Sistemas Especialistas Probabilísticos;
b) compreender as funções disponibilizadas pela API da s*hell* Netica;
c) utilizar a Netica Java API em um SEP a ser desenvolvido em Java;
d) oferecer a documentação sobre a utilização da Netica Java API no desenvolvimento de um SEP;
e) oferecer atualização da base de conhecimento e dos demais módulos de um SEP desenvolvido na UNESC.

1.3 JUSTIFICATIVA

Em alguns domínios de aplicação, o raciocínio para a solução de um determinado problema não é exato, sendo acompanhado por um certo grau de incerteza, como acontece em certos diagnósticos médicos (NASSAR, 2006), e, para tratar desta questão, entre outras técnicas surgiram os modelos probabilisticos que caracterizam os sistemas especialistas probabilísticos, cuja BC é representada por meio de RB.

Segundo Luna (2004) existem motivos práticos para a utilização de redes bayesianas, pois essa forma de representação do conhecimento incerto permite analisar grandes quantidades de dados, para extrair conhecimentos úteis e, dessa forma, auxiliar os especialistas em seu processo de tomada de decisão em domínios de conhecimento como: saúde, indústria, computação, redes, entre outros.

Apesar da utilização de redes bayesianas na saúde, os SEP desenvolvidos na área médica costumam ser pouco utilizados no apoio à prática clínica, pois, conforme surgem novas informações sobre dado sinal, sintoma e prevalência de uma doença na população (informações que definem as probabilidades condicionais *a priori* da rede bayesiana), essas probabilidades condicionais precisam ser atualizadas, o que demanda tempo do engenheiro do conhecimento e do especialista do domínio de aplicação nesse processo continuado, e acaba tornando esses sistemas apenas resultados de pesquisa, desatualizados com o passar do tempo (MACHADO, 2006).

Nesse contexto, muitas aplicações que utilizam redes bayesianas para representação da base de conhecimento de SEP, utilizam a *shell* Netica e a Netica C API, pois essa s*hell* é muito aplicada nesses sistemas, por oferecer gratuitamente uma versão limitada, e por ser utilizada no apoio ao ensino de redes bayesianas na disciplinas de graduação e pós graduação do cursos de Ciência da Computação, além ser uma ferramenta didática, de fácil utilização e que permite boa interação com os especialistas do domínio de aplicação. (NASSAR, 2006).

Considerando a utilização da *shell* Netica, já foram desenvolvidos no Brasil, por meio dessa ferramenta, vários sistemas especialistas probabilísticos voltados à área da saúde, destacando-se a Universidade Federal de Santa Catarina (NASSAR, 2006), a Pontifícia Universidade Católica do Paraná (ÁVILA et al, 2005), e a Universidade do

Extremo Sul Catarinense - UNESC (ANTUNES,2002; GOULARTE, 2002; RODRIGUES, 2002; ROSA, 2002 ;TASCA, 2002).

Na UNESC, todos os SEP utilizam diferentes formas de chamada à mesma API (Netica C API), e foram projetados em ambientes como o C++ Builder e Delphi.

Para utilizar uma nova API da s*hell* Netica, não basta traduzir o código fonte inicial para uma nova linguagem, é necessário entender a filosofia e metodologia dessa nova API baseada no paradigma de programação orientada a objetos, além de explorar seus recursos na linguagem de programação escolhida para o desenvolvimento do sistema, como é caso da Netica C API, em que o engenheiro do conhecimento precisa desenvolver uma série de rotinas adicionais, necessárias para tratar algumas particularidades do motor de inferência (SIMÕES, 2001). Para isso, o programador tem que possuir bom conhecimento da linguagem de programação, da API que será utilizada, além de bom embasamento na área de IA, no que se refere ao processo de construção de SEP.

E, finalizando, como a Netica Java API foi desenvolvida para SEP a serem desenvolvidos em Java, Cozman (2001) destaca grande potencial da linguagem Java na construção de SEP por oferecer quatro vantagens: por apresentar portabilidade; segundo ser adotado por *browsers* na internet; possibilita trabalhar e utilizar ferramentas como a s*hell* Netica, que permitem modelar sistemas sobre o domínio da incerteza; e finalmente, quarta vantagem, por ser uma linguagem orientada à objetos.

1.4 ESTRUTURA DO TRABALHO

Este trabalho é composto por sete capítulos. No Capítulo 1 encontra-se uma contextualização ao tema proposto, bem como os objetivos e justificativa para realização deste trabalho.

No Capítulo 2 são abordados os conceitos fundamentais de Sistemas Especialistas Probabilísticos, sua arquitetura e sua maneira de tratar a incerteza por aleatoriedade.

O conceito de Redes Bayesianas e sua inferência (Teorema de Bayes) são assuntos abordados no Capítulo 3.

A *shell* Netica e as classes disponibilizadas na API para Java são descritas no Capítulo 4.

O Capítulo 5 contempla alguns trabalhos correlatos que envolvem aplicações de RB e a tecnologia Java.

O trabalho desenvolvido é apresentado no Capítulo 6, pela sua metodologia, incluindo os resultados obtidos.

E por fim, tem-se a conclusão, onde encontram-se também algumas sugestões para trabalhos futuros.

2 SISTEMAS ESPECIALISTAS PROBABILÍSTICOS

Os SEP são sistemas computacionais desenvolvidos para auxiliar na resolução de determinados problemas em diversas áreas do conhecimento que representam a incerteza por aleatoriedade. O conhecimento necessário para a solução dos problemas é adquirido por meio de um especialista do conhecimento (WILLIAMSON, 2005).

A presença da incerteza por aleatoriedade influencia em termo enfáticos[1] o modo como um especialista toma suas decisões. Em uma aplicação esta incerteza pode estar presente tanto nos dados de entrada como na solução do problema. Em um caso médico, por exemplo, podem existir dois pacientes com sinais e sintomas similares, porém, seus diagnósticos são diferentes. Com isso, o médico nesta situação para prescrever o diagnóstico de determinado paciente deve racionar por meio de incerteza, baseando-se em probabilidades já ocorridas anteriormente com muita freqüência sob o domínio da incerteza (MARQUES e MARIN, 2002)

O SEP deve ser capaz de simular e tomar decisões que seriam realizadas por especialistas, sendo que estas decisões devem se comportar de forma semelhante à decisão do especialista (TIBIRIÇA; NASSAR, 2003; SAHEKI, 2005).

Entre os personagens envolvidos na construção desse tipo de sistema destaca-se o especialista, pessoa com conhecimento e habilidades específicas capazes de resolver problemas em sua área de conhecimento e atuação, a qual deve ser a mesma do SEP em desenvolvimento.

O que caracteriza um sistema especialista probabilístico é o seu alto grau de conhecimento sobre uma área específica, sendo então muito eficientes em suas

[1] Termo que dá ênfase. (FERREIRA, 2004).

respostas. Os SEP devem também ter a capacidade de informar ao usuário como obteve a resposta para um determinado problema. Existem diversos benefícios associados ao desenvolvimento de um SEP como a distribuição de conhecimento especializado, memória institucional, flexibilidade no fornecimento de serviços, possibilidade de tratar informações a partir de conhecimentos incompletos ou incertos, entre outros (HUDSON; COHEN, 1999).

Conclui-se que um sistema especialista probabilístico é um programa inteligente de computador que usa procedimentos e conhecimentos inferênciais, para resolver problemas que são bastante complicados, de forma a requererem para sua solução muita perícia humana (WILLIAMSON, 2005).

Os dados incompletos ou incertos adquiridos por meio do especialista são representados no SEP por meio de uma relação hierárquica conforme descreve sua arquitetura no item a seguir.

2.1 ARQUITETURA DE SISTEMA ESPECIALISTA PROBABILÍSTICO

Conforme Campos e Saito (2004) descrevem a arquitetura de um SEP consiste basicamente de um conjunto de dados de entrada, onde o engenheiro do conhecimento desenvolve a base de conhecimento, tendo como base o conhecimento de um especialista humano; um motor de inferência onde são realizados os cálculos probabilísticos dos dados que se encontram na memória de trabalho; e uma interface com o usuário para a entrada/saída de dados e resultados. Nessa arquitetura o engenheiro do conhecimento desenvolve o protótipo do SEP, bem como a interface com o usuário, trabalhando junto com o especialista para o aperfeiçoamento do mesmo.

Na Figura 1 é ilustrada a arquitetura de um SEP.

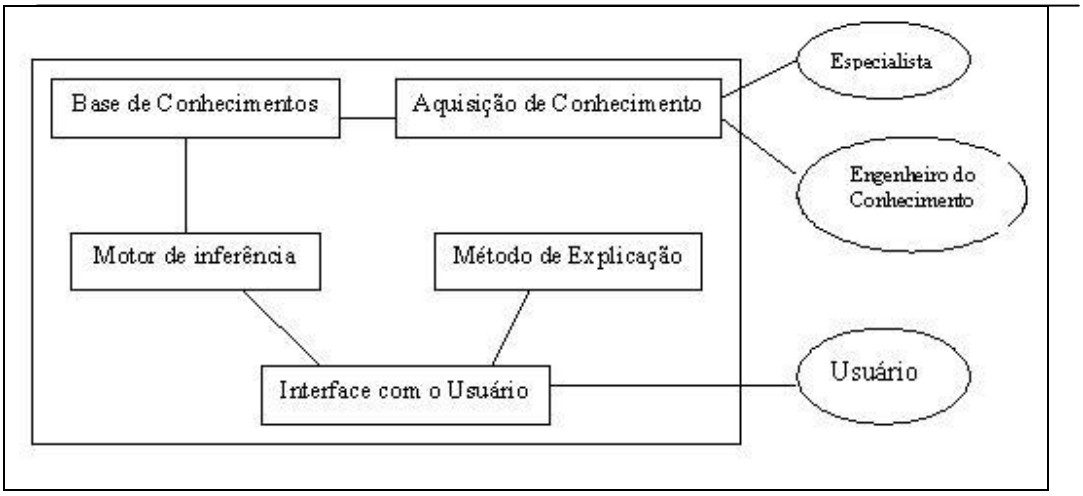

Figura 1 . Arquitetura de um Sistema Especialista Probabilístico
Fonte: Adaptado de JOSÈ (2004)

2.1.1 Base de Conhecimento

A Base de Conhecimento (BC) de um SEP é representada por meio de redes bayesianas, sendo o conhecimento representado esquematicamente na forma de um grafo acíclico direcionado[2]. Os *nós* do grafo representam dois tipos de variáveis: as variáveis de entrada (sinais e evidências) e as variáveis de saída (conjunto de hipóteses diagnósticas) (HUDSON, COHEN, 1999).

Conforme Marques e Marin (2002) relatam a BC dá as características para o funcionamento do SEP, podendo ser formada por meio de pesquisas validadas, de observações e experimentos clínicos, da autoridade de quem a desenvolve, da prática do cotidiano, de livros-textos e de base de dados. Estas informações e dados que formam a BC significam a coleção de dados do conhecimento do domínio, ou seja, as informações necessárias para resolver problemas de um domínio específico, portanto, uma base de conhecimento pode ser vista como um conjunto de regras, cada qual podendo ser validada independentemente de estrutura de controle.

2.1.2 Motor de Inferência

O Motor de Inferência (MI) é o processador do conhecimento do SEP, ou seja, é o elemento mais importante, o núcleo do sistema, pois tem como principal função inferir conclusões e gerar novos fatos (ROVER, 2001).

A interpretação do conhecimento é reduzida por meio de técnicas como a dedução e a lógica probabilísticas onde o usuário pode solicitar o conhecimento armazenado na BC também é vista como motor de inferência (MARQUES; MARIN, 2002).

O MI aplica o conhecimento à solução de problemas reais e executa o ciclo de controle reconhece-atua com o uso da memória de trabalho. Os procedimentos que implementam o ciclo de controle executam duas tarefas de maneira simples: na primeira tarefa os dados armazenados na base de conhecimento e na memória de trabalho são examinados e determina-se quais fatos já são conhecidos, a partir daí tem-se a segunda tarefa que trata e decide em que ordem são tiradas as inferências, tendo como objetivo descobrir novas informações a partir das já existentes. O MI conduz consultas ao usuário transferindo fatos e regras, que poderam ser utilizados na memória de trabalho (LUGER, 2004).

Por seu intermédio que os fatos, regras e heurísticas que compõem a base de conhecimento são aplicadas no processo de resolução do problema. A capacidade do motor de inferência é baseada em uma combinação de procedimentos de raciocínio.

---

[2] Quando as arestas têm uma direção associada e indicada por setas na representação gráfica (LUGER, 2004).

2.1.3 Interface com o Usuário

A interface com o usuário procura tornar o uso do sistema fácil e agradável, eliminando-se as suas complexidades (SPIEGELHALTER, 2004).

Entre seus componentes, os SEP devem possuir uma interface homem-máquina que, segundo Bittencourt (2001) é parte fundamental para o sucesso de qualquer *software*. Por sua vez o SEP não é diferente, sendo por meio de sua interface que ocorre a comunicação entre homem (usuário final) e máquina, sendo ainda que esta pode se apresentar de duas maneiras: por meio de linguagem natural ou interface gráfica.

Estudos de Spiegelhalter (2004) mostram que a interface deve ser bem projetada, baseando-se em técnicas interativas como menus, gráficos, cores, janelas, pois deve ser considerar o grau de familiarização com o usuário, considerando-se que estes podem ser especialistas no assunto ou iniciantes, sendo que, toda informação solicitada e mensagem enviada ao usuário deve ser de forma direta, clara e compreensível pelo mesmo. Para que isto ocorra é necessário que a interface com o usuário seja bastante flexível, assim, a interação entre sistema e usuário conduz a um processo de navegação eficiente com a base de conhecimento.

A interface permitirá que o usuário descreva o problema ou os objetivos que deseja alcançar. Permite, ainda, que usuário e sistema adotem um modelo estruturado de consultas facilitando o processo de recuperação do caminho percorrido pelo sistema em tentativas de solucionar o problema.

2.1.4 Aquisição de Conhecimento

A aquisição de conhecimento é uma tarefa fundamental para a construção da base de conhecimento, sendo a parte mais sensível no processo de construção e atualização da BC de um SEP.

Segundo Bittencourt (2001), a aquisição de conhecimento é a maneira de reunir informações de um ou mais especialistas e organizá-las, ou seja, analisar, interpretar e precodificar o conhecimento para uma forma compreensível pela máquina. Estas informações são adquiridas por meio do engenheiro do conhecimento e especialistas possibilitando a produção de uma documentação de forma ordenada onde as mesmas são traduzidas posteriormente para a base de conhecimento do SEP.

O processo de aquisição de conhecimento pode ser realizado de duas formas: o modelo clássico por meio de entrevistas; e a forma automatizada pela aprendizagem bayesiana (GUINZANI et al, 2006).

No que se refere ao modelo clássico, a aquisição de conhecimento pode ocorrer por meio de entrevistas do engenheiro do conhecimento com o especialista do domínio da aplicação, sendo que este é o responsável em transmitir o conhecimento, e muitas vezes pode não deixar clara e bem definida sua maneira de raciocinar, da possibilidade do processo tornar-se ineficiente e consumir muito tempo.

O especialista encontra dificuldades em associar valores de probabilidades condicionais que compõem a parte quantitativa, bem como estabelecer as relações que compõem a parte qualitativa de uma rede bayesiana ao grau de solução dos problemas. (FLORES; PEROTTO; VICCARI, 2003; REZENDE, 2003; TIBIRIÇÀ; NASSAR, 2003;).

Já na forma automatizada, que diz a respeito a aprendizagem bayesiana[3], ou também denominada mineração de dados em rede bayesianas, pode-se prever a participação do especialista somente na supervisão do processo de aquisição de conhecimento (REZENDE, 2003).

A mineração de dados por meio da descoberta de conhecimento é baseada na tecnologia de redes bayesianas representam uma área de pesquisa muito difundida nos últimos anos e de grande relevância na IA com muitas aplicações já desenvolvidas (ANTUNES, 2002); (BOGO, 2005), sendo capaz de estimar os valores das probabilidades e também identificar os nós da rede a partir de base de dados. Dessa forma, este processo visa minimizar o tempo gasto no processo de aquisição de conhecimento, tornando-o mais rápido e consequentemente mais eficiente (KOEHLER, 2002).

Na AC dois personagens podem ser identificados: o engenheiro do conhecimento e o especialista.

2.1.4.1 Personagens

O engenheiro do conhecimento é a pessoa responsável pelo processo de aquisição de conhecimento. Ele é o especialista em representações do conhecimento e sua tarefa principal é auxiliar o especialista do domínio a articular o conhecimento necessário para o desenvolvimento do SEP, além de transformar as informações adquiridas em dados manipuláveis pelo computador (LUGER, 2004).

O engenheiro do conhecimento ainda deve possuir algumas funções como:

[3] Aprendizagem utilizada para combinar acontecimentos semelhantes por meio de algoritmos (REZENDE, 2003).

a) esclarecer ao especialista como se dá o processo de aquisição de conhecimento (se possível sempre exemplificar para um melhor entendimento);

b) gerenciar a aquisição de conhecimento por meio de cronogramas e metas a serem atingidas, sempre envolvendo o especialista.

c) concretizar o conhecimento por meio da base de conhecimento

d) construir o SEP propriamente dito;

e) validar e comparar as informações e resultados entre SEP e especialista;

f) desenvolver a interface com o usuário;

g) treinar e dar suporte aos possíveis usuários.

O especialista do domínio de aplicação fornece o conhecimento da área do problema e deve possuir experiências e formação na área específica do domínio em que a base de conhecimento será construída. É a pessoa responsável em repassar as informações para o engenheiro do conhecimento.

Este especialista deve apresentar algumas qualidades que são determinantes na hora da escolha do mesmo, qualidades estas que o engenheiro tem a responsabilidade de avaliar antes de realizar a escolha (LUGER, 2004). Neste contexto, algumas características que o especialista deve apresentar são descritas a seguir:

a) possuir conhecimento especializado e reconhecimento superior em sua área de atuação;

b) saber usar e organizar seus conhecimentos de forma adequada;

a) reconhecer suas limitações e admitir problemas fora da sua área de competência;

c) reflexão sobre o seu próprio estado de domínio do conhecimento;

d) capacidade e disposição de expressar seus conhecimentos;

e) motivação de um projeto compreendendo pelo menos a sistemática do mesmo, além de possuir conhecimento das potencialidades do computador;

f) disposição de tempo para auxiliar no desenvolvimento do projeto sendo que o mesmo pode ser um processo de longo prazo cujas experiências são difíceis de adquirir onde estas devem ser confidenciais.

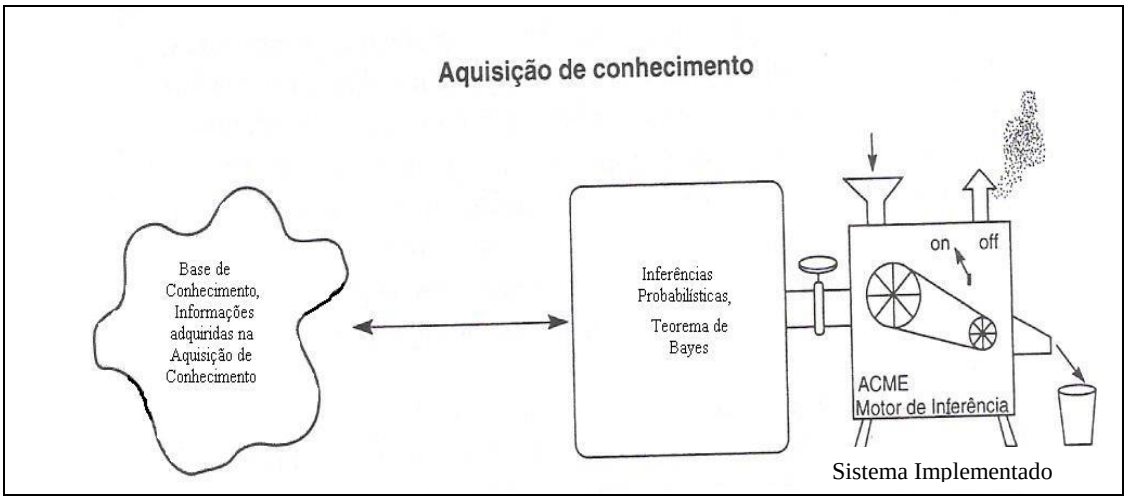

Figura 2 . A visão padrão da construção de um Sistema Especialista Probabilístico.
Fonte: Adaptado de LUGER, G. F. (2004)

A Figura 2 ilustra que a aquisição de conhecimento é um processo para a compreensão dos problemas e incertezas originadas durante a formalização do conhecimento pelo especialista. Este conhecimento é muitas vezes vago e impreciso em suas inferências e probabilidades, com isso o engenheiro do conhecimento deve traduzir e interpretar as informações adquiridas durante a aquisição de conhecimento em inferências probabilísticas com uma linguagem mais formal para que o sistemas possam ser construídos (LUGER, 2004).

Durante a aquisição de conhecimento o especialista raciocina por meio de incertezas na tomada de decisão, e sua forma de tratamento pode ser solucionada pela incerteza por aleatoriedade.

2.2 TRATAMENTO DE INCERTEZA POR ALEATORIEDADE

Conforme Cozman (2002) descreve, uma das atividades mais difíceis de se solucionar pela Inteligência Artificial é a manipulação de incertezas, e nesse sentido, técnicas como o raciocínio probabilístico, estão sendo utilizadas para o apoio nesta área onde as informações são vagas e incompletas.

Por meio de representações gráficas das evidências probabilísticas é possível manipular a incerteza com base em princípios matemáticos fundamentados e modelar o conhecimento do especialista do domínio de forma intuitiva (RUSSEL; NORVIG, 2004).

Um dos fatores de destaque do raciocínio probabilístico refere-se a complexidade dos métodos de inferência, pois devido a presença desta característica, ainda é complicado imaginar a manipulação inteligente de incertezas (RUSSEL; NORVIG, 2004).

Segundo estudo realizado por Flores, Perotto e Viccari (2003) o raciocínio probabilístico permite melhorar o desempenho dos sistemas que apresentam a incerteza por aleatoriedade, considerando-se sua grande aceitação em publicações nacionais e internacionais.

A incerteza está presente em muitos casos, sendo que estes necessitam de alguma tomada de decisão baseada em fatos que representam uma verdade absoluta. Este raciocínio pode levar a várias conclusões e com diversas alternativas para determinado problema.

Assim, existem várias técnicas de IA que auxiliam a tomar decisões para implementar um sistema. Entre as diferentes abordagens encontram-se as Redes Bayesianas que são utilizadas para tratar a incerteza por aleatoriedade (probabilidade) e

a outra abordagem existente é Sistemas Especialistas *difuso* que tratam da incerteza por imprecisão (possibilidade). Em alguns casos de domínio coexistem os dois tipos de abordagens de incerteza: a imprecisão e a aleatoriedade tornando um sistema híbrido (TIBIRIÇÁ; NASSAR, 2003).

Conforme apresentado anteriormente, a imperfeição da informação geralmente é conhecida como incerteza, o conceito de incerteza volta-se as informações quando na maioria de seus casos não se têm acesso a toda verdade sobre o seu ambiente, as informações que estes possuem não garantem quaisquer resultados, mas, porém, podem fornecer algum grau de crença de que os resultados poderão ser alcançados. Sendo assim, chega-se a um raciocínio de que a decisão tomada pelo especialista deve ser de forma racional, pois, depende tanto da importância relativa de várias metas quanto da probabilidade de que elas serão alcançadas (RUSSEL; NORVIG, 2004).

Representar e manipular o conhecimento em um domínio de área particular, sempre envolve uma certa complexidade. Na área médica é comum raciocinar sob incerteza, em que por exemplo pode-se ter sintomas comuns a várias doenças. Nestes casos o médico especialista deve indicar tratamentos baseados em evidências já ocorridas e posteriormente já constatadas por meio de exames clínicos (WILLIAMSON, 2005).

As pessoas realizam simplificações para resolver problemas e tomar decisões em ambientes reais onde as informações que circulam podem ser parciais ou aproximadas. Para resolver estes problemas apresentam-se apenas soluções com incerteza e esta pode estar presente nos dados de entrada ou até mesmo nos resultados adquiridos (LADEIRA; FLORES; VICCARI, 2002).

Estas incertezas são modificadas periodicamente após observações de novos dados ou resultados gerados por meio dos fatos e regras. A operação que potencializa a

medida das incertezas é conhecida como operação bayesiana sendo baseada na fórmula de Bayes[4] e pelo conceito de RB (RUSSEL; NORVIG, 2004).

[4] Teorema que calcula a probabilidade de uma hipótese dado uma nova evidência (RUSSEL; NORVIG, 2004).

3 REDES BAYESIANAS

A Rede Bayesiana (RB) é um modelo gráfico matemático que tem como estrutura um grafo direcionado acíclico que representa a distribuição de probabilidades conjuntas das variáveis que modelam o domínio do conhecimento incerto (LADEIRA; FLORES; VICCARI, 2002), (LADEIRA et al, 2003).

A RB permite expressar as assertivas de independência de forma visual e de fácil percepção; representa e armazena uma distribuição conjunta de probabilidades condicionais de forma sucinta, explorando a esparcidade da rede no relacionamento entre as variáveis, ou seja, quantificando a intensidade das relações nas variáveis, tornando assim o processo de inferência eficiente computacionalmente (SPIEGELHALTER, 2004).

As redes bayesianas permitem analisar grandes quantidades de dados para extrair conhecimentos úteis em tomada de decisões do especialista, controlar ou prever o comportamento de um sistema, diagnosticar as causas de um fenômeno, entre outros comportamentos. São utilizadas em vários domínios como: saúde (diagnóstico de doenças exantemáticas (ANTUNES, 2002)); indústria (controle de automação (BONGIOLO, 2002)); computação e redes (agentes inteligentes); marketing (mineração de dados, gestão da relação com os clientes); banca e finanças (análise financeira); gestão (tomada de decisões, gestão de conhecimento e risco) (LUNA, 2004).

Para a modelagem destes problemas e a transformação das suas probabilidades de ocorrência em uma RB, é preciso modelar sua arquitetura dado que suas inferências são denominadas sob conhecimento do especialista do domínio, e sua topologia para a representação destas informações.

3.1 TOPOLOGIA E ARQUITETURA

A topologia de uma RB representa um modelo probabilístico completo, com representação das informações qualitativas (dependência, probabilidade) e quantitativas (distribuição de probabilidade, valores) (LADEIRA; VICCARI; COELHO, 2004).

A topologia surge por meio do especialista do domínio, de modelos causais disponíveis na literatura do domínio em questão, podendo ser aprendida também diretamente a partir de dados históricos ou base de dados. As probabilidades podem ser fornecidas pelo especialista do domínio, obtidas em estudos estatísticos publicados, obtidas analiticamente por meio da aplicação da análise combinatória para domínios específicos como a genética, ou ainda, aprendidas diretamente a partir de dados históricos (LADEIRA et al, 2003).

Quando um especialista cria a estrutura de uma rede bayesiana, é necessário estimar ou aprender as probabilidades associadas a cada variável (JENSEN, 2001).

À medida que o número de variáveis cresce em um modelo, dificulta ao especialista definir a relação de dependência entre elas, no entanto, é vantajoso utilizar um banco de dados para realizar a inferência na estrutura da rede bayesiana e suas probabilidades (WILLIAMSON, 2005).

A rede bayesiana em sua arquitetura é definida com a parte qualitativa que corresponde à estrutura gráfica da rede, composta pelas variáveis de entrada (evidência) e as variáveis de saída (hipóteses), onde as variáveis são os nós, e os arcos direcionados as regras que representam as relações de dependência entre as variáveis (NASSAR, 2006).

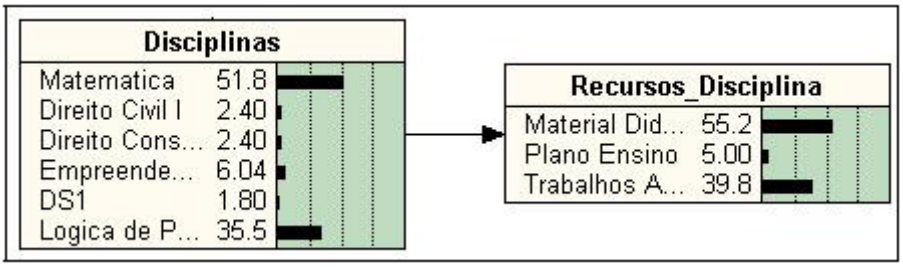

Figura 3 . Exemplo de uma estrutura de RB
Fonte: FERRARI, et al (2003).

Conforme ilustra a Figura 3 a parte quantitativa da rede bayesiana corresponde aos valores de probabilidade da base de conhecimento da rede, ou seja, é o conjunto de probabilidades condicionais (PC) associadas aos arcos existentes no modelo gráfico da parte qualitativa, e as probabilidades estimadas *a priori* das hipóteses diagnosticadas.

O grafo[5] resultante é tido como a parte qualitativa, sendo utilizado para representar a estrutura e a complexidade tanto do problema como dos procedimentos de busca (LUGER, 2004).

Os nós representam variáveis e os arcos significam a influência causal entre variáveis diretamente ligadas, sendo que a intensidade destas influências são expressas por meio das probabilidades condicionais (SAHEKI, 2005).

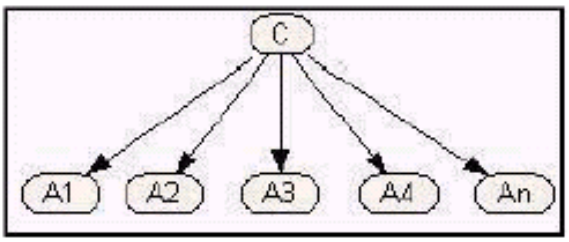

Figura 4 . Exemplo gráfico de uma Rede Bayesiana
Fonte: FERRARI, et al (2003).

Considerando-se que a RB é um modelo gráfico acíclico orientado, a Figura 4 ilustra que os seus nós representam as variáveis aleatórias e o arco unindo dois nós representa a dependência probabilística entre as variáveis associadas. Cada um destes

[5] O grafo consiste no conjunto de nós e um conjunto de arcos conectando pares de nós (LUGER, 2004).

nós (A1, A2, A3, A4, An) é associado a uma função de distribuição de probabilidade condicional dos valores que podem ser assumidos pela variável aleatória associada ao nó, dado os valores de seus nós pais (C) (WILLIAMSON, 2005).

Associado ao grafo, existe uma representação de probabilidades. As redes bayesianas adotam uma representação compacta onde são definidas somente as probabilidades condicionais de cada nó em relação aos seus pais. Para a aplicação das redes bayesianas é necessário que se obedeça à condição de Markov[6], que indica não existir uma relação de dependência direta entre quaisquer dois nós a não ser que exista um arco entre eles na rede. A distribuição de probabilidades correspondente à rede é calculada a partir de probabilidades condicionais (SAHEKI, 2005).

Saheki (2005) ainda afirma que em aplicações práticas, os valores das probabilidades conjuntas não são muito significativos na análise de alguns problemas modelados. De maior interesse são as probabilidades *a posteriori* de cada nó não observado. Fazendo uso destas probabilidades conjuntas, pode-se obter as probabilidades posteriores somando-se para cada estado de cada variável, todas as probabilidades em que a variável encontra-se no estado desejado. Normalizam-se as probabilidades obtidas e encontrando-se as probabilidades a *posteriori* para cada nó.

[6] Os estados anteriores são irrelevantes para a definição dos estados seguintes, desde que o estado atual seja conhecido (SAHEKI, 2005).

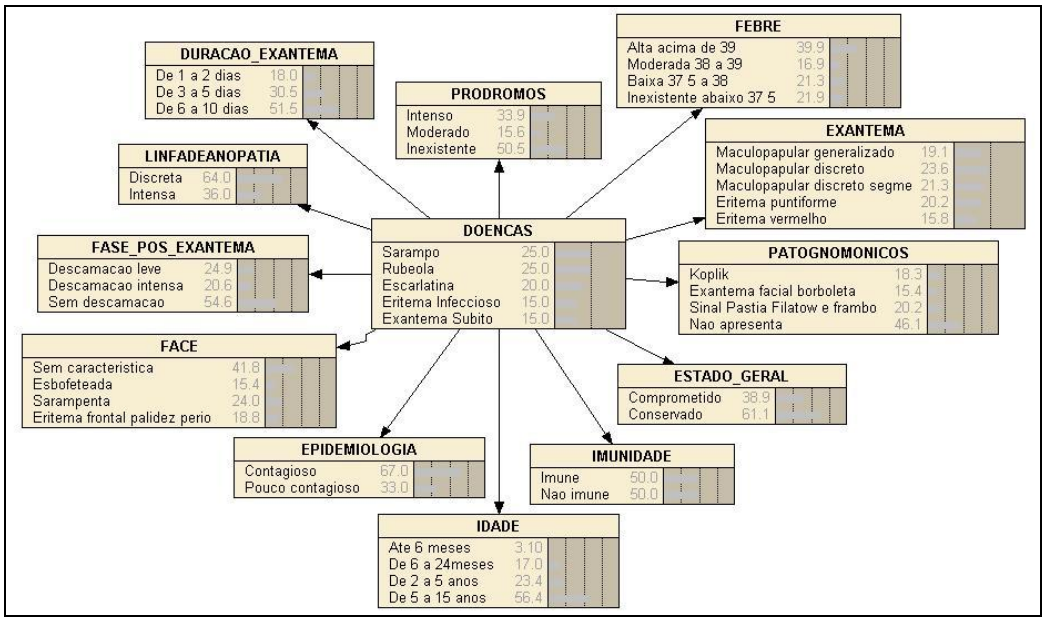

Figura 5 . Rede Bayesiana SEDDEM (Nós, evidencias, probabilidades)

Fonte: ANTUNES, L. (2002).

A Figura 5 ilustra os componentes da RB SEDDEM, onde mostram os *nós* de evidências, hipóteses de diagnósticos, as probabilidades a *priori* de cada evidência e a relação entre os *nós*.

Os casos mais comuns do conhecimento podem oferecer apenas um grau de crença nas sentenças mais relevantes, sendo assim o formalismo ideal para essa problemática é a teoria da probabilidade, sendo que no próximo item será abordada a maneira de realizar inferência por meio do Teorema de Bayes.

3.2 TEOREMA DE BAYES

Em 1973 surgiu o Teorema de Bayes, por meio de uma publicação realizada pelo matemático Thomas Bayes, da pequena cidade inglesa Tunbridge Wells, que formalizou seu teorema, indicando que o cálculo quando expressado matematicamente é um simples resultado em teoria da probabilidade (SPIEGELHALTER, 2004).

Conforme afirmação de Costa e Simões (2004) a teoria da probabilidade é

um método quantitativo de tratamento do conhecimento incerto, onde não se pode prever com toda certeza o que acontecerá num novo caso, sendo ainda que a mesma representa a incerteza por aleatoriedade por meio de valores numéricos simples (0 e 1) e atribui os mesmos em um teorema chamado Teorema de Bayes que fundamenta as inferências como uma conseqüência ou como uma causa.

A estas causas atribuem-se valores e graus conforme a crença nas regras e nos fatos já acontecidos, estes ainda podem vir com incertezas em relação a força deste grau ou valor atribuído. Com a incerteza e insatisfação dos graus de crença em assuntos de grande importância, tem-se a oportunidade de ser mais exato nos graus e atribuí-los valores numéricos. Para que estas crenças sejam melhores definidas aconselha-se usá-las com regras que as associem (TARONI, 2006).

O raciocínio deste teorema é baseado na realização de inferências probabilísticas, ou seja, no cálculo da probabilidade condicional de um evento, dadas todas as evidências disponíveis (LADEIRA; VICCARI; COELHO, 2004).

Na utilização do teorema de Bayes, a ocorrência de independência condicional entre variáveis aleatórias que descrevem os dados pode simplificar os cálculos para responder perguntas e também reduzir consideravelmente o número de probabilidades condicionais que precisam ser especificadas. Nesse sentido, as redes bayesianas representam a dependência entre as variáveis, oferecendo uma especificação concisa da distribuição das probabilidades conjuntas (LUNA, 2004).

Conforme estudos e afirmações de Tibiriçá e Nassar (2003) o teorema de Bayes indica uma maneira de organizar as regras de probabilidade a fim de atualizar as probabilidades *a priori*, ou crenças iniciais, dada a nova informação, resultando na crença *a posteriori*. Sendo assim, esse processo de inferência é o responsável por gerar os valores de probabilidades de cada estado da variável de saída, definindo as relações

causais entre entradas, saídas e os valores probabilísticos quantificados e refletidos entre eles mesmos.

Esse teorema também é responsável pelo tratamento da imperfeição da informação nos sistemas baseados em conhecimento, segundo afirmações de Taroni (2006), baseando-se nas expressões SE ENTÃO, que em RB funciona conforme especificado abaixo:

SE CAUSA ENTÃO CONSEQUÊNCIA

representado por: P(CONSEQUENCIA/CAUSA)

Um exemplo desta expressão pode ser demonstrado da seguinte maneira: dado que uma pessoa está sentido dor de dente, qual a probabilidade, ou seja, chance da dor está sendo provacada por uma cárie?

SE Cárie ENTÃO DorDente

P(DorDente / Carie)

Depois de se adicionar valores e probabilidades para a sua propagação de inferência, realizam-se os cálculos probabilísticos, obtendo-se os valores ou as chances da dor de dente estar sendo provocada pela cárie.

A probabilidade condicional é vista com uma medida de crença no evento, dadas as evidências disponíveis (LADEIRA;VICCARI; COELHO, 2004).

O espaço de probabilidade (ε, P) e os eventos compostos *e,* H_1, H_2,...,H_k $\underline{C}\varepsilon$, desde que nenhum desses eventos tenha probabilidade nula, então:

$$P(Hi / e) = \frac{P(e / Hi).P(Hi)}{P(e)} \qquad \text{(a)}$$

Onde:

a) a probabilidade para todo P (H_i ^ e) ≠ 0 para todo i;

b) os eventos H_1 U H_2 U ...U H_k = ε e $H_i \cap H_j$ = ø para todo o $i \neq j$ (isto é, os Hi formam uma partição do espaço ε)

Logo: $e = (H_1 \cap e)\ U\ (H_2 \cap e)\ U\ ...U\ (H_m \cap e)$

$P(e) - P(H_1).P(e/H_1) + P(H_2).P(e/H_2) + ...+ P(H_k).P(e/H_k)$

Resultando em:

$$P(Hi/e)\frac{-P(e/Hi).P(Hi)}{\sum_{j-1}^{k}(P(Hj).P(e/Hj))} \quad \text{(b)}$$

Onde:

P(Hi\e): probabilidade de que a hipótese Hi seja verdadeira, dada a evidência E.

P(e\Hi): probabilidade de que observa-se a evidência E, dado que a hipótese Hi é verdadeira.

P(Hi): probabilidade a *priori* de que a hipótese i seja verdadeira na ausência de qualquer evidência específica. Estas probabilidades são chamadas de probabilidades prévias, ou simplesmente prévias.

K: número de hipóteses possíveis.

Nas RB os Hi são as hipóteses concorrentes (saída). O evento *e* pode ser pensado como uma evidência (entrada). O conhecimento da ocorrência desta evidência leva a mudanças na probabilidade a *priori* P(Hi) para a probabilidade condicional P(Hi/*e*), que por sua vez considera a evidência *e*.

As redes bayesianas apresentam como resultado um vetor de probabilidades,

onde a cada categoria da variável de saída é atribuída a sua respectiva probabilidade de acordo com as evidências informadas na inferência bayesiana descrita a seguir.

3.3 INFERÊNCIA EM REDES BAYESIANAS

O processo de inferência em redes bayesianas mostra que, uma vez construída uma representação probabilística por meio do modelo de RBs, para a incerteza presente no relacionamento entre variáveis de um domínio de dados, uma das tarefas mais importantes consiste em obter estimativas de probabilidades de eventos relacionados aos dados, à medida que novas informações ou evidências sejam conhecidas (RUSSEL; NORVIG, 2005).

A importância da inferência bayesiana não é restrita somente ao cálculo de probabilidades interessantes ao surgimento de novas evidências, podendo-se também tirar proveito da sua funcionalidade no processo de aprendizagem de RB. Este processo pode ocorrer de duas maneiras: estrutura gráfica e parâmetros numéricos; sendo que estes processos iniciam por meio dos dados incompletos na amostra da base de dados, onde a inferência é utilizada para estimar estes valores faltosos (LUNA, 2004).

A tarefa fundamental nas RB é calcular a distribuição de probabilidade a *posteriori* para um conjunto de variáveis de consulta, dado algum evento observado – isto é, alguma distribuição de valores a um conjunto de variáveis de evidência (RUSSEL; NORVIG, 2005).

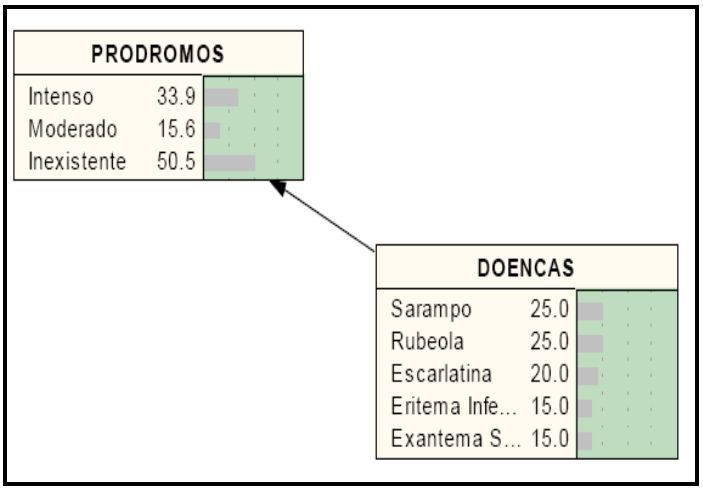

Figura 6 . Calculo inicial da base SEDDEM
Fonte: ANTUNES, L. (2002)

Na Figura 6 é demonstra como realiza a consulta dado uma nova hipótese, ou seja, dado que o apresentar PRODROMOS INTENSOS.

Sendo assim a Figura 7 ilustra a fórmula utilizada para encontrar o resultado do cálculo probabilístico, nela está contida o valor INTENSO do nodo PRODROMOS em relação ao nodo DOENÇA. Ou seja, o sistema baseia-se nas probabilidades iniciais de DOENÇA (25%SARAMPO, 25%RUBEOLA, 20%ESCARLATINA, 15%ERITEMA e 15%EXANTEMA) e faz este cálculo em relação às probabilidades iniciais do nodo PRODROMOS onde está contido INTENSO. Basicamente no início ele se baseia nos valores de entrada adquiridos no momento da representação do conhecimento para a obtenção dos resultados.

P(INTENSO) = (P(SARAMPO) ∩ P(INTENSO/SARAMPO) U P(RUBEOLA) ∩ P(INTENSO/RUBEOLA) U (P(ESCARLATINA) ∩ P(INTENSO/ESCARLATINA) U (P(ERITEMA) ∩ P(INTENSO/ERITEMA) U (P(EXANTEMA) ∩ P(INTENSO/EXANTEMA)).
P(INTENSO) = (P(SARAMPO) * P(INTENSO/SARAMPO) + P(RUBEOLA) * P(INTENSO/RUBEOLA) + (P(ESCARLATINA) * P(INTENSO/ESCARLATINA) + (P(ERITEMA) * P(INTENSO/ERITEMA) + (P(EXANTEMA) * P(INTENSO/EXANTEMA)).

P(**INTENSO**) = (0,25) * (0,75) + (0,25) * (0,01) + (0,2) * (0,7) + (0,15) * (0,01) + (0,15) * (0,05)
P(**INTENSO**) = (0,1875) + (0,0025) + (0,14) + (0,0015) + (0,0075)
P(**INTENSO**) = 0,339 ou 33,9%

Figura 7 . Calculo probabilístico na base SEDDEM
Fonte: ANTUNES (2002)

Para a construção de uma Rede Bayesiana os sistemas utilizam representações gráficas de dependências probabilísticas. Essa representação permite manipular a incerteza com base em princípios matemáticos fundamentados, além de modelar o conhecimento do especialista do domínio de uma forma intuitiva.

Para isso são utilizadas algumas ferramentas de apoio a construção dessas redes, e entre essas, a *shell* Netica[7] (Norsys) (FLORES; PEROTTO; VICCARI, 2003) .

[7] . Disponível em: http://www.norsys.com.

4 *SHELL* NETICA

Atualmente no mercado existem muitas ferramentas para a construção de RB, desde as gratuitas até as comerciais. Dentre estas ferramentas destaca-se a *shell* Netica, desenvolvida pela empresa *Norsys Software Corporation* de Vancouver, Canadá. Esta *shell* auxilia a criação de RB possibilitando a integração com outros ambientes e linguagens de programação incluindo-se o desenvolvimento da interface com o usuário.

Nesta *shell*, o usuário especifica a entrada de distribuições que caracterizam o conhecimento disponível e por meio de cálculos baseado no Teorema de Bayes (a) são visualizadas as saídas que a ferramenta realiza nos cálculos de inferência (BRANDT; RENNIE, 2004).

A s*hell* Netica permite realizar vários tipos de inferência usando algoritmos modernos e rápidos. Dado um novo caso, que o usuário tem conhecimento limitado, essa *shell* encontra os valores ou probabilidades apropriadas para todas as variáveis desconhecidas. Entre suas características destaca-se a utilização de diagramas de influência[8] para representar as melhores decisões, que maximizam os valores esperados das variáveis especificadas (NASSAR, 2006).

O fabricante Norsys relata que por se basear na inferência bayesiana, essa *shell* permite por meio das entradas, a quantificação do resultado em porcentagem de todas as variáveis relacionadas no sistema, ou seja, determinando a probabilidade de algo acontecer de acordo com os valores iniciais.

A Figura 8, ilustra uma RB no momento de execução da propagação de uma evidência, onde a cada evidência propagada pelo usuário as probabilidades condicionais

[8] Forma de representar graficamente que uma situação X pode influenciar nos resultados da situação Y (Nassar, 2006).

são alteradas por meio de cálculos que indicam o diagnóstico por meio da maior probabilidade de condicional.

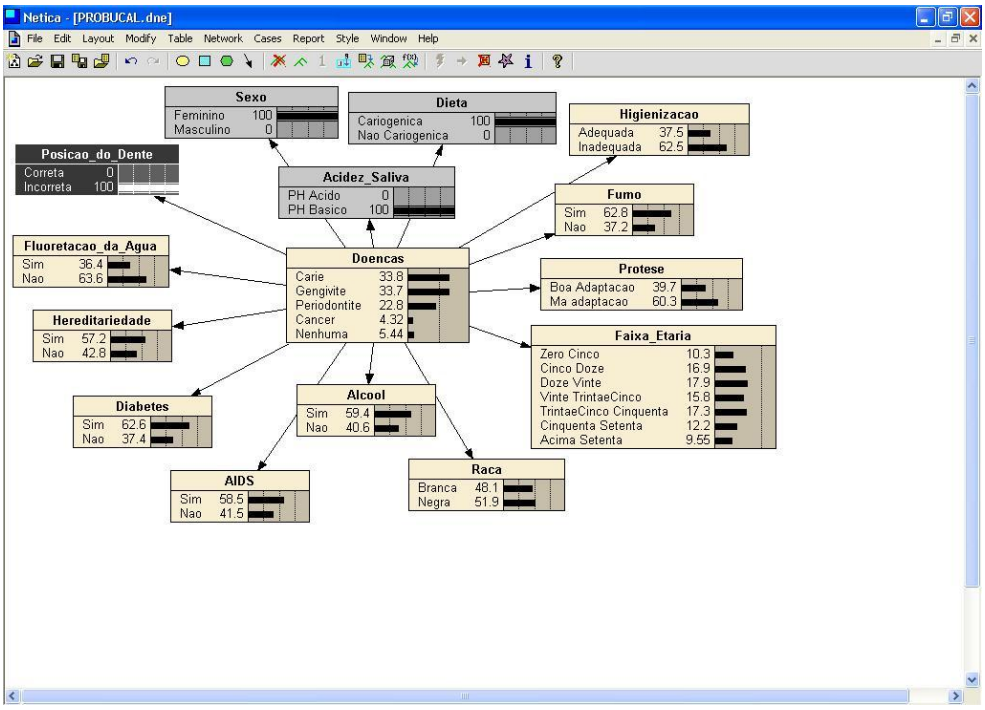

Figura 8 . Apresentação gráfica da inferência realizada em uma RB.
Fonte: ROSA (2002)

No exemplo da Figura 8 na RB onde as seguintes evidências estão selecionadas: *Incorreta* para a *Posição do Dente*; *Feminino* para o *Sexo*; *PH Básico* para a *Acidez da Saliva* e *Cariogênica* para a *Dieta*, e a partir desta seleção, os valores indicam que 37,8 % dos pacientes podem estar no vetor de probabilidades *Cárie*[9].

[9] È uma doença infecto-contagiosa degenerativa que destrói os dentes (FERREIRA, 2004).

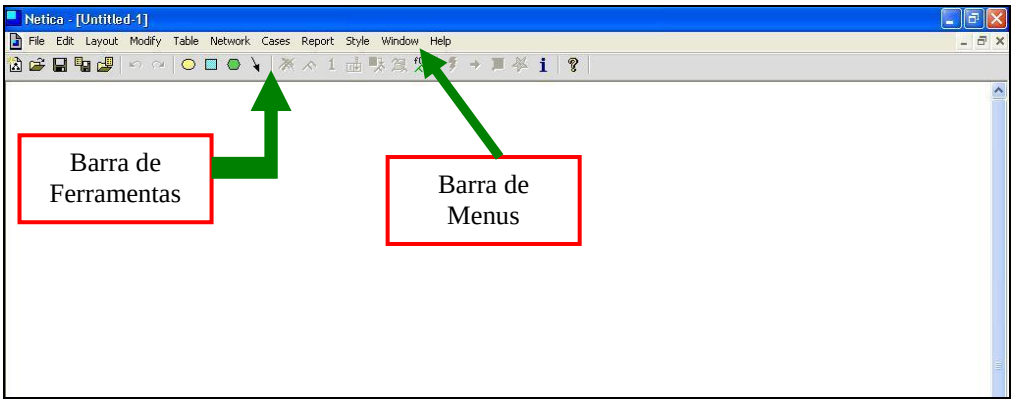

Figura 9 . Ambiente Gráfico da *Shell* Netica

A Figura 9 é possível visualizar a interface que possui diversos menus interativos com opções de criar, modificar e editar a RB, manipular as tabelas de probabilidades condicionais, estilo de formatar a RB, realizar inferência na RB por meio de Cases, além de possuir uma barra de ferramentas que possui atalhos da barra de menus como salvar, criar nova RB, adicionar nós, arcos.

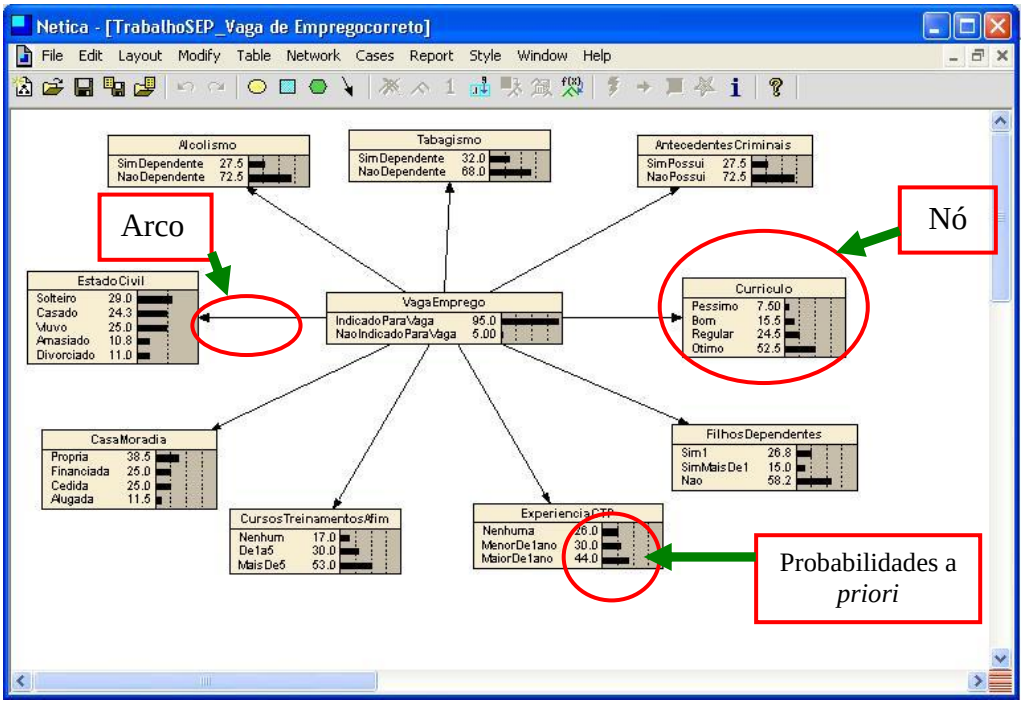

Figura 10 . Componentes da RB.

Como ilustra a Figura 10, esta *shell* possui uma interface gráfica interativa, que permite a criação da Rede Bayesiana que representam as variáveis, além da

definição dos arcos entre esses nós que representam as dependências causais entre essas variáveis, como também atribuir as probabilidades *a priori* para cada nó possibilitando representar a base de conhecimento de um Sistema Especialista Probabilístico.

Ainda segundo o fabricante, o Netica é uma ferramenta para a construção de Redes Bayesianas por possuir uma série de vantagens além das já citadas e características como:

Possibilita a realização da integração no desenvolvimento de uma interface gráfica de qualidade nas linguagens de programação nas quais possui a API é disponibilizada no *site* do fabricante;

Possui o C como programa padrão de interface, podendo ser utilizado por meio de programas escritos em C, C++, Java, Delphi ou qualquer linguagem que podem chamar funções da linguagem C por meio das dll's.

Essa *shell* é composta do Netica *Application* e Netica *API*. O Netica *Application* é a interface gráfica que permite desenvolver a RB de forma gráfica. O Netica *API* apresenta uma biblioteca com funções que representam as diversas opções disponíveis na versão *Application.*

O Netica API permite comunicar a Netica Application com alguns ambientes de programação como, por exemplo, o Java, utilizado para criar a interface do usuário, sendo possível gerenciar as RB por meio dessa biblioteca e do ambiente de programação (Norsys, 2006).

Atualmente, o fabricante deste software oferece algumas API's que permitem a integração entre o Netica e as seguintes linguagens de programação: C; C++; Visual Basic e Java, destacando-se a Netica Java API utilizada no estudo de caso desta pesquisa.

As bibliotecas e funções disponibilizadas pela *shell* Netica para o processo

de integração com um ambiente ou linguagem de programação são várias, dentre elas pode-se destacar a Netica Java API, disponibilizada para o ambiente de programação Java e utilizada no estudo de caso desta pesquisa.

4.1 NETICA JAVA API

O Netica Java *API* dispõe de uma biblioteca com classes e métodos escritos na linguagem C. Estes métodos possibilitam a integração entre a *shell* Netica e algumas linguagens de programação como por exemplo, o Java que por meio por exemplo, do ambiente o NetBeans IDE é utilizado para criar a interface do usuário, onde é possível gerenciar as opções presentes na versão application dessa *shell* (Norsys, 2006).

Segundo Norsys (2006) o site da *shell* disponibiliza o seu manual de utilização com as especificações necessárias para a construção de RB por meio dessa API.

O manual disponibiliza informações sobre os arquivos que se encontram no pacote da Netica Java API, neste pacote se encontram várias pastas (diretórios) com os seus respectivos arquivos e descrição para a sua utilização, conforme descrito nas tabelas abaixo:

Tabela 1 . Conteúdo da pasta *doc* do pacote da Netica Java API

Diretório	***Arquivo/site***	***Descrição***
doc	NeticaJ_Man.pdf	Manual de utilização da Netica Java API
	NeticaAPIMan_C.pdf	Manual da API do Netica C que é base para a Netica Java API
	Javadocs http://www.norsys.com/netica-j/docs/javadocs/index.html	Documentação da Netica Java API
	LicAgree.txt	Documento sobre as licenças de utilização da Netica Java API

Na tabela 1 são descritos os arquivos existentes na pasta *doc*, que contêm o manual de utilização da Netica Java API; bem como o manual da API na linguagem C que segundo o fabricante deve ser utilizado como base para o entendimento das demais API's; o javadoc com a documentação em Java e um documento no formato texto que contêm as especificações de utilização da licença da Netica Java API.

Tabela 2 . Conteúdo da pasta *bin* do pacote da Netica Java API

Diretório	***Arquivo***	***Descrição***
bin	NeticaJ.jar	Classe com as bibliotecas de definições da Netica Java API
	Neticaj.dll	Biblioteca Nativa da Interface Windows
	Netica.dll	Biblioteca Nativa da Netica Java API

A tabela 2 apresenta uma descrição dos arquivos da pasta *bin*, que contêm as bibliotecas necessárias para a utilização da Netica Java API. O arquivo NeticaJ.jar é utilizado para a configuração do projeto no ambiente que utiliza a tecnologia Java, já os arquivos Neticaj.dll e Netica.dll são utilizados para a configuração do ambiente de variáveis do sistema operacional Windows, procedimento descrito no tutorial de instalação da netica Java API apresentado no apêndice A.

Os arquivos de dll são bibliotecas utilizadas para a configuração do ambiente Java e para um correto funcionamento da Netiva Java API.

Tabela 3 . Conteúdo da pasta *src/NeticaEx* do pacote da Netica Java API

Diretório	***Arquivo***	***Descrição***
src/NeticaEx	NetEx.java	Arquivo que contém algumas rotinas que facilitam a utilização da classe Net
	NodeEx.java	Arquivo que contém algumas rotinas que facilitam a utilização da classe Node
	NodeListEx.java	Arquivo que contém algumas rotinas que facilitam a utilização da classe NodeList

Na tabela 3 é ilustrada a estrutura de arquivos no diretório *src/NeticaEx*, que apresenta os arquivos Java que contém as principais classes e métodos utilizados nesta pesquisa.

Tabela 4 . Conteúdo da pasta *src/NeticaEx/aliases* do pacote da Netica Java API

Diretório	***Arquivo***	***Descrição***
src/NeticaEx/aliases	Net.java	Alias da classe NetEx.java
	Node.java	Alias da classe NodeEx.java
	NodeList.java	Alias da classe NodeListEx.java

Na tabela 4 são descritas as classes do diretório *src/NeticaEx/aliases*, neste diretório encontra-se as classe utilizadas para a realização da integração, onde estas classes possuem apenas o seu construtor, estes por sua vez são utilizados nas classes do diretório *src/NeticaEx.*

Tabela 5 . Conteúdo da pasta *src/demo* do pacote da Netica Java API

Diretório	***Arquivo***	***Descrição***
src/demo	Demo.java	Uma amostra do aplicativo para testar sua Netica Java API
	Compile.bat	Uma amostra do lote de compilar o Demo
	Run.bat	Uma amostra de executar o Demo

A tabela 5 descreve o conteúdo do diretório *demo*, onde estes arquivos são uma amostra de como a Netica Java API funciona, pode-se usá-los a fim de testar e comparar a integração desenvolvida com a do exemplo.

Tabela 6 . Conteúdo da pasta *examples* do pacote da Netica Java API

Diretório	***Arquivo***	***Descrição***
examples	BuildNet.java	Descreve como criar uma RB
	DoInference.java	Descreve como realizar inferência na RB
	SimulateCases.java	Descreve como simular um caso estatisticamente

	LearnCPTs.java	Descreve a aprendizagem a partir de casos
	LearnLatent.java	Descreve a aprendizagem pelo algoritmo EM Learning
	ClassifyData.java	Descreve a classificação de dados médicos do mundo real
	MakeDecision.java	Descreve a construção e a escolha da melhor decisão
	DrawNet.java	Descreve a utilização do pacote *gui* para desenhar redes
	NetViewer.java	Descreve a utilização do pacote *gui* para edição redes e suas descobertas
	TestNet.java	Descreve o desempenho do teste na ferramenta
	Compile.bat	Permite compilar todos os arquivos Java deste diretório
	Run.bat	Permite executar todos os programas Java deste diretório, depois de eles foram compilados

No diretório *examples* descrito na tabela 6, são ilustrados os exemplos de como utilizar com o Netica Java API, por exemplo, para a criação de uma RB; realização de inferência; simulação de casos; aprendizagem; classificação dentre outras aplicações da utilização da Netica Java API.

Tabela 7 . Conteúdo da pasta *exemples/Data Files* do pacote da Netica Java API

Diretório	***Arquivo***	***Descrição***
exemples / Data Files	ChestClinic.dne	Um exemplo utilizado pela classe ou arquivo SimulateCases / LearnCPTs / TestNet.java
	BreastCancer.dne	Um exemplo utilizado pela classe ClassifyData.java
	ChestClinic.cas	Um arquivo criado por SimulateCases.java e utilizado pelo TestNet.java
	ChestClinic_WithVisuals.cas	chestClinic.dne mas incluindo todas as dimensões / posição / cor para exibir as informações
	LearnLatent.cas	um processo utilizado pela LearnLatent.java
	BreastCancer.cas	um processo utilizado pela ClassifyData.java

Nos arquivos apresentados na tabela 7 e no diretório *examples / Data Files,* encontram-se exemplos de Redes Bayesianas e base de dados para a realização de integração.

As classes utilizadas nesta pesquisa encontram-se nas tabelas 3 e 4 (diretório src) as quais realizam a chamada dos métodos utilizados na integração com a Netica Java API. Existem muitos métodos e classes disponibilizados cuja a documentação é apresentada pelo javadocs disponibilizados no Apêndice B. A seguir são abordadas as principais classes e seus métodos considerando a posterior utilização no estudo de caso.

4.1.1 Classe Environ

A classe Environ é utilizada para criar um novo ambiente[10] para a RB a ser construída. Para a utilização desta classe é necessário declarar uma variável do tipo da classe e uma variável do tipo *string* onde coloca-se a licença[11] de utilização da API que se faz necessária somente quando a RB possuir mais de 16 *nós*, caso contrário usa-se a definição *null* (NORSYS, 2006).

Figura 11 . Classe Environ

Fonte: NORSYS (2006)

A Figura 11 ilustra a classe Environ, que se encontra no pacote *norsys.netica.*

[10] Espaço de memória alocado para a execução do gerenciamento da RB Norsys (2006).
[11] Atualmente a UNESC possui uma licença para utilização da *shell* Netica e da API.

Dentre os métodos da classe Environ, destaca-se o método *finalize,* utilizado posteriormente no estudo de caso.

Esse método tem como objetivo liberar todos os recursos alocados na memória durante a execução da RB. Depois que o ambiente for finalizado nenhum método da Netica Java API terá validade de execução, pois o ambiente não estará mais preparado para qualquer outra chamada. Ressalta-se que este método é chamado automaticamente quando um objeto (rede) for destruído (NORSYS, 2006).

```
1    protected void finalize () trows Throwable {
2        try {
3            env.finalize ();
4          }catch (NeticaException ne){
5              ne.printStackTrace ();
6          }
7    }
```

Figura 12 . Método finalize da classe Environ
Fonte: NORSYS (2006)

Na Figura 12, a ilustração indica na linha 3 que a forma de fazer a finalização do ambiente de execução da RB é por meio da variável que representa a classe *Environ*. Se faz necessária a utilização do tratamento de exceção para que o endereço do ambiente a ser finalizado seja válido, sendo assim este método finalizará com sucesso o ambiente em execução.

4.1.2 Classe Net

Segundo Norsys (2006), a classe *Net* é utilizada para criar uma nova RB. Esta classe possui vários construtores sobrecarregados que criam e executam a RB de forma diferente no ambiente atual. Entre os construtores existentes nesta classe destaca-se o *Net (Streamer)* utilizado para a leitura de um arquivo de RB. Na execução da

leitura do arquivo, é retornado a referência da nova rede do arquivo ou uma exceção com a informação de que não foi possível ler o arquivo.

```
net = new Net (new Streamer (this.getClass().getResourceAsStream<"/arquivo.dne">, "<nome da RB>", <nome do ambiente>));
```

Figura 13 . Construtor Net (Streamer)

A Figura 13 ilustra que, para a declaração da RB com este construtor é necessário passar como parâmetro o arquivo da RB, a variável que representa a RB e o nome do ambiente criado para esta RB ser executada (NORSYS, 2006).

Onde:

net: é a variável que recebe o valor da nova RB aberta pelo construtor.

arquivo.dne: é o nome do arquivo que contém a RB.

nome da RB: variável do tipo string que recebe o nome dado para a RB.

nome do ambiente: variável do tipo *Environ* que contém o endereço do ambiente criado para realizar o gerenciamento da RB.

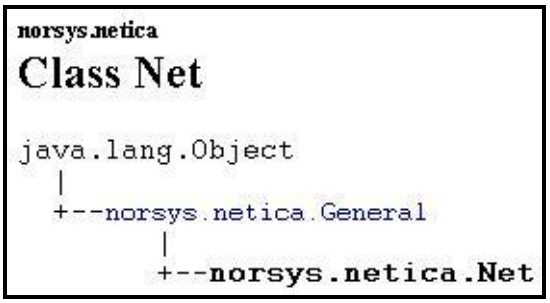

Figura 14 . Classe Net
Fonte: NORSYS (2006)

A Figura 14 ilustra a classe *net* que está no pacote *norsys.netica*, e extende a classe *General* do mesmo pacote.

Entre os métodos existentes na classe *Net* destaca-se o *compile() e* o *finalize().*

O método *compile()* é responsável pela compilação e atualização das probabilidades na RB. Executar este método duas vezes não tem nenhum efeito, a não ser que seja removido algum nó da RB, neste caso a RB será recompilada (NORSYS, 2006).

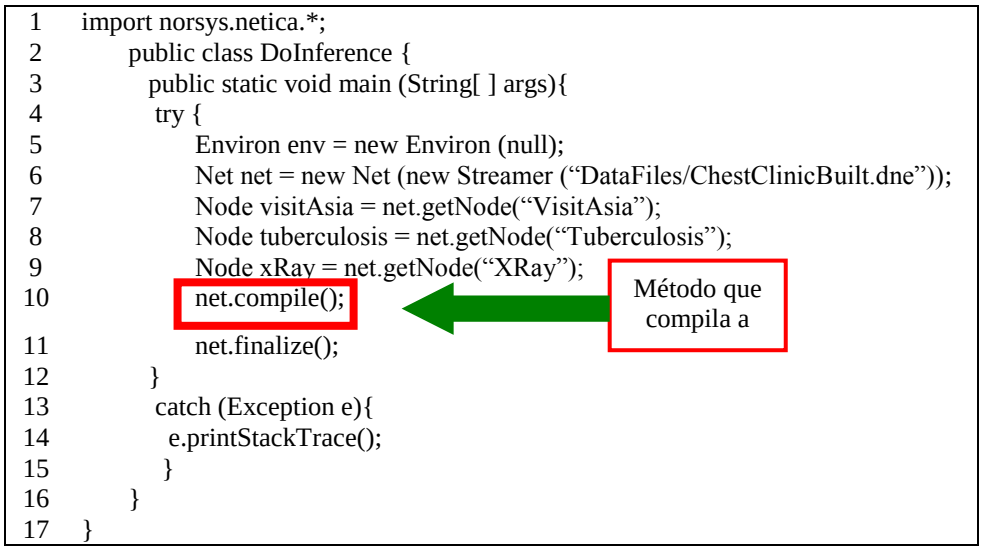

Figura 15 . Método Compile
Fonte: NORSYS (2006)

A Figura 15 ilustra um exemplo da chamada do método *compile()* que compila a RB a partir das suas probabilidade a *priori* definidas previamente.

Outro método disponibilizado nesta classe é o finalize que finaliza a RB e libera todos os recursos que ele utiliza (por exemplo, liberação da memória), incluindo também todas as suas subestruturas (por exemplo, os nós), este método ainda pode ser chamado diretamente para garantir a liberação dos recursos utilizados (NORSYS, 2006).

```
1   import norsys.netica.*;
2         public class DoInference {
3           public static void main (String[ ] args){
4            try {
5                 Environ env = new Environ (null);
6                 Net net = new Net (new Streamer ("DataFiles/ChestClinicBuilt.dne"));
7                 Node visitAsia = net.getNode("VisitAsia");
8                 Node tuberculosis = net.getNode("Tuberculosis");
9                 Node xRay = net.getNode("XRay");
10                net.compile();
11                net.finalize();
12           }
13           catch (Exception e){
14             e.printStackTrace();
15            }
16        }
17  }
```

Método que finaliza a RB

Figura 16 . Método Finalize() da classe Net
Fonte: NORSYS (2006)

Na Figura 16 o código em destaque ilustra o exemplo da chamada do método *finalize()*. Ainda segundo Norsys (2006), não é necessário a finalização da RB, mas é um bom hábito para acelerar o processo de liberação dos recursos que a RB utiliza no momento de execução.

4.1.3 Classe Node

A classe Node representa o nós da RB, sendo assim, esta classe é utilizada para representar as ações que podem ser realizadas como a parte que trata da RB.

A Figura 17 ilustra, a classe Node está no pacote *norsys.netica*, e extende a classe *General* desse mesmo pacote (NORSYS, 2006).

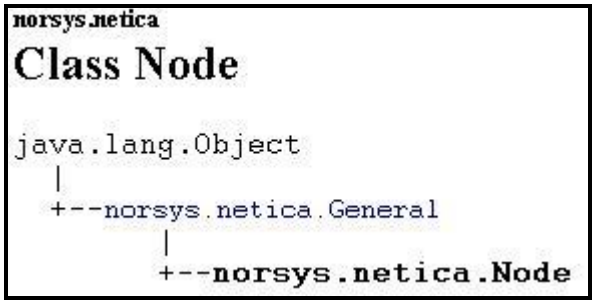

Figura 17 . Classe Node
Fonte: NORSYS (2006)

Entre os métodos existentes na classe *Node* destaca-se o *getBelief() e* o *enterFinding()*.

O método *getBelief()* é utilizado para retornar a probabilidade de cada estado de um determinado nó da RB.

Na Figura 18, a linha 1 ilustra que a variável *belief* do tipo double recebe a probabilidade do estado de *present* para o *nó tuberc*ulosis denominando-se a seguir na linha 2 esta mesma variável é apresentada ao usuário com o valor correspondente ao nó . *present.*

```
1    Double belief = tuberculosis.getBelief (“present”);
2    System.out.println(“\n The probability of tuberculosis is” + belief);
```

Figura 18 . Método GetBelief
Fonte: NORSYS (2006)

Outro método disponibilizado na classe Node é o *enterFinding()* responsável pela propagação das probabilidades condicionais, dado uma nova evidência (NORSYS, 2006).

A Figura 19 exemplifica a propagação considerando-se a evidência informada no método da evidência denominada *Cariogênica* do *nó dieta.* Considerando essa seleção, a inferência bayesiana é realizada para os demais *nós* da RB.

```
1    Dieta.enterFinding(“Cariogenica”);
```

Figura 19 . Método enterFinding
Fonte: NORSYS (2006)

4.1.4 Classe NodeList

A classe NodeList contém um vetor utilizado para armazenar apenas os nós a partir de uma única rede. Se outros objetos forem inseridos, ou nós com nomes diferentes e de redes diferentes, uma exceção será lançada quando o NodeList for passado para a Netica (NORSYS, 2006).

A Figura 20 ilustra que a classe NodeList está no pacote *norsys.netica*, que extende um vetor de lista, que por sua vez pertence a uma extensão de uma coleção abstrata.

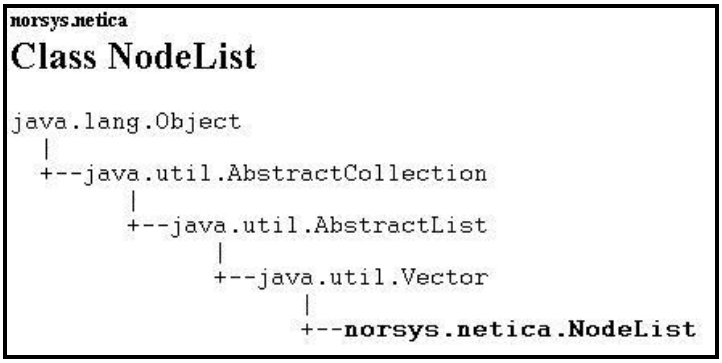

Figura 20 . Classe NodeList
Fonte: NORSYS (2006)

Entre os métodos existentes na classe *Node* destaca-se o *getNode(), que o* fabricante Norsys (2006) define que o método *getNode()* é o responsável pela associação do *nós* declarados na RB com os *nós* existentes lidos de um arquivo.

Na ilustração da Figura 21, a linha 1 mostra a associação do *nó* existente no arquivo *visitAsia* com o *nó* criado *VisitAsia*, sendo assim na linha 2 a associação acontece entre o *nó* existente no arquivo *tuberculosis* com o *nó* criado *Tuberculosis*.

```
1    Node visitiAsia = net.getNode ("VisitAsia");
2    Node tuberculosis = net.getNode ("Tuberculosis");
```

Figura 21 . Método getNode
Fonte: NORSYS (2006)

Segundo o fabricante Norsys (2006), estas classes e seus respectivos métodos proporcionam ao Netica Java API algumas características que auxiliam na sua implementação, como o manual completo de utilização da Netica Java API fornecido pelo fabricante, pois este possibilita ao desenvolvedor da integração um maior entendimento de cada uma das suas classes.

O fabricante ainda disponibiliza um guia completo de instalação da API e vários exemplos de aplicações, além de que o Netica Java API consegue executar todas as funções disponibilizadas no Netica C API.

Conforme apresentado neste capítulo, a seguir descreve-se algumas pesquisas que envolvem a utilização de redes bayesianas por meio da tecnologia Java.

5 TRABALHOS CORRELATOS

A área de incerteza tem abrangência em IA, e os SEP que apresentam sua BC representada por meio de Redes Bayesianas, por sua vez auxiliam na resolução e na distribuição de probabilidades para resolver determinado problema.

Os problemas que as RB se propõe a solucionar envolvem a representação da incerteza por aleatoriedade.

Neste aspecto estão sendo desenvolvidos muitos trabalhos no Brasil e em outros países que envolvem a modelagem da interface de uma BC por meio da linguagem Java, porém foram encontrados poucos trabalhos que utilizam a Netica Java API, foco principal desta pesquisa, mas foram encontradas várias pesquisas de desenvolvimento de RB por meio da tecnologia Java na área medica e em outras áreas afins .

5.1 EXPERT SYSTEM FOR MINE BURIAL PREDICTION

Esta pesquisa foi desenvolvido no laboratório de Física Aplicada do Curso de Ciência da Computação no ano de 2004 na Universidade Johns Hopkins, em Baltimore no EUA (BRANDT, RENNIE, 2004).

Seu objetivo foi desenvolver um sistema inteligente capaz de resolver e prever minas marítimas, sendo que, sua principal finalidade foi a de prever minas com o apoio da frota operacional da Marinha.

Para o desenvolvimento deste sistema foi utilizada a *shell* Netica para a construção da rede bayesiana, para a criação da interface gráfica foi utilizada a linguagem de programação Java com o auxilio da biblioteca de funções Netica Java API.

Conforme descrito pelos autores os resultados deste trabalho foram satisfatórios, pois o mesmo auxiliou na busca de minas marítimas.

5.2 CONSTRUÇÃO DE UMA REDE BAYESIANA APLICADA AO DIAGNÓSTICO DE DOENÇAS CARDÍACAS

Esta dissertação de mestrado do curso de Engenharia Mecatrônica e de Sistemas Mecânicos foi desenvolvido na Escola Politécnica da Universidade de São Paulo no ano de 2005 em São Paulo (SAHEKI, 2005).

O objetivo deste trabalho é a construção de uma ferramenta, de uso livre e gratuito, que possibilite a utilização da Rede Bayesiana desenvolvida neste trabalho de uma forma prática, o que não se encontrava em outras ferramentas para Redes Bayesianas livremente disponíveis.

Durante o desenvolvimento do sistema especialista, também foi um construído um software, livremente distribuído, que tem como objetivo a criação e manipulação de Redes Bayesianas. O software, denominado iBNetz, foi programado na linguagem de programação Java.

Segundo o autor o desempenho da rede obtido na última etapa da avaliação com casos de uso foi considerado adequado pelo especialista, ou seja, as probabilidades fornecidas pela rede correspondem às expectativas do especialista.

5.3 SISTEMA ESPECIALISTA ON-LINE DE AUXÍLIO AO DIAGNÓSTICO DE CÂNCER DE PRÓSTATA

Esta dissertação do curso de Pós Gradução em Engenharia Elétrica foi desenvolvido na Universidade Federal de Santa Catarina no ano de 2004 em Florianópolis – Santa Catarina (PEREIRA, 2004).

Seu objetivo volta-se ao desenvolvimento de um sistema especialista on-line na área urológica para o auxílio ao diagnóstico de câncer de próstata (SEDACaP).

O sistema SEDACaP foi desenvolvido com arquitetura em camadas, e de forma modular. Sua estrutura de componentes voltados ao usuário, como toda uma infra-estrutura de interfaces de interação com o usuário. Já a estrutura de suporte aos serviços é responsável pela segurança de transação de dados, suporte robusto a um alto número de requisições simultâneas e segurança da informação.

Para o desenvolvimento deste trabalho foram utilizados a plataforma Java por meio de JavaServer Pages juntamente com o banco de dados Sybase PowerDesigner 9.

Conforme descrição do autor o sistema apresentou bons resultados, mostrando um grande potencial de auxílio médico para o diagnóstico. Na Figura 22 tem-se uma das telas do sistema SEDACaP.

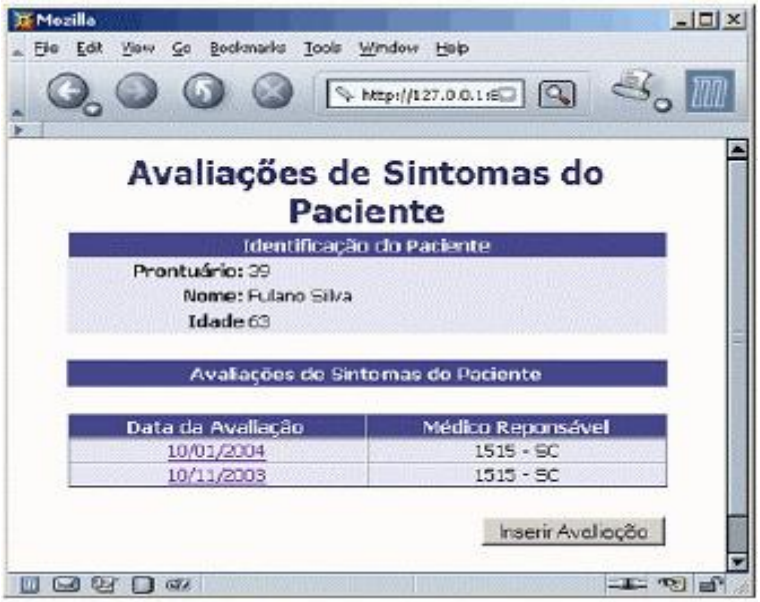

Figura 22 . Interface do SEDACaP
Fonte: PEREIRA (2004)

5.4 MONITORAÇÃO DO POTENCIAL DE RISCO DE INFECÇÃO HOSPITALAR EM UTI-NEONATAL

A presente pesquisa de iniciação científica refere-se a um artigo desenvolvido no Departamento do Programa de Pós Graduação em Ciência da Computação na Pontifícia Universidade Católica do Paraná no ano de 2005 em Curitiba no Paraná (ÁVILA et al, 2005).

Seu objetivo é fazer interagir, de forma dinâmica e inteligente, fontes de informações distribuídas que auxiliem as comissões de controle de infecções hospitalares na sua tarefa de evitar ou minimizar as infecções hospitalares.

Para o desenvolvimento deste trabalho foi utilizado o JADE, ferramenta de desenvolvimento em tecnologia Java. e a RB foi desenvolvida para calcular a predição do cálculo do potencial de risco. O sistema coletou as informações através dos simuladores dos sistemas hospitalares de forma distribuída e apresentou ao monitor da comissão de controle de infecções hospitalares os valores corretos calculados para o paciente em tempo real.

Segundo descrição dos autores o sistema alcançou o objetivo de prover uma ferramenta para monitorar em tempo real a possibilidade de infecções hospitalares em UTI-Neonatal.

5.5 SISTEMA ESPECIALISTA PROBABILÍSTICO PARA DEFINIÇÃO DE ESQUEMAS TÁTICOS

A presente pesquisa foi desenvolvida no Programa de Pós-Graduação em Engenharia e Gestão do Conhecimento na Universidade Federal de Santa Catarina no ano de 2005 em Florianópolis – Santa Catarina (BOGO et al, 2005).

O objetivo deste trabalho é o desenvolvimento de um sistema especialista probabilístico no domínio de uma partida de futebol para pode ajudar a comissão técnica na tomada da melhor decisão no momento mais apropriado.

Assim como aconteceria na maioria dos contextos, a implementação de um sistema especialista no Spirit que utiliza estas variáveis não é a tarefa mais difícil no processo. A tarefa mais complicada é a de mapear as variáveis corretas para poder atingir a resposta desejada na saída do sistema especialista.

O Spirit se mostrou uma ferramenta de fácil entendimento e facilitou a implementação do SEP, principalmente pela sua interface gráfica amigável e intuitiva. A implementação feita em na linguagem de programação Java também facilita o desenvolvimento por permitir a execução tanto em ambientes Windows como em ambientes Linux.

Conforme descrição dos autores os resultados obtidos foram satisfatório devido os testes realizados pelos especialistas, ou seja, as probabilidades fornecidas na rede correspondem às expectativas dos mesmos.

5.6 Ceres Sefs5: UM SISTEMA ESPECIALISTA PARA O CÁLCULO DA NECESSIDADE DE CALAGEM E RECOMENDAÇÃO DE CORRETIVO

Este trabalho foi desenvolvido no Curso de Ciência da Computação, na Universidade Federal de Lavras no ano de 2005 em Lavras – Minas Gerais (SOARES, SILVA, ZAMBALDE, 2004).

O objetivo deste trabalho é o desenvolvimento de um sistema especialista que realiza o cálculo da necessidade de calagem do solo para diferentes culturas, instaladas em diversos estados brasileiros.

O sistema foi desenvolvido na linguagem de programação Java podendo ser executado tanto pela *Internet* (como um *applet*) quanto fora dela (como uma aplicação), usando para modelagem a linguagem UML e banco de dados MySQL.

Conforme a descrição dos autores um sistema desse tipo torna-se cada vez mais importante e indispensável aos técnicos, de um modo geral, pelo fato de agilizar o processo de recomendação de corretivo, ter facilidade no uso e comodidade (pois pode ser acessado de qualquer lugar, desde que se tenha acesso à *Internet*).

5.7 UM SISTEMA ESPECIALISTA PROBABILÍSTICO PARA O APOIO A ANÁLISE DE PLANOS DE NEGÓCIOS DE EMPRESAS DE BASE TECNOLÓGICA

Esta pesquisa de iniciação científica refere-se a um artigo desenvolvido no Programa de Pós-Graduação em Engenharia e Gestão do Conhecimento na Universidade Federal de Santa Catarina no ano de 2005 em Florianópolis – Santa Catarina (JULIANI et al, 2005).

O artigo tem como objetivo realizar o desenvolvimento de um SEP na área empresarial como forma de apoiar a análise dos planos de negócio de empresas de tecnologia, por parte dos consultores especializados, estruturados para fim de ingresso em incubadoras de empresas.

O emprego dos sistemas especialistas probabilísticos, neste sentido corresponde a uma alternativa interessante para a minimização desta problemática, através da formalização do conhecimento tático e procedural, empregado pelos consultores especializados, na análise dos planos.

O SEP foi desenvolvido através da ferramenta *shell* Spirt e sua interface foi programada utilizando-se a linguagem Java, possuído portabilidade de plataforma em diversos sistemas operacionais.

Segundo as considerações dos autores os resultados obtidos com este SEP foram válidos no sentido em que os especialistas aprovaram a alternativa de minimização de problemas gerados no seu ambiente de trabalho.

5.8 UM MÓDULO PARA FUSÃO DE DADOS UTILIZANDO REDES BAYESIANAS

Esta pesquisa de graduação do curso de Ciência da Computação do Departamento de Ciência da Computação foi desenvolvida na Universidade Católica de Goiás no ano de 2004 em Goiânia – Goiás (JOSÉ, 2004).

O objetivo desta pesquisa consiste na construção de um módulo capaz de executar a Fusão de Dados utilizando como técnica principal as Redes Bayesianas. Como aplicação do trabalho propõe-se a automatização do Interrogatório Sintomatológico, parte componente da anamnese, um procedimento médico.

Para o desenvolvimento deste trabalho foi utilizada a ferramenta Microsoft Belief Networks para o desenvolvimento da RB e para a implementação da interface foi utilizada a tecnologia Java.

Conforme o autor foi possível a realização do objetivo como conseqüência natural da assimilação da natureza do problema que se esta modelando, sendo que o mesmo procura somente simular o pensamento probabilístico que o médico realiza durante a execução do Interrogatório Sintomatológico ao paciente. A Figura 23 abaixo mostra a tela de localização do problema no paciente.

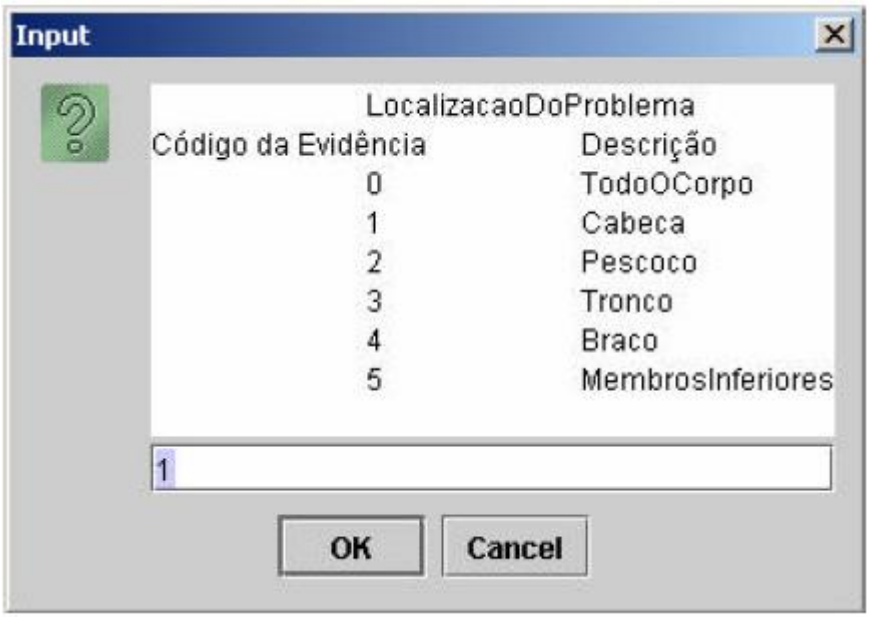

Figura 23 . Tela de Localização do Problema

Fonte: JOSE (2004)

Com a realização desta pesquisa observou-se que a área de inteligência artificial na saúde por meio de RB desenvolvidas com a tecnologia Java é um campo que cresce a cada dia com novas pesquisas e descobertas em todo o mundo.

6 MÉTODO DE INTEGRAÇÃO DE UMA BASE DE CONHECIMENTO DE UM SISTEMA ESPECIALISTA PROBABILÍSTICO UTILIZANDO A NETICA JAVA API

Com o passar dos anos torna-se cada vez mais necessário que sistemas computacionais sejam desenvolvidos em ambientes e linguagens de programação para que possam ser executadas tanto em desktop, como na Web.

Esta pesquisa consiste na integração da *shell* Netica por meio da Netica Java API que auxilia a construção de Redes Bayesianas, com o ambiente de programação Java que por meio de *applets* permite que o sistema seja utilizado em várias plataformas e ser acessado pela internet.

Para atingir os objetivos propostos algumas etapas metodológicas foram seguidas e, dentre estas, a primeira indica o levantamento bibliográfico realizado por meio de livros, artigos, trabalhos de conclusão de curso, teses, dissertações, entre outros materiais científicos na área médica.

Ainda nesta etapa foram encontrados alguns trabalhos correlatos, que se baseiam em SEP desenvolvidos por meio da tecnologia Java (BRANDT, RENNIE, 2004).

A seguir são apresentadas as demais etapas de metodologia: definição da base de conhecimento do estudo de caso; atualização das informações do sistema do estudo de caso; estudo, integração e documentação de uma RB por meio da Netica Java API, realização de testes e os resultados obtidos.

6.1 DEFINIÇÃO DA BASE DE CONHECIMENTO DO ESTUDO DE CASO

A problemática escolhida para o estudo de caso diz respeito a uma RB já desenvolvida na Universidade do Extremo Sul Catarinense em um trabalho de conclusão de curso denominado Sistema Especialista para o Apoio ao Diagnóstico de Doenças Exantemáticas Maculopapulosas com Erupção Obrigatória (ANTUNES, 2002), (ANTUNES et al, 2004).

Esta RB foi desenvolvida em 2002 pelo acadêmico Luciano Antunes do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, vinculado aos grupos de pesquisa em Inteligência Computacional Aplicada, e Informática Médica e Telemedicina tendo como especialista pelo médico e professor do curso de Medicina da UNESC Frank Traebert Júnior.

A escolha da utilização dessa RB se deu em virtude da necessidade de atualização deste sistema, e considerando também sua utilidade para os acadêmicos do curso de Medicina da Universidade do Extremo Sul Catarinense no Laboratório de apoio ao ensino médico.

É importante ressaltar também conforme apresentado na justificativa desta pesquisa que muitos sistemas probabilísticos costumam ser pouco utilizados na prática em decorrência de tecnologia utilizada no desenvolvimento, tornando este sistema apenas uma fonte de referência com uma BC desatualizada.

Assim, busca-se com a utilização da RB do SEDDEM oferecer uma atualização de interface e probabilidades condicionais conforme necessidade do especialista e usuários a fim de que este sistema possa ser utilizado com maior freqüência no curso de Medicina da Universidade do Extremo Sul Catarinense e de outra instituições.

As probabilidades a *priori* das hipóteses diagnósticas e das evidências são apresentadas nas tabelas 8 e 9.

Tabela 8 . Hipóteses diagnosticadas da RB SEDDEM
Fonte: ANTUNES (2002)

Evidências	***Hipóteses Diagnosticadas***
Sarampo	25 %
Rubéola	25 %
Escarlatina	20 %
Eritema Infeccioso	15 %
Exantema Subito	15 %

Tabela 9 . Evidências da RB SEDDEM
Fonte: ANTUNES (2002)

Evidências		**Hipóteses Diagnosticas**				
		P(ek/ Sarampo) 25%	***P(ek/ Rubéola) 25%***	***P(ek/ Escarlatina) 20%***	***P(ek/ Eritema Infeccioso) 15%***	***P(ek/ Exantema Súbito) 15%***
Prodromos	Intenso	75	1	70	1	5
	Moderado	20	9	20	4	25
	Inexistente	5	90	10	95	70
Febre	Alta acima de 39	60	1	70	1	70
	Moderado de 38 a 39	30	9	20	1	20
	Baixa 37,5 a 38	8	60	9	8	9
	Inexistente abaixo 37,5	2	30	1	90	1
Idade	Ate 6 meses	1	1	1	1	15
	De 6 a 24 meses	10	4	4	5	80
	De 2 a 5 anos	40	15	10	47	4
	De 5 a 15 anos	49	80	85	47	1
Exantema	Maculopapular generalizado	70	3	3	1	3
	Maculopapular discreto	20	25	1	1	80
	Maculopapular discreto segme	5	70	1	1	15
	Eritema puntiforme	3	1	90	7	1
	Eritema vermelho	2	1	7	90	1
Face	Sem Característica	5	97	8	1	97
	Esbofeteada	1	1	1	97	1
	Sarampenta	93	1	1	1	1
	Eritema frontal palidez perio	1	1	90	1	1
Duração Exantema	De 1 a 2_dias	2	20	1	2	80
	De 3 a 5 dias	18	75	19	8	15
	De 6 a 10 dias	80	5	80	90	
Estado Geral	Comprometido	75	1	95	5	1
	Conservado	25	99	5	95	99
Epidemiologia	Contagioso	99	99	80	5	5
	Pouco Contagioso	1	1	20	95	95
Pantagnomônicos	Koplik	70	1	1	1	1
	Exantema facial borboleta	1	1	1	97	1
	Sinal Pastia Filatow e frambo	1	1	97	1	1
	Não apresenta	28	97	1	1	97
Imunidade	Imune	99	99	1	1	1
	Não imune	1	1	99	99	99
Fase Pos Exantema	Descamação leve	90	1	9	1	1
	Descamação intensa	8	1	90	1	1
	Sem descamação	2	98	1	98	98
Linfadeanopatia	Discreto	80	1	80	95	90
	Intenso	2	99	20	5	10

Com a BC já definida é necessários estabelecer as atualizações e executá-las por meio da integração da Netica Java API com o ambiente de programação NetBeans IDE 5.5.

6.2 ATUALIZAÇÃO DAS INFORMAÇÕES DOS SISTEMAS DE ESTUDO DE CASO

O SEDDEM escolhido para a realização da integração foi desenvolvido inicialmente no ambiente de programação C++ Builder[12], oferecendo uma interface desktop com as opções de inferência bayesiana e contando com um módulo de ajuda em texto livre.

Na atualização da interface realizou-se alguns ajustes de diagramação conforme ilustra a Figura 24, que foi desenvolvida no ambiente de programação NetBeans IDE 5.5.1, disponibilizando-se as informações deste sistema por meio de uma applet[13].

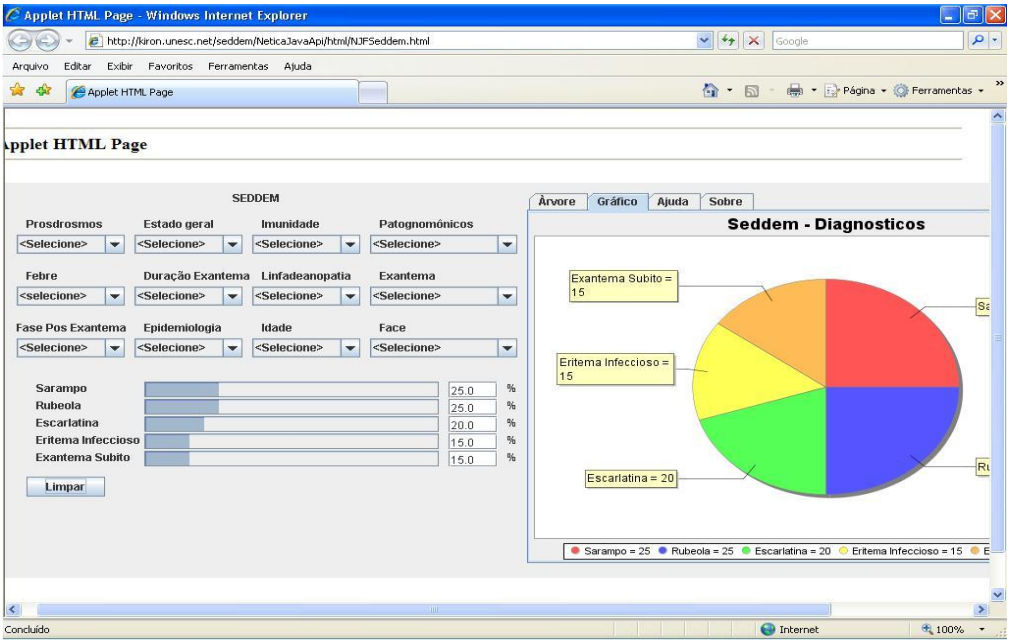

Figura 24 . Applet SEP SEDDEM.

[12] Disponível em: http://www.borland.com/
[13] Um programa escrito em Java que possui uma página html (http://java.sun.com/applets/).

Este novo projeto de interface incluiu duas novas informações que não constavam na versão anterior, sendo o módulo de explicação desenvolvido por meio de árvores (mostrando cada opção que o usuário seleciona), sendo esta um componente essencial na arquitetura de um SEP e um gráfico em modelo de pizza para uma melhor visualização do usuário em relação as hipóteses diagnosticadas geradas pelo SEP.

Para o desenvolvimento do módulo de explicação utilizou-se o estudo e padronização realizado por Goulart (2007) que trata de um Trabalho de Conclusão de Curso do curso de Ciência da Computação da UNESC, que teve como objetivo implementar um agente assistente no sistema de apoio ao ensino e diagnóstico etiológico de lombalgia. Nesta pesquisa desenvolveu-se uma forma de representação do raciocínio por meio de árvores.

Conforme a opinião do especialista as probabilidades condicionais não necessitariam ser atualizadas durante essa pesquisa. No próximo semestre com uma utilização deste sistema por meio da applet, o especialista concederá aos acadêmicos de Medicina a tarefa de atualizar as probabilidades da BC conforme novos casos diagnosticados. Com as probabilidades atualizadas os acadêmicos poderão auxiliar a validação deste SEP.

O módulo de ajuda foi reestruturado por meio de *links* html, facilitando a sua utilização com mais aplicabilidade na busca de ajuda que o usuário realiza.

Depois de realizar a atualização de interface com o usuário e inclusão do módulo de explicação passou-se à etapa da utilização da Netica Java API para a comunicação do ambiente NetBeans 5.5 com a RB desenvolvida na s*hell* Netica.

6.3 ESTUDO, INTEGRAÇÃO E DOCUMENTAÇÃO DO SEDDEM POR MEIO DA NETICA JAVA API

A integração da RB do SEDDEM foi realizada entre a *shell* Netica e o ambiente de programação NetBeans IDE 5.5, por meio da Netica Java API descrita anteriormente.

Nesta etapa documentou-se as principais classes e métodos utilizados, incluindo inclusive um tutorial com os procedimentos necessários para a instalação da Netica Java API no ambiente Net NetBeans IDE 5.5, na plataforma Windows (Apêndice A),

6.3.1 Inicialização do ambiente Netica no SEDDEM

O ambiente para a utilização da RB do SEDDEM é referenciado conforme ilustra a Figura 25 que detalha a inicialização do ambiente Netica, onde: na linha 1 é declarado que a variável *env* é do tipo *Environ* e recebe *null*, logo depois na linha 4 esta variável *env* recebe uma referência ao novo ambiente com parâmetro *null* que representa não necessitar da licença.

A linha 3 apresenta um tratamento de exceção é obrigatório para a lógica, pois segundo o fabricante Norsys deve-se prever que o engenheiro do conhecimento não possa criar um ambiente nulo sem execução.

```
1   Environ env = null;
2   {
3     try {
4           env =new Environ (null);
5     } catch (Exception e)
6          e.printStackTrace( );
7      }
8   }
```

Figura 25 . Criação do ambiente do SEP SEDDEM.

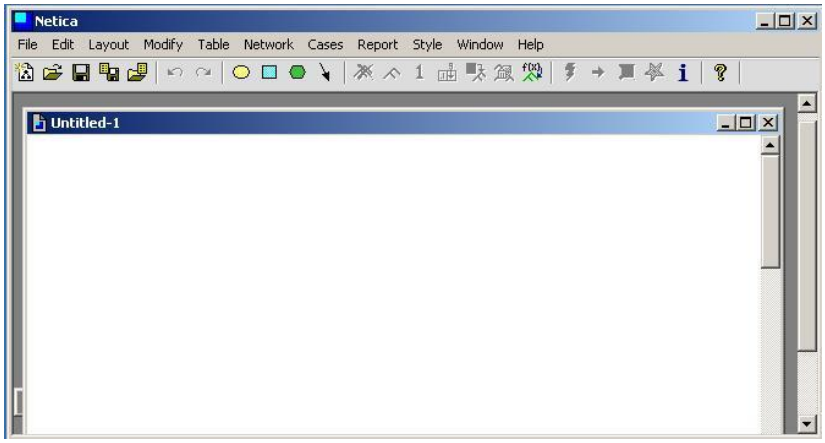

Figura 26 . Criação do Ambiente na *shell* Netica

A Figura 26 ilustra a forma comparativa de criação de ambiente na *shell* Netica para um melhor entendimento da lógica de inicialização.

6.3.2 Utilização da RB do SEDDEM via API

```
1     public void Seddem(){
2         try {
3             Net net = null;
4             try{
5                 InputStream is = this.getClass().getResourceAsStream("/Base.dne");
6                 net = new Net(new Streamer(this.getClass().getResourceAsStream("/Base.dne"), "Seddem", env));
7             }catch(Exception e){
8                  e.printStackTrace();
9             }
10            doencas   =  net.getNode("DOENCAS");
11            prodromos  =  net.getNode("PRODROMOS");
12            estadoGeral  =  net.getNode("ESTADO_GERAL");
13            net.compile();
14            acao()
15            System.out.println("\nexecutou o seddem" );
16            net.finalize();   // not strictly necessary, but a good habit
17        }catch (Exception e) {
18            e.printStackTrace();
19        }
20    }
```

Figura 27 Construção da RB do SEP SEDDEM

A Figura 27 ilustra o código que apresenta a utilização da RB, onde na linha 3 foi declarada uma variável do tipo Net com o nome net. Declarando-se a variável, na linha 5 foi realizada a abertura do arquivo por um construtor da classe Net, onde na linha 6, a RB é aberta recebendo os parâmetros *Base.dne*(arquivo da RB), um nome a

RB, neste caso SEDDEM, e o seu endereço de ambiente criado por meio da classe Environ, neste caso env.

O tratamento de exceção é indicado na construção da RB e na abertura do arquivo para que não seja possível declarar uma rede e um arquivo nulo ou não existente, conforme ilustra a linha 4.

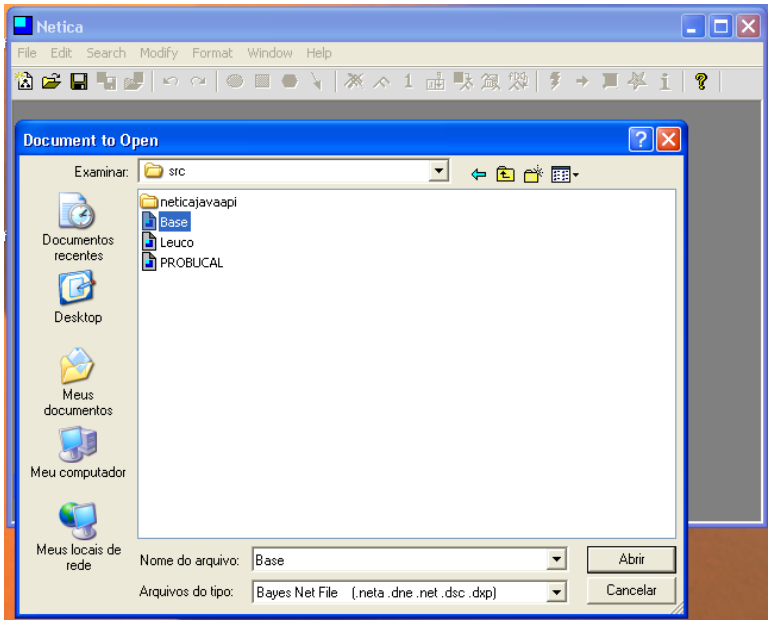

Figura 28 . Construção de uma RB na *shell* Netica

A Figura 28 ilustra a forma comparativa de abrir uma RB na *shell* Netica, onde basta acessar o menu *File* seguido de *Open* e localizar o arquivo desejado.

Para realizar a inferência bayesiana é necessária a criação ou associação dos *nós* já existentes. Na integração do SEDDEM realizou-se a associação dos *nós* existentes com os *nós* endereçados na integração, sendo realizada por meio do método GetNode que associa o *nó* endereçado com o *nó* do arquivo.

```
1    public void Seddem(){
2        try {
3            Net net = null;
4            try{
5                InputStream is = this.getClass().getResourceAsStream("/Base.dne");
6                net = new Net(new Streamer(this.getClass().getResourceAsStream("/Base.dne"), "Seddem", env));
7            }catch(Exception e){
8                 e.printStackTrace();
9            }
10           doencas  = net.getNode("DOENCAS");
11           prodromos  = net.getNode("PRODROMOS");
12           estadoGeral  = net.getNode("ESTADO_GERAL");
13           net.compile();
14           acao()
15           System.out.println("\nexecutou o seddem" );
16           net.finalize();   // not strictly necessary, but a good habit
17       }catch (Exception e) {
18            e.printStackTrace();
19       }
20   }
```

Associação dos nós da RB

Figura 29 . Associação dos nós da RB SEDDEM.

A Figura 29 ilustra a associação dos *nós* da RB onde na linha 10 a variável *doencas* (*nó* endereçado na integração) recebe a referência do *nó DOENCAS* existente fisicamente na RB do arquivo que foi aberto. O comando getNode associa os valores de probabilidade a *priori* que estão nos *nós* do arquivo da RB repassando-os para os *nós* endereçados na integração.

Com os *nós* e as respectivas probabilidades a *priori* já associada as variáveis, à próxima etapa da integração visa compilar a RB.

```
1    public void Seddem(){
2        try {
3            Net net = null;
4            try{
5                InputStream is = this.getClass().getResourceAsStream("/Base.dne");
6                net = new Net(new Streamer(this.getClass().getResourceAsStream("/Base.dne"), "Seddem", env));
7            }catch(Exception e){
8                 e.printStackTrace();
9            }
10           doencas  = net.getNode("DOENCAS");
11           prodromos  = net.getNode("PRODROMOS");
12           estadoGeral  = net.getNode("ESTADO_GERAL");
13           net.compile();
14           acao()
15           System.out.println("\nexecutou o seddem" );
16           net.finalize();   // not strictly necessary, but a good habit
17       }catch (Exception e) {
18            e.printStackTrace();
19       }
20   }
```

Compilação da RB

Figura 30 . Compilação da RB SEDDEM.

O método *compile()* da classe Net ilustrado em destaque na Figura 30 na linha 13, realiza a propagação das probabilidades a *priori*.

A lógica que pode ser associada a *shell* Netica é a opção compile network.

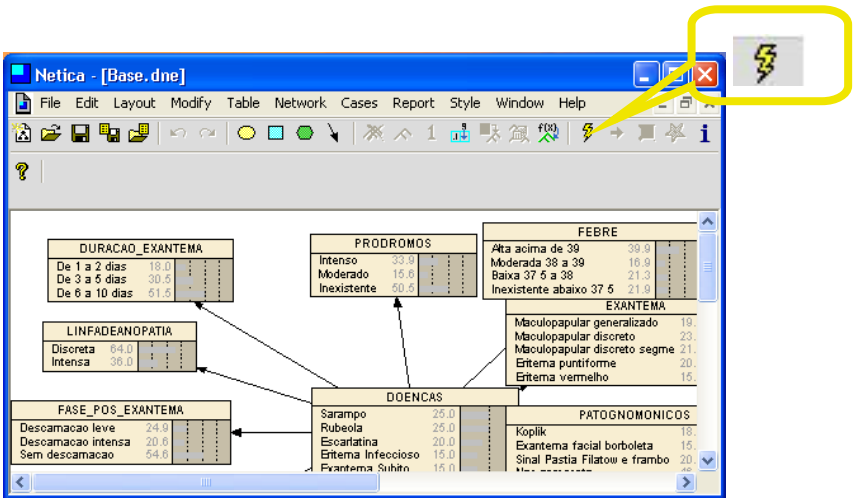

Figura 31 . Compilação de uma RB na *shell* Netica

Para obter os valores resultantes da compilação, inferência em um estado de um *nó* utiliza-se o método getBelief da classe Node.

```
1  public void acao(){
2      try{
3          belief = doencas.getBelief("Sarampo");
4          belief = belief * 100;
5          jTFSarampo.setText(String.valueOf(round2(belief, 3)));
6          System.out.println("\n A probabilidade de ter sarampo eh" + belief);
7          jPBSarampo.setMinimum(0);
8          jPBSarampo.setMaximum(100);
9          jPBSarampo.setValue(round(belief, 0));
10     }catch(NeticaException net){
11         net.printStackTrace();
12     }
13         popularArvore();
14         jTabbedPane1.setComponentAt(1, createPizzaPanel());
15 }
```

Figura 32 **.** Cálculo da Inferência do nó DOENCAS da RB SEDDEM.

Na Figura 32 a parte em destaque ilustra na linha 3 que a variável *belief* (variável global e do tipo *double*) recebe do *nó doencas* a probabilidade da evidência *Sarampo.*

.Figura 33 . Método getBelief no ambiente *shell* Netica

A Figura 33 ilustra qual valor o método getBelief irá mostrar ao usuário, nesta caso 25 % de chance de o paciente possuir sarampo.

A propagação das probabilidades condicionais são realizadas a cada evidência escolhida pelo usuário. Essa lógica é realizada por meio do método *enterFinding* que propaga os valores na inferência conforme a evidência escolhida.

```
1   int opcao =  jCBEpidemiologia.getSelectedIndex();
2     try{
3           switch (opcao){
4   case 1:
5           epidemiologia.enterFinding("Contagioso");
6           break;
7   case 2:
8           epidemiologia.enterFinding("Pouco_contagioso");
9           break;
10        }
11     }catch(NeticaException ne){
12          ne.printStackTrace();
13     }
14      acao();
```

Figura 34 .Método de propagação de evidência *enterFinding* no *nó Epidemiologia*

A figura 34 ilustra na linha 5 o valor contagioso propagado do *nó* denominado *epidemiologia* propagado na RB.

O fabricante Norsys afirma ser obrigatório o tratamento de exceção neste método, a fim de evitar passar valores nulos nas evidências, como ilustrado na linha 2.

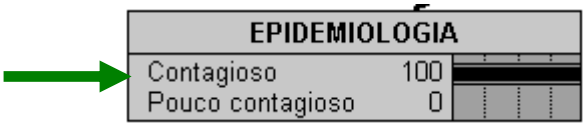

Figura 35 . Método enterFinding no ambiente *shell* Netica

A Figura 35 ilustra o mesmo procedimento na *shell* Netica, ou seja, ao selecionar a opção *contagioso* do nó *epidemiologia* se propaga esta evidência para toda a RB.

A finalização da RB é realizada por meio do método *finalize* da classe Net onde são liberados todos os recursos de alocação de memória utilizados durante o processo de execução do sistema. Na figura 36 é destacado o trecho de código na linha 16, onde é utilizado o método *finalize.*

```
1    public void Seddem(){
2        try {
3            Net net = null;
4            try{
5                InputStream is = this.getClass().getResourceAsStream("/Base.dne");
6                net = new Net(new Streamer(this.getClass().getResourceAsStream("/Base.dne"), "Seddem", env));
7            }catch(Exception e){
8                 e.printStackTrace();
9            }
10           doencas   =  net.getNode("DOENCAS");
11           prodromos  =  net.getNode("PRODROMOS");
12           estadoGeral  =  net.getNode("ESTADO_GERAL");
13           net.compile();
14           acao()
15           System.out.println("\nexecutou o seddem" );
16           net.finalize();   // not strictly necessary, but a good habit
17       }catch (Exception e) {
18           e.printStackTrace();
19       }
20   }
```

Figura 36 . Finalização da RB SEDDEM**.**

Depois de realizar a integração a próxima etapa da metodologia descreve os testes realizados.

6.4 REALIZAR TESTES

Conforme a opinião do especialista, a nova versão do SEDDEM será melhor utilizada em suas aulas no curso de medicina da UNESC.

Sendo que o software desenvolvido se comportou conforme a expectativa do especialista.

6.5- RESULTADOS OBTIDOS

Como resultado desta pesquisa foram oferecidas novas versões para os sistemas SEDDEM, PROBUCAL e BIOWOMAN com portabilidade de execução na internet.

Para atender esta característica foram desenvolvidas algumas applets.

No SEDDEM, a applet foi desenvolvida utilizando sua interface sem grandes alterações. Sua implementação é apresentada na Figura 37.

```
1  import neticajavaapi.visao.NFJSeddem;
2  public class NJFSeddemApplet extends javax.swing.JApplet{
3      public void init (){
4          try{
5              NJFSeddem f = new NJFSeddem();
6              getContentPane(). add(f.getContentPane());
7              f.Seddem();
8              }catch (Exception e) {
9                      e.printStackTrace();
10                 }
11     }
12 }
```

Figura 37 Código de implementação da applet do SEP SEDDEM

Esta applet encontra-se no site do grupo de pesquisa em Informática Médica e Telemedicina denominado Kiron (http://www.kiron.unesc.net), que tem como objetivo por meio do desenvolvimento científico-tecnológico, desenvolver projetos de pesquisa nos quais são utilizados elementos dos domínios da Medicina e da Computação para suporte aos profissionais de saúde no atendimento ao público, além de criar um ambiente de pesquisa em Informática Médica na região sul do estado de Santa Catarina para gerar e difundir tecnologias aplicadas à área médica.

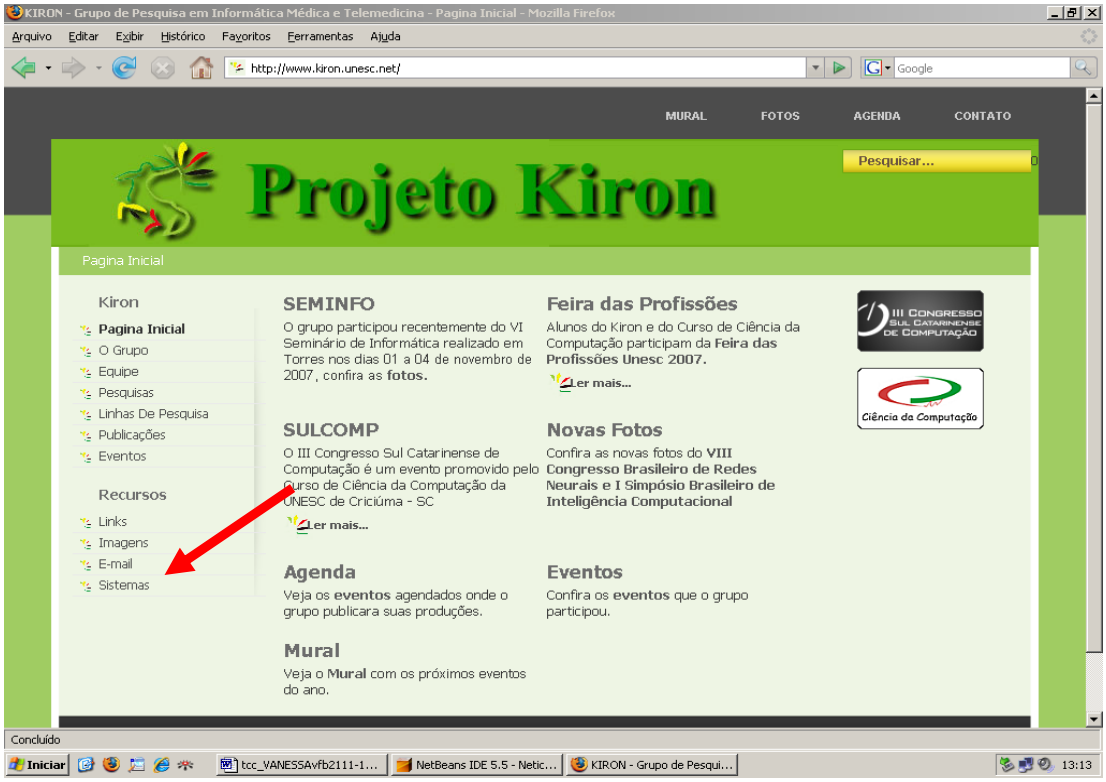

Figura 38 . Página do Grupo de Pesquisa Kiron – Acesso ao link

A Figura 38 ilustra a home-page do Kiron, que apresenta link *Sistemas* para o acesso da applet do SEDDEM.

Ao clicar neste link a página oferecerá a opção de download onde um arquivo instalador (copia as dll Netica.dll e NeticaJ.dll para o caminho especificado) deverá ser executado no computador em uso no seguinte caminho (C:\WINDOWS\system32), é necessário salvar o arquivo neste diretório citado para o correto funcionamento do SEP. Este procedimento deverá ser realizado somente uma vez em cada computador, pois a *shell* Netica possui uma limitação com o ambiente de programação NetBeans IDE 5.5

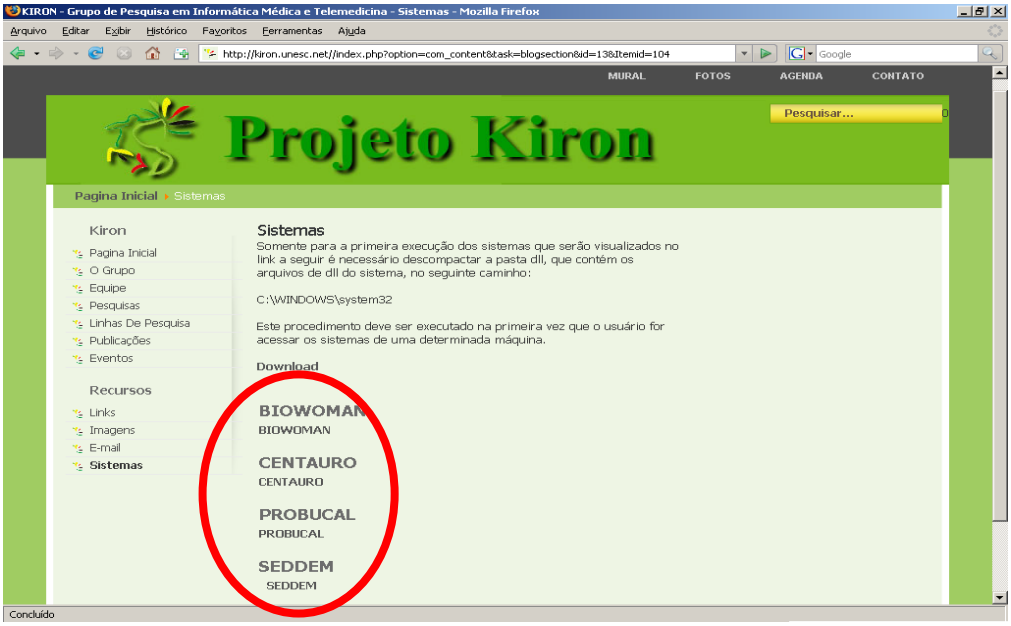

Figura 39. Site do Grupo Kiron com o menu dos SEP desenvolvidos

A Figura 39 ilustra a página do site do Kiron que apresenta hiperlinks dos SEP desenvolvidos nesta pesquisa. Ao selecionar um SEP para ser executado, a applet terá seu start .

A Figura 40 ilustra o momento que a applet é executada e pede para que o usuário aceite a assinatura com a política de segurança da applet, indicando que a applet a ser executada não é mal intencionada. Esta aceitação permite que a applet seje executada na máquina do usuário.

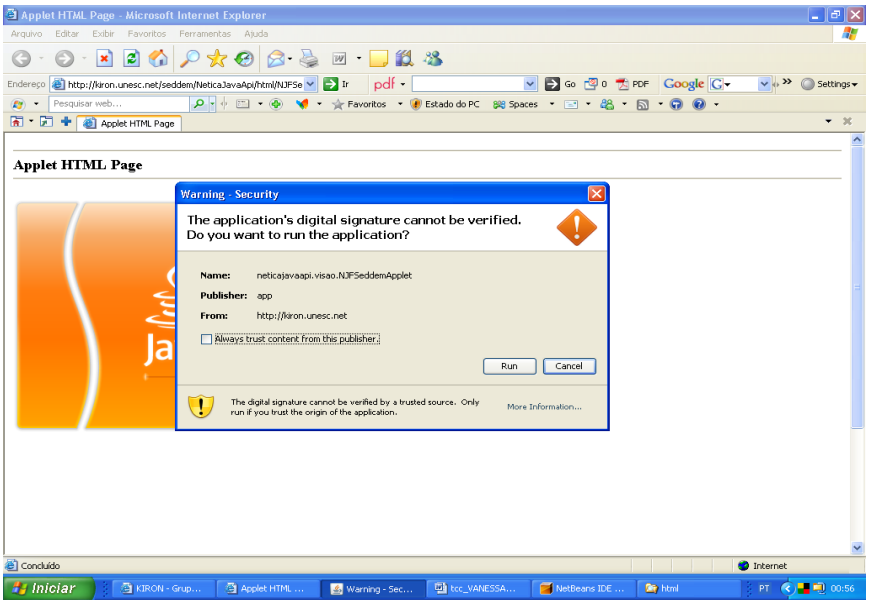

Figura 40. Página do Grupo de Pesquisa Kiron – Aceitação de execução da Applet

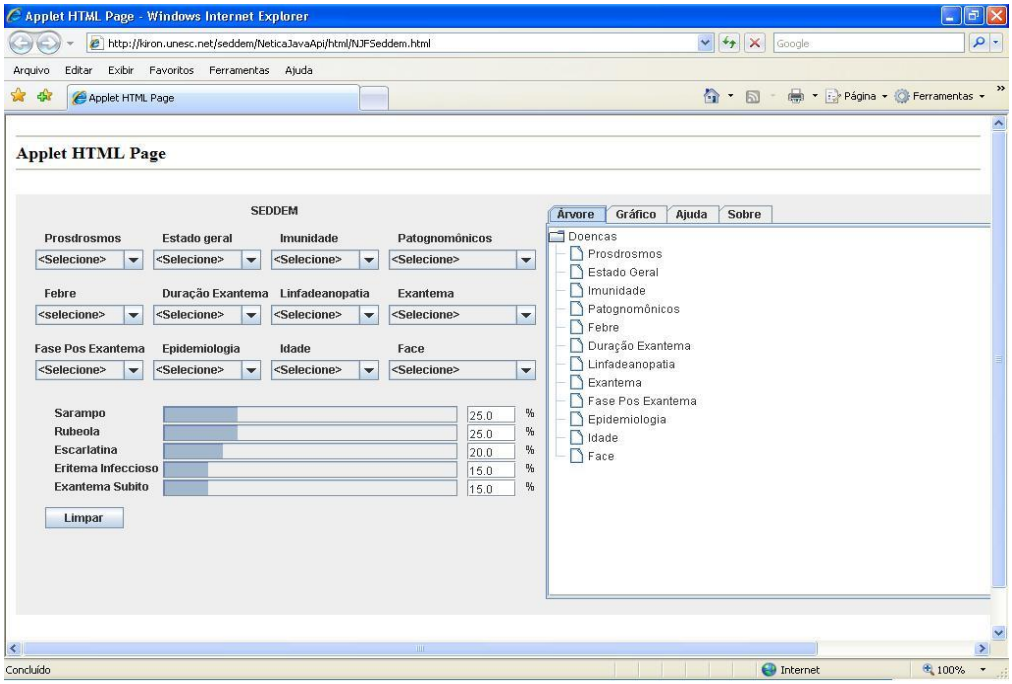

Figura 41 . Interface inicial da applet do SEP SEDDEM

A Figura 41 é descreve a interface inicial da applet do SEDDEM onde todos os componentes são inicializados, ou seja, os métodos net(streamer), getNode, compile são executados, podendo-se realizar consultas.

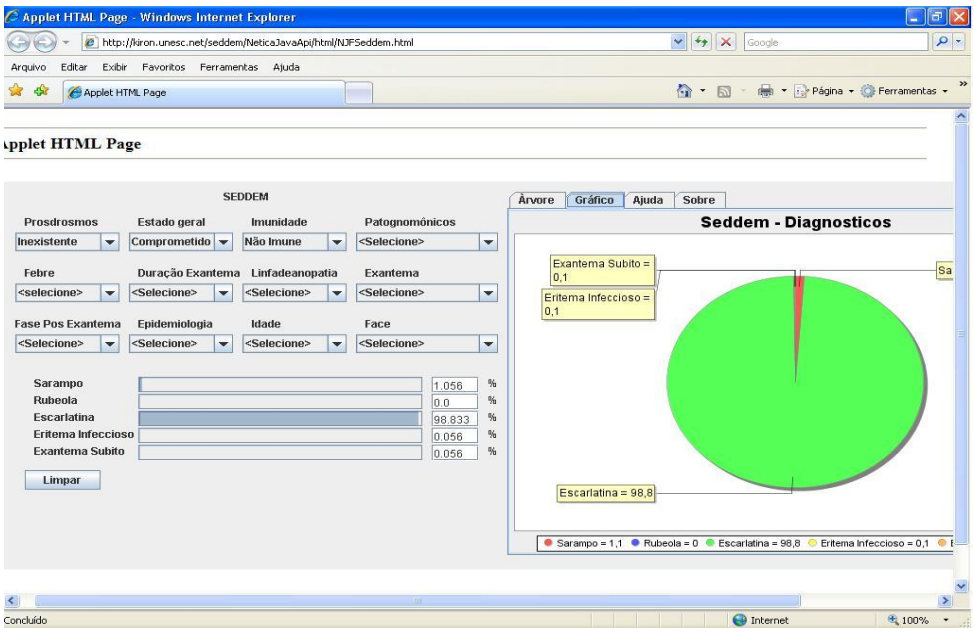

Figura 42 . Interface inicial da applet do SEP SEDDEM com visualização do gráfico

A Figura 42 apresenta o gráfico inserido nesta versão, que permite ao usuário um melhor entendimento dos diagnósticos estabelecidos por meio das inferências.

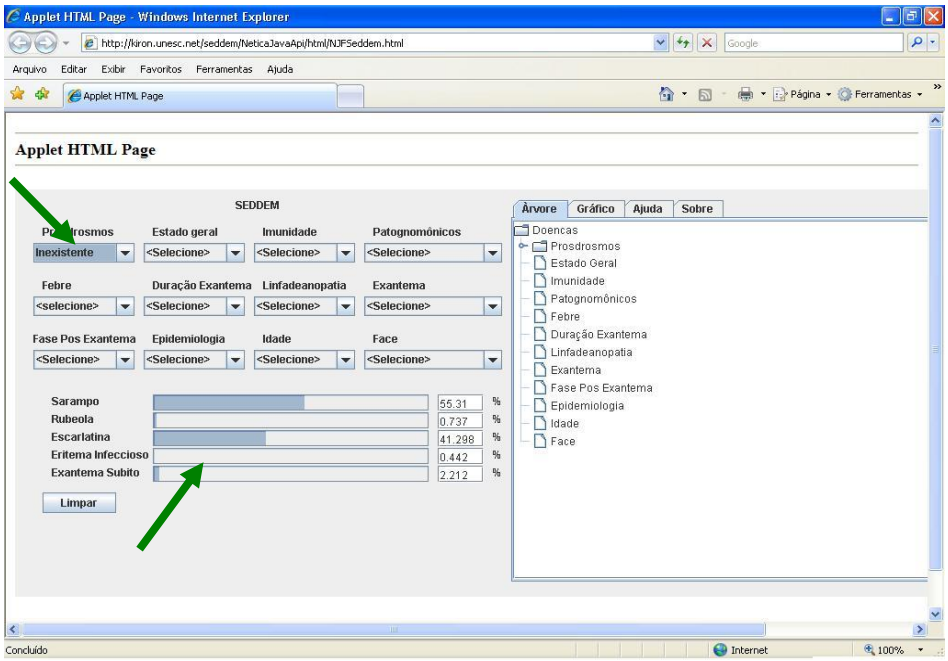

Figura 43 . *Applet* do SEP SEDDEM com apenas 1 evidência selecionada

A Figura 43 ilustra apenas a seleção da evidência *inexistente* para o *nó Prodromos podendo-se* visualizar que as suas probabilidades a *priori* foram alteradas conforme inferência realizada.

A Figura 44 ilustra a applet com três evidências selecionadas, sendo que a cada nova evidência as probabilidades são propagadas.

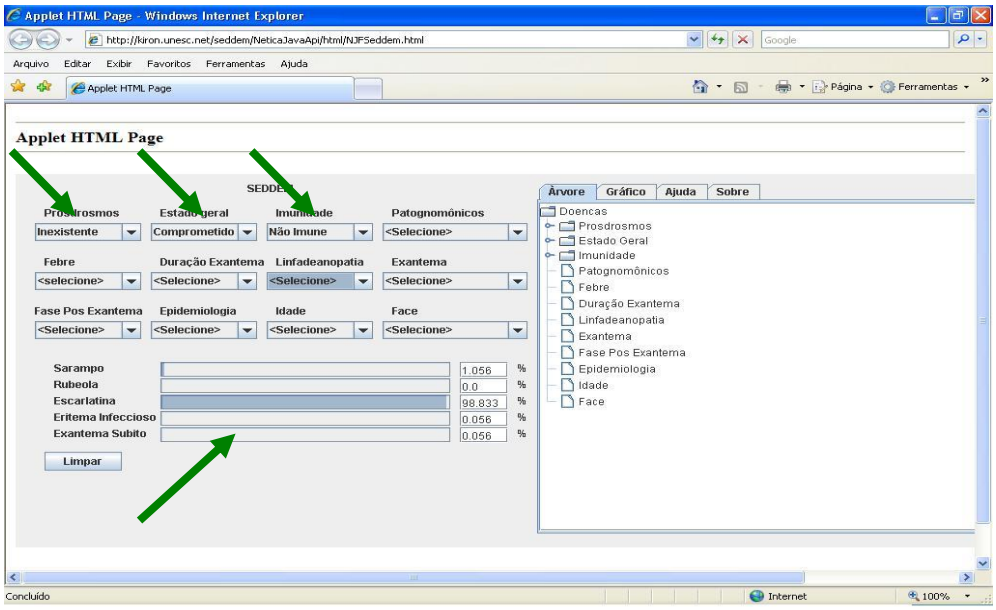

Figura 44 . *Applet* do SEP SEDDEM com três evidências propagadas

Na Figura 45 é possível visualizar o módulo de explicação que foi atualizado conforme a seleção das três evidências, passando agora a possuir três *subnós* sendo um com a informaçãode cada *nó* selecionado.

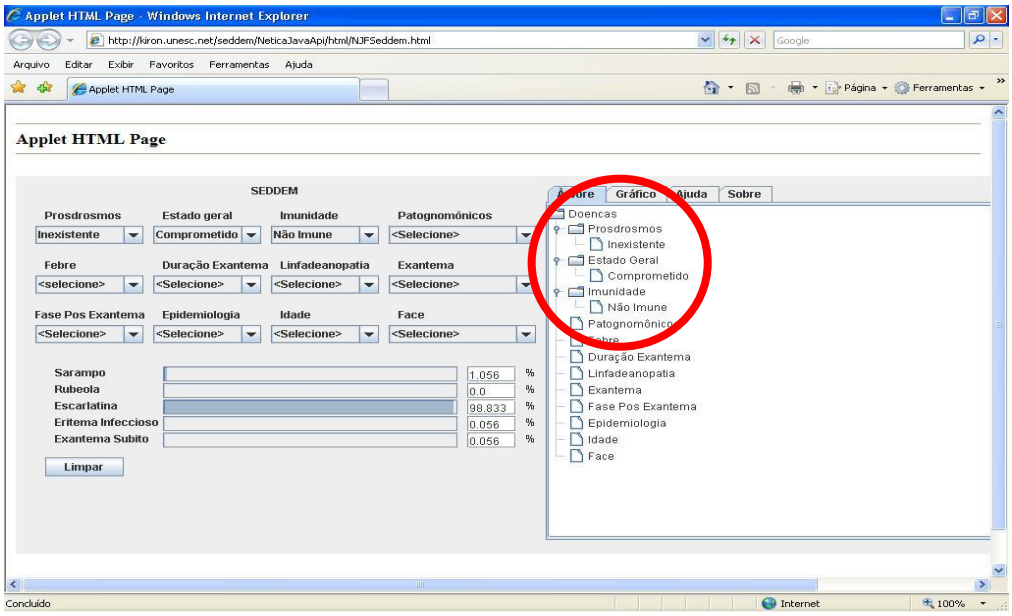

Figura 45 . *Applet* do SEP SEDDEM com três evidencias propagadas e visualizadas no módulo de explicação

Com esta mesma propagação de evidências é possível visualizar na Figura 46 o gráfico de pizza que apresenta a proporção das probabilidades condicionais das hipóteses diagnosticas.

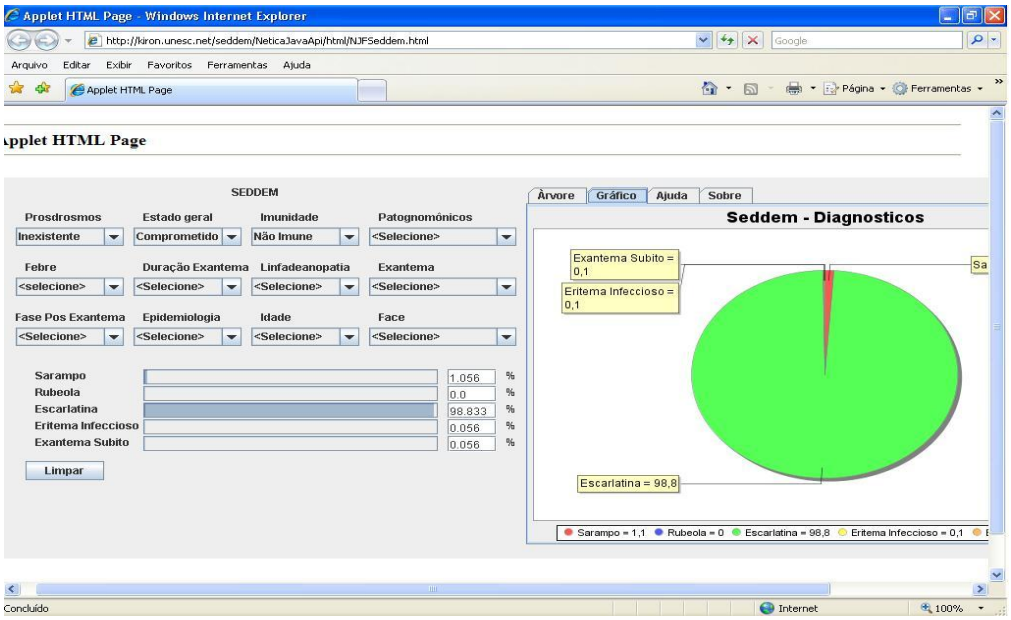

Figura 46. *Applet* do SEP SEDDEM com três evidencias propagadas e visualizadas no gráfico

6.5.1. OUTROS SISTEMAS ESPECIALISTAS PROBABILISTICOS INTEGRADOS PELA NETICA JAVA API

Além do SEDDEM também foi possível realizar a integração de mais dois SEP da área da saúde já desenvolvidos na UNESC.

Estes sistemas foram desenvolvidos por meio dos mesmos critérios e padrões de implementação adotados no SEDDEM.

Realizou-se a integração do PROBUCAL e BIOWMAN descritos detalhadamente nos itens a seguir.

6.5.1.1 PROBUCAL

O PROBUCAL é um sistema especialista probabilístico, desenvolvido para prognosticar algumas doenças bucais como: cárie, gengivite, periodontite e câncer, além de verificar a possibilidade da inexistência das mesmas (ROSA, 2002), (ROSA et al, 2003).

Sua RB foi desenvolvida em 2002 pela cadêmica Lílian Regina Junkes da Rosa como trabalho de conclusão do curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, vinculado aos grupos de pesquisa em Inteligência Computacional Aplicada, e Informática Médica e Telemedicina tendo como especialista a odontóloga Maria Emília Moreira Wessler Philippi.

As probabilidades a *priori* das hipóteses diagnósticas e das evidências são apresentadas nas tabelas 10 e 11.

Tabela 10– Hipóteses diagnosticadas da RB PROBUCAL

Fonte: ROSA (2002)

Evidências	***Hipóteses Diagnosticadas***
Carie	35 %
Gengivite	30 %
Periodontite	20 %
Câncer	6 %
Nenhuma	9 %

Tabela 11 – Evidências da RB PROBUCAL
Fonte: ROSA (2002)

Evidências		**Hipóteses Diagnosticas**				
		P(ek/ Carie) 35%	***P(ek/ Gengivite) 30%***	***P(ek/ Periodontite) 20%***	***P(ek/ Cancer) 6%***	***P(ek/ Nenhuma) 9%***
Acidez Saliva	PH Acido	65	60	58	65	40
	PH Basico	35	40	42	35	60
Faixa Etária	Zero Cinco	14	10	3	7	22
	Cinco Doze	26	15	7	8	21
	Doze Vinte	24	18	10	12	17
	Vinte Trinta e Cinco	13	19	16	15	14
	Trinta e Cinco Cinqüenta	11	21	22	18	13
	Cinqüenta Setenta	7	10	23	19	8
	Acima Setenta	5	7	19	21	5
Fumo	Sim	62	65	65	70	40
	Nao	38	35	35	30	60
Alcool	Sim	60	60	60	65	45
	Nao	40	40	40	35	55
Fluoretação Agua	Sim	35	36	36	33	52
	Nao	65	64	64	67	48
AIDS	Sim	58	60	60	65	40
	Nao	42	40	40	35	60
Sexo	Masculino	48	45	47	55	48
	Feminino	52	55	53	45	52
Raca	Branca	52	45	45	60	46
	Negra	48	55	55	40	54
Hereditariedade	Sim	57	58	58	60	48
	Nao	43	42	42	40	52
Posição Dente	Correta	40	35	35	40	60
	Incorreta	60	65	65	60	40
Dieta	Cariogenica	73	65	65	63	40
	Não Cariogenica	27	35	35	37	60
Diabetes	Sim	62	65	65	65	40
	Não	38	35	35	35	60
Higienização	Adequada	35	35	38	36	68
	Inadequada	65	65	62	64	32
Protese	Boa Adaptacao	40	40	35	35	60
	Ma Adaptacao	60	60	65	65	40

O PROBUCAL foi desenvolvido com os recursos de inferência bayesiana, módulo de ajuda interativo, módulo de explicação desenvolvido por meio de árvores (mostrando cada opção que o usuário seleciona), sendo esta um componente essencial na arquitetura de um SEP. O sistema também oferece um gráfico de pizza para uma melhor visualização do usuário em relação as hipóteses diagnosticadas geradas pelo SEP.

Este sistema encontra-se disponível na página do Grupo de Pesquisa Kiron, com o seguinte link para acesso:http://www.kiron.unesc.net/probucal.

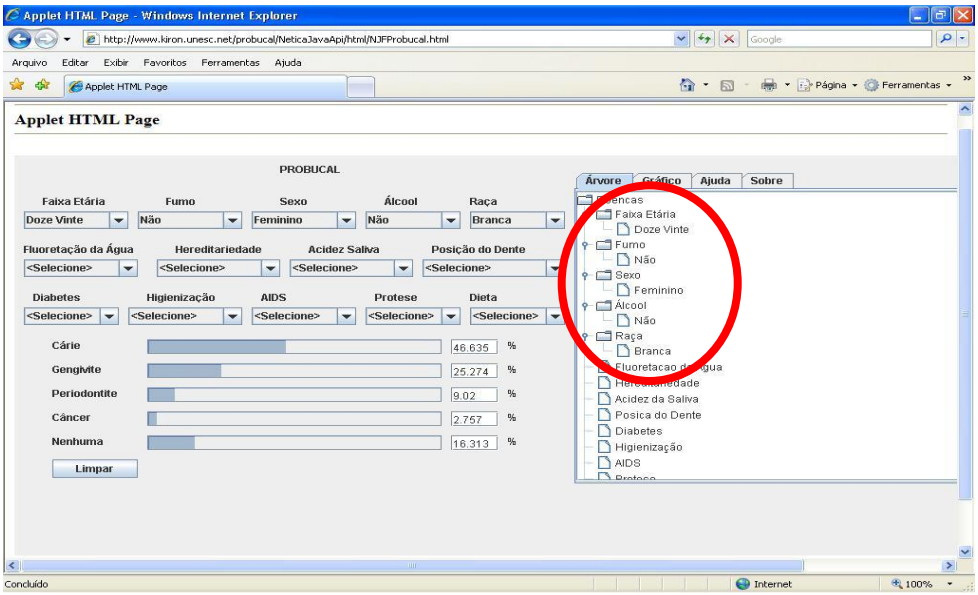

Figura 47 . *Applet* do SEP PROBUCAL com algumas evidências propagadas

A Figura 47 ilustra a applet PROBUCAL com as seguintes evidências propagadas: *Doze Cinco* para o *nó Faixa Etária*; *Não* para o *nó Fumo*; *Feminino* para o *nó Sexo*; *Não* para o *nó Álcool* e *Branca* para o *nó* correspondente a *Raça*, na figura é possível visualizar o módulo de explicação atualizado.

6.5.1.2 BIOWOMAN

O SEP BIOWOMAN que objetiva auxiliar o profissional médico no diagnóstico de algumas doenças relacionadas à leucorréia, encontradas no trato genital inferior da população feminina (RODRIGUES, 2002), (MACHADO, SIMÕES, MATTOS, 2006).

Esta RB foi desenvolvida em 2002 pela acadêmica Ana Carolina Braga Rodrigues como trabalho de conclusão do curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, vinculado aos grupos de pesquisa em

Inteligência Computacional Aplicada, e Informática Médica e Telemedicina tendo como especialista a médica e professora do curso de Medicina da UNESC Sandra Manenti.

As probabilidades a *priori* das hipóteses diagnósticas e das evidências são apresentadas nas tabelas 12 e 13.

Tabela 12 . Hipóteses diagnosticadas da RB BIOWOMAN
Fonte: RODRIGUES (2002)

Evidências	***Hipóteses Diagnosticadas***
Flora Vaginal Normal	30 %
Flora Mista	12 %
Aumento de Lactobacilo	10 %
Cocus	3 %
Inflamacao Descamativa	1,5 %
Candidiase	20 %
Gardnerela	15 %
Trichomonas	5 %
Gonococo	2 %
Clamidia	1,5 %

Tabela 13 – Evidências da RB BIOWOMAN
Fonte: RODRIGUES (2002

Evidências		***Hipóteses Diagnosticas***									
		P(ek/Flora Vginal Normal) 30%	***P(ek/ Flore Mista) 12 %***	***P(ek/ Aumento Lactobacilo) 10 %***	***P(ek/ Cocus) 3%***	***P(ek/ Inflamação Descamativa) 1,5%***	***P(ek/Candidíase) 20 %***	***P(ek/Gardnerela) 15%***	***P(ek/Trichomonas) 5%***	***P(ek/Gonococo) 2%***	***P(ek/Clamidia) 1,5%***
Ardência	Presente	50	50	70	60	60	80	50	70	60	50
	Ausente	50	50	30	40	40	20	50	30	40	50
Exame a Fresco	Cel epiteliais pouco lacto	68,5	9	29	4,5	9,5	19	24	9,5	9,5	24
	Lacto Grande Qtde	0,5	9,5	59	0,5	0,5	9	0,75	0,5	0,5	0,75
	Hifas pseudoHifas esporos	6	0,5	9	0,5	0,5	69	0,75	0,5	0,5	24
	Clue cells	6	0,5	0,5	0,5	0,5	0,5	24	0,5	0,5	0,75
	Trichomonas presente	0,5	0,5	0,5	0,5	0,5	0,5	0,75	69	0,5	0,75
	Diplococo Intracelular	0,5	0,5	0,5	0,5	0,5	0,5	0,75	0,5	69	0,75
	Cocus	6	0,5	0,5	69	9,5	0,5	24	9,5	9,5	24
	Cocus e bacilos	6	59,5	0,5	19,5	9,5	0,5	24	9,5	9,5	24
	Leucocitos	6	19,5	0,5	4,5	69	0,5	24	9,5	9,5	24
Uso Antib	Sim	50	50	50	50	50	70	50	50	50	50
	Não	50	50	50	50	50	30	50	50	50	50
Gravidez	Sim	60	50	60	50	50	80	50	50	50	50
	Nao	40	50	40	50	50	20	50	50	50	50
Imunossupressao	Sim	50	50	50	50	50	80	50	50	50	50
	Não	50	50	50	50	50	20	50	50	50	50
Suspeita DST	Sim	50	50	50	50	50	50	50	99,5	9,5	99,5
	Não	50	50	50	50	50	50	50	0,5	0,5	0,5
Prurido	Presente	1	5	70	1	1	80	60	75	5	5
	Ausente	99	95	30	99	99	20	40	25	95	95
Odor Teste KOH	Positivo	50	50	50	60	50	50	80	40	50	50
	Negativo	50	50	50	40	50	50	20	60	50	50
Disperiunia	Ausente	99	40	20	15	15	20	35,5	30,5	5,5	5,5
	Inicial	0,5	58,5	79,5	15	15	79,5	25	40	15	15
	Profunda	0,5	1,5	0,5	70	70	0,5	39,5	29,5	79,5	79,5
Rel Cicli Mens	Fase Luteas	20	33	70	33	33	80	15	30	33	33
	Fase Ovulatoria	60	33	15	33	33	10	15	30	33	33
	Fase Folicular	20	34	15	34	34	10	70	40	34	34
PH Vaginal	Menor que 3,8	0,5	0,5	5	0,5	0,5	35	0,5	0,5	0,5	0,5
	Entre 3,8 e 4,5	98	89	89,5	0,5	5	59,5	5	0,5	0,5	0,5
	Entre 4,5 e 6,8	1	10	5	49,5	49,5	98,5	9,5	98,5	98,5	98,5
	Entre 6,8 e 8,5	0,5	0,5	0,5	49,5	49,5	0,5	45	0,5	89,5	0,5
Idade	Nascimento Menarca	50	10	10	15	5	20	10	5	10	10
	Menarca Mwenopausa	30	80	80	15	70	60	70	85	70	70
	Pos Menopausa	20	10	10	70	25	20	20	10	20	20
Odor	Ausente	59,5	40	59,5	59,5	59,5	40	10	10	40	59,5
	Discreto	40	59,5	40	40	40	59,5	20	60	59,5	40
	Fetido	0,5	0,5	0,5	0,5	0,5	0,5	70	30	0,5	0,5
Cor	Incolor	58	29	0,5	0,5	39	0,5	0,5	0,5	0,5	0,5
	Muco Turvo	0,5	2,5	19	2	2	19	0,5	0,5	0,5	8,5
	Branco Grumoso	0,5	15	0,5	25	25	0,5	0,5	19	49	10
	Branco Leitoso	38,5	50,5	5,5	60	60	10	19	9	20	10
	Branco Acinzentado	0,5	20	44	1,5	1,5	29,5	0,5	0,5	0,5	0,5
	Amarelo	0,5	5,5	0,5	0,5	0,5	0,5	0,5	59,5	0,5	0,5
	Amarelo Purulento	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	14	68,5
	Amarelo Esverdeado	0,5	5	0 ,5	7,5	7,5	0,5	29	10	14	0,5
	Amarelo Esverdeado Bolhoso	0,5	0,5	0,5	2,5	2,5	0,5	49	0,5	0,5	1

O SEP BIOWOMAN foi implementado com os recursos de inferência bayesiana, módulo de ajuda interativo, módulo de explicação desenvolvido por meio de árvores (mostrando cada opção que o usuário seleciona), sendo esta um componente essencial na arquitetura de um SEP. O sistema também oferece um gráfico de pizza para uma melhor visualização do usuário em relação as hipóteses diagnosticadas geradas pelo SEP.

Este sistema se encontra disponível na página do Grupo de Pesquisa Kiron, com o seguinte link para acesso: http://www.kiron.unesc.net/biowoman.

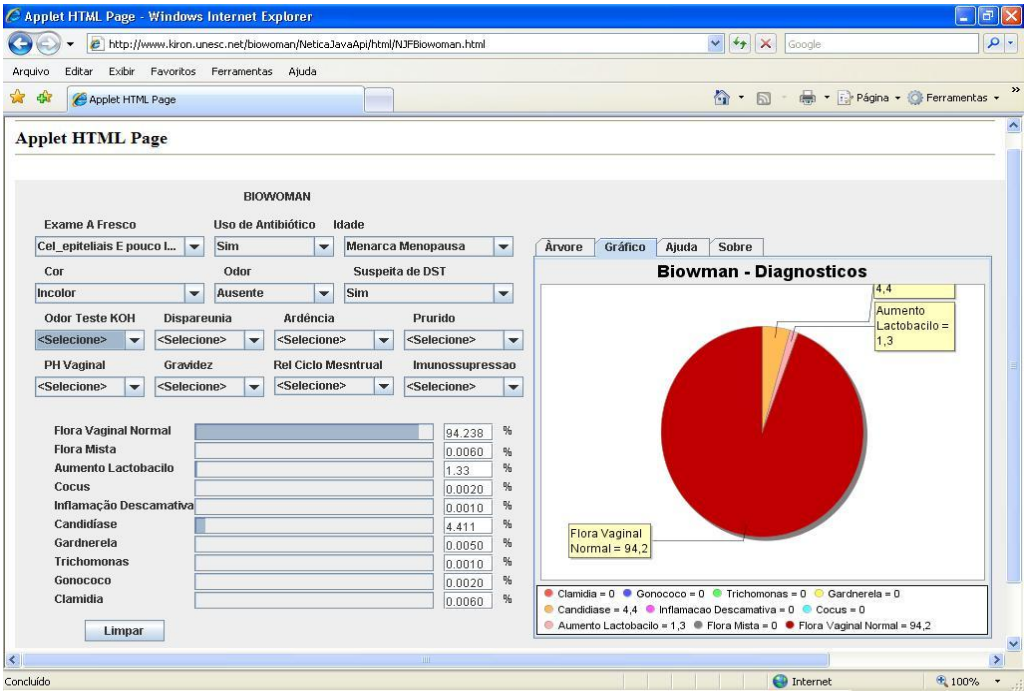

Figura 48 . *Applet* da interface do Biowoman em execução

A Figura 48 ilustra a applet do BIOWOMAN com as seguintes evidências propagadas: *Cel_epiteliais_e_pouco* para o *nó Exame A Fresco*; *Sim* para o nó *Uso de Antibióticos*; *Monarca Menopausa* para o *nó* que corresponde a *Idade*; *Incolor* para o *nó Cor*; *Ausente* para o *nó Odor* e *Sim* para o *nó* correspondente a *Suspeita de DST*, na figura é possível visualizar o gráfico atualizado conforme o diagnóstico de doenças.

Sendo estes os resultado obtidos pode afirmar que a integração entre a Netica Java API e o ambiente de programação NetBeans IDE 5.5 foi de grande relevância para a conclusão desta pesquisa.

7. CONCLUSÃO

Os sistemas desenvolvidos para a utilização na inteligência computacional médica são em grande parte destinados a apoiar aos profissionais de saúde no decorrer normal de suas atividades, auxiliando-os em tarefas que se baseiam na manipulação de dados e de conhecimentos.

Com a realização desta pesquisa observou-se que a área de inteligência artificial na saúde é um campo que cresce a cada dia com novas pesquisas e descobertas em todo o mundo.

As ferramentas de gerenciamento para a construção de RB auxiliam na implementação de aplicações inteligentes para diminuir a complexidade dos cálculos computacionais, e assim a *shell* Netica contribui com os métodos de gerenciamento e inferência na RB.

Desta forma pode-se concluir que os objetivos foram alcançados, compreendendo-se o processo de desenvolvimento de Sistemas Especialistas Probabilísticos.

O objetivo principal desta pesquisa consequentemente foi concluído pela integração da Netica Java API com o ambiente de programação NetBeans IDE 5.5, considerando-se que durante este processo foram compreendidos os principais recursos disponibilizados pela API da s*hell* Netica para o ambiente Java, que resultaram nas novas versões dos sistemas SEDDEM, PROBUCAL e BIOWOMAN.

Com o intuito de facilitar futuras pesquisas na área de conhecimento referente ao Netica Java API ofereceu-se uma documentação sobre a utilização desta API no ambiente de desenvolvimento NetBeans IDE 5.5.

No estudo de caso realizado, para cada aplicativo, foram atualizados alguns módulos clássicos de sua arquitetura, entre estes destacando-se a interface com o usuário oferecida via applet e acessada pelo site do Grupo de Pesquida em Informática Médica e Telemedicina da UNESC, além da inclusão do módulo de explicação em cada sistema.

Teve-se como limitação da pesquisa, que applet pode ser executada de forma satisfatória no navegador Internet Explorer em virtude de algumas configurações de segurança do demais navegadores. Ainda nesse contexto, para o correto funcionamento neste browser é necessário realizar o download dos arquivos NeticaJ.dll e Netica.dll necessários para a execução da API da *shell* Netica nas applets.

Como trabalhos futuros recomenda-se a integração da Netica Java API com outros ambientes de programação Java, como por exemplo o Eclipse, JSP; a realização de um estudo de eficiência entre os algoritmos implementados pela Netica C API e a Netica Java API; realizar a avaliação e validação do SEDDEM, PROBUCAL e BIOWOMAN; integrar os demais SEP já desenvolvidos na UNESC a fim de expandir e socializar suas informações com a comunidade acadêmica.

REFERÊNCIAS

ANTUNES, Luciano. ***SEDDEM* - Sistema Especialista para o Apoio ao Diagnóstico de Doenças Exantemáticas Maculopapulosas com Erupção Obrigatória.** 2002. 76 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2002.

ANTUNES, Luciano et al. **Apoio ao Diagnóstico de Doenças Exantemáticas Maculopapulosas por meio do Raciocínio Probabilístico**. In: CONGRESSO BRASILEIRO DE INFORMÁTICA EM SAÚDE, 9., 2004, Criciúma. **Anais eletrônicos...** Criciúma: UNESC, 2004. Disponível em: http://www.sbis.org.br/cbis9/arquivos/845.pdf Acesso: 20 out. 2007.

ÁVILA, Bráulio Coelho et al. **Monitoração do potencial de Riscos de Infecção Hospitalar em UTI – Neonatal.** Departamento de Ciência da computação – Pontifícia Católica do Paraná, Curitiba, Paraná, 2005.

AZEVEDO, Thiago dos Anjos. **SEATTRC - Um Sistema Especialista de Apoio a Decisão dos Tipos de Trabalhadores e Rescisões de Contratos.** 2005. 67 f.Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Universidade Federal da Bahia, Salvador, Bahia, 2005.

BARRETO, Jorge Muniz. **Inteligência artificial no limiar do século XXXI.** 3.ed. Florianópolis: Duplic, 2001.

BITTENCOURT, Guilherme. **Inteligência artificial:** ferramentas e teorias. Florianópolis: UFSC, 1998.

BOGO, Luis H. et al. **Sistema Especialista Probabilístico para Definição de Esquemas Tático.** Programa de Pós-Graduação em Engenharia e Gestão do Conhecimento - Universidade Federal de Santa Catarina, 2005, Florianópolis.

BONGIOLO, Silvia Biff . **SARA- Sistema Inteligente de Auxílio à Reparação na Injeção Eletrônica Automotiva.** 2002. 74 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2002.

BRANDT, Alan; RENNIE, Sarah. **Expert System For Mine Burial Prediction** Johns Hopkins University Applied Physics Laboratory, 2004, Johns Hopkins.

CAMPOS, Mario Massa de; SAITO, Kaku. **Sistemas inteligentes em controle e automação de processos.** Rio de Janeiro: Ciência Moderna, 2004.
COSTA, Ernesto; SIMÕES, Anabela. **Inteligência Artificial:** fundamentos e aplicações. Rio de janeiro: FCA, 2004.

COSTA, Ernesto; SIMÕES, Anabela. **Inteligência artificial:** fundamentos e aplicações. Lisboa: FCA, 2004.

COZMAN, Fabio Gagliardi. **JavaBayes.** Disponível em: http://www.cs.cmu.edu/afs/cs/user/fgcozman/www/home.html. Acesso: 20 out. 2006

FERREIRA, Andrei Luis et al. **Utilização de Rede Bayesiana Para Auxílio de Tomada de Decisão na Escolha de Clientela de Games.** Programa de Pós-Graduação em Ciências da Computação – Universidade Federal de Santa Catarina. 2003, Florianópolis.

FLORES, Cecília Dias; PEROTTO, Filipo; VICCARI, Rosa Maria. **Sistemas Baseados em Conhecimento para a Área da Saúde.** Programa de Pós Graduação em Computação – Universidade Federal do Rio Grande do Sul. 2003, Porto Alegre.

GOULARTE, Fábio Bif. ***SÉLO* - Sistema Especialista Probabilístico de Apoio ao Ensino do Diagnóstico Etiológico de Lombalgia.** 2002. 74 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2002.

GOULARTE, Jackson Pirola. **Agente Assistente no Sistema de Apoio ao Ensino e Diagnóstico Etiológico de Lombalgia.** 2007. 100f. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2007.

GUINZANI, Jhonas Bonfante et al. **Mineração De Dados Em Redes Bayesianas Utilizando a API Da Shell Belief Network Power Constructor (BNPC).** In: CONGRESSO SUL CATARINENSE DE COMPUTAÇÃO, 2., 2006, Criciúma. **Anais eletrônicos...** Criciúma: UNESC, 2006. Disponível em: www.dcc.unesc.net/sulcomp. Acesso: 20 out. 2006.

HUDSON, Donna L.; COHEN, Maurice E. **Neural Networks and Artifical Intelligence for Biomedical Enginnering.** New York: IEEE, 1999.

HUGIN EXPERT. Disponível em: http://www.hugin.com/. Acesso: 07 dez. 2006.

JENSEN, Finn V.. **Bayesian networks and decision graphs.** New York: Springer, 2001.

JOSÉ, Alexandre Bellezi. **Um módulo para Fusão de Dados Utilizando Redes Bayesianas**. 2004. 76 f. Trabalho de Conclusão de Curso. (Graduação em Ciência da Computação) - Universidade Católica de Goiás, Goiânia, Goiás, 2004.

JULIANI, Jordan P. et al. **Sistema Especialista Probabilístico para o Apoio a Análise de Planos de Negócios de Empresas de Base Tecnológica.** Programa de Pós-Graduação em Engenharia e Gestão do Conhecimento - Universidade Federal de Santa Catarina, 2005, Florianópolis.

KOEHLER, Cristiane. **Modelagem de Redes Bayesianas a partir da Identificação de Padrões em Base de Dados.** 2002. 44f. Tese de Doutorado (Programa de Pós Graduação em Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

LADEIRA, Marcelo; FLORES, Cecília Dias; VICCARI, Rosa Maria. **Tratamento Eficiente da Incerteza em Sistemas de Apoio à Decisão.** Departamento de Ciência da computação – Universidade de Brasília. 2002, Brasília.

LADEIRA, Marcelo; VICCARI, Rosa Maria; COELHO, Helder. **Redes Bayesianas Multiagentes.** Departamento de Ciência da computação – Universidade de Brasília. 2004, Brasília.

LADEIRA, Marcelo et al. **Ferramenta Aberta e Independente de Plataforma para Redes probabilísticas.** Departamento de Ciência da computação – Universidade de Brasília. 2003, Brasília.

LUGER, George F. **Inteligência artificial :** estruturas e estratégias para a resolução de problemas complexos. 4. ed Porto Alegre: Bookman, 2004.

LUNA, José Eduardo Ochoa. **Algoritmo em para Aprendizagem de Redes Bayesianas a Partir de Dados Incompletos.** 2004. 118 f. Dissertação – Departamento de Computação e Estatística, Universidade Federal de Mato Grosso do Sul, Campo Grande, Mato Grosso do Sul, 2004.

MACHADO, Marcos Paulo. ***DINBAYES* – Componentes Dinâmicos em Sistema Especialista Probabilístico.** 2006. Trabalho de Conclusão de Curso (Graduação) –

Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2006.

MACHADO, Marcos P.; SIMÕES, Priscyla W. T. A.; MATTOS, Merisandra C. de. **O Desenvolvimento de um Sistema de Gerenciamento de Base de Conhecimento Dinâmicas em Sistema Especialista Probabilístico.** In: CONGRESSO SUL CATARINENSE DE COMPUTAÇÃO, 2., 2006, Criciúma. **Anais eletrônicos...** Criciúma: UNESC, 2006. Disponível em: www.dcc.unesc.net/sulcomp. Acesso: 20 out. 2006.

MARQUES, Isaac R.; MARIN, Heimar F. **Sistema de Apoio à Decisão em enfermagem..** Programa de Pós-Graduação em Ciências da Computação – Universidade Federal de Santa Catarina. 2002, Florianópolis

NASSAR, Sílvia Modesto. **Tratamento de Incerteza: Sistemas Especialistas Probabilísticos**. Disponível em: http://www.inf.ufsc.br/~silvia/disciplinas/sep/MaterialDidatico.pdf: Acesso: 30 out. 2006.

Netica Bayesian Network Software from Norsys. **Norsys Software Corp**. Disponível em: http://www.norsys.com. Acesso: 10 out. 2006.

PEREIRA, Marcos Aurélio. **Sistema Especialista On-Line de Auxílio ao Diagnóstico de Câncer de Próstata.** 2004. 179 f. Dissertação – Departamento de Pós Graduação de Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, 2004.

RABUSKE, R. A. **Inteligência artificial.** Florianópolis: UFSC, 1995.

REZENDE, S. O. **Sistemas inteligentes:** fundamentos e aplicações. São Paulo: Manole, 2003.

RODRIGUES, Ana Carolina Braga. **Biowoman** – Base de Conhecimento Dinâmica para Sistemas Especialistas Probabilísticos. 2002. Trabalho de Conclusão do curso (Graduação) - Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2002.

ROSA, Lilian Regina Junkes da . **Sistema Especialista Probabilístico Para Prognóstico De Doenças Bucais**. 2002. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2002.

ROSA, Lilian Regina Junkes et al. **Aplicação do Raciocínio Probabilístico no Desenvolvimento de um Sistema Especialista para Apoio ao Prognóstico de Doenças Bucais**. Revista de Iniciação Científica (Criciúma), v. 1, p. 101-114, 2003.

ROVER, Aires José. **Informática no direito:** inteligência artificial :introdução aos sistemas especialistas legais. Curitiba: Juruá, 2001.

RUSSELL, Stuart J.; NORVIG, Peter. **Inteligência artificial**. Rio de Janeiro: Elsevier,

2004

SAHEKI, André Hideaki. **Construção de uma Rede Bayesiana Aplicada ao Diagnóstico de Doenças Cardíacas.** 98 f. Dissertação – Título de Mestre em Engenharia, Escola Politécnica da Universidade de São Paulo, São Paulo, São Paulo, 2005.

SIMÕES, Priscyla W. T. A**.** ***SACI*** **- Sistema Especialista Probabilístico para Avaliação do Crescimento Infantil**. Dissertação – Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, 2001.

SOARES, Alexandre H. V.; SILVA, Carlos A.; ZAMBALDE, Andre L.. *Ceres*Sefs**: Um Sistema Especialista para o Cálculo da Necessidade de Calagem e Recomendação de Corretivo**. Curso de Ciências da Computação – Departamento de Ciência da Computação - Universidade Federal de Lavras. 2004, Lavras.

SPIEGELHALTER, D. J.; ABRAMS, K. R.; MYLES, Jonathan P. **Bayesian approaches to clinical trials and health care evaluation.** New Jersey: Wiley, c2004. 391 p.

TARONI, Franco; et al. **Bayesian networks and probabilistic inference in forensis science.** New Jersey: John Wiley & Sons, c2006.

TASCA, Rafael Fontana. ***SEPADI*** **- Sistema Especialista Probabilístico Para O Apoio Ao Diagnóstico E Profilaxia De Doenças Bovinas.** 2002..Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2002.

TIBIRIÇÁ, Carlos A. G.; NASSAR, Sílvia M. **Desenvolvimento de uma Abordagem Híbrida Difuso-Probabilística para Modelagem de Incertezas.** Programa de Pós-

Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, 2003.

WILLIAMSON, Jon. **Bayesian nets and causality:** philosophical and computational foundations. New York: Oxford University Press, 2005.

BIBLIOGRAFIA COMPLEMENTAR

BRIGNOLI, J. T.: **Modelo Híbrido Difuso-Probabilístico**: uma alternativa para sistemas especialistas. N f. Dissertação-Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, 2001.

CHAN, Mark C.; GRIFFITH, Steven W.; IASI, Anthony F. **Java 1001 dicas de programação**. São Paulo: Makron Books, 1999.

DEITEL, H.M; DEITEL, P. J. **Java**: **como programar**. São Paulo: Pearson Education do Brasil, 2005.

FARIAS, Anderson Rodrigo. **Métodos para Integração De Uma Base de Conhecimento De Um Sistema Especialista Probabilístico Utilizando O Ambiente De Programação C++ Builder.** 2001. Trabalho de Conclusão de Curso (Gradução) – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2001.

FERNANDES, Anita Maria da Rocha. Inteligência artificial: noções gerais. Florianópolis: Visual Books, 2003.

FERNANDES, Anita Maria da Rocha e Colaboradores. **Inteligência artificial aplicada à saúde.** Florianópolis: Visual Books, 2004.

GANASCIA, Jean Gabriel. **Inteligência artificial.** São Paulo: Ática, 1997.

GUINZANI, Jhonas Bonfante. **Mineração De Dados Em Redes Bayesianas Utilizando a API Da Shell Belief Network Power Constructor (BNPC).** 2006. n f.Trabalho de Conclusão de Curso (Gradução) – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2006.

HAGGAR, Peter. **Java**: **guia prático de programação**. Rio de Janeiro: Ed. Campus, 2000.

KOOSIS, Donald; KOOSIS, David; ROQUE, Kátia A. **Programação com Java**. Rio de Janeiro: Ed. Campus, 1999.

MANARIN, Daiane de Nez .**Aquisição de Conhecimento em Sistemas Especialistas Probabilísticos por meio da Descoberta de Conhecimento em Base de Dados para Construção de Redes Bayesianas**. 2004. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2004.

NEGRINO, Tom; SMITH, Dori. **Javascript: para a World Wide Web.** Rio de Janeiro: Ed. Campus, 2000.

OLIVEIRA, Adelize Generini de. **Java: a linguagem de programação da internet.** Florianópolis: Bookstore, 1996.

RICH, E., KNIGHT, K. **Inteligência artificial.** São Paulo: Makron Books, 1993.
RODRIGUES, Ana Carolina Braga. ***BIOWOMAN* – Base De Conhecimento Dinâmica para Sistema Especialista Probabilístico.** Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2002.

SCUSSEL, Telma. **Base de Conhecimento Para Um Sistema Especialista Probabilístico De Apoio Ao Ensino Do Diagnóstico Etiológico Da Lombalgia.** 2001.Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2001.

SILVA, Osmar J. **Javascript:** guia prático do webmaster. São Paulo: Érica, 2000.

TIBIRIÇÁ, Carlos Augusto Gonçalves. **Uma Abordagem Híbrida Fuzzy-Bayesiana para Modelagem de Incertezas.** N f.Dissertação – Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, 2005.

TODD, Nick; SZOLKOWSKI, Mark. **JavaServer Pages:** o guia do desenvolvedor. Rio de Janeiro: Elsevier, 2003.

WHITBY, Blay. **Inteligência artificial:** um guia para iniciantes. São Paulo: Madras,

2004.

APÊNDICE

APÊNDICE A

TUTORIAL DE INSTALAÇÃO DA NETICA JAVA API

A Netica Java API *(Aplication Program Interface)* foi desenvolvida pela empresa *Norsys Software Corporation* de Vancouver, no Canadá, esta *shell* é utilizada para a construção de redes bayesianas. Por meio do código da integração é possível utilizar o arquivo da RB para encontrar o resultado do sistema, permitindo que por meio da identificação de determinadas entradas (evidências), seja quantificado o valor de todas as variáveis relacionadas no sistema (nós), ou seja, determinando a probabilidade de algo acontecer de acordo com os valores iniciais.

A Netica Java API, consiste em uma biblioteca de funções que permitem manipulação de redes bayesianas. A biblioteca com funções API da *shell* Netica, pode ser obtida na página da Norsys (www.norsys.com) , sendo que a mesma é constituída pelos arquivos apresentados na Figura 49, sendo assim a biblioteca onde contém todas as funções da imagem se encontra dentro da pasta *bin*, cujo seu nome é NeticaJ.jar.

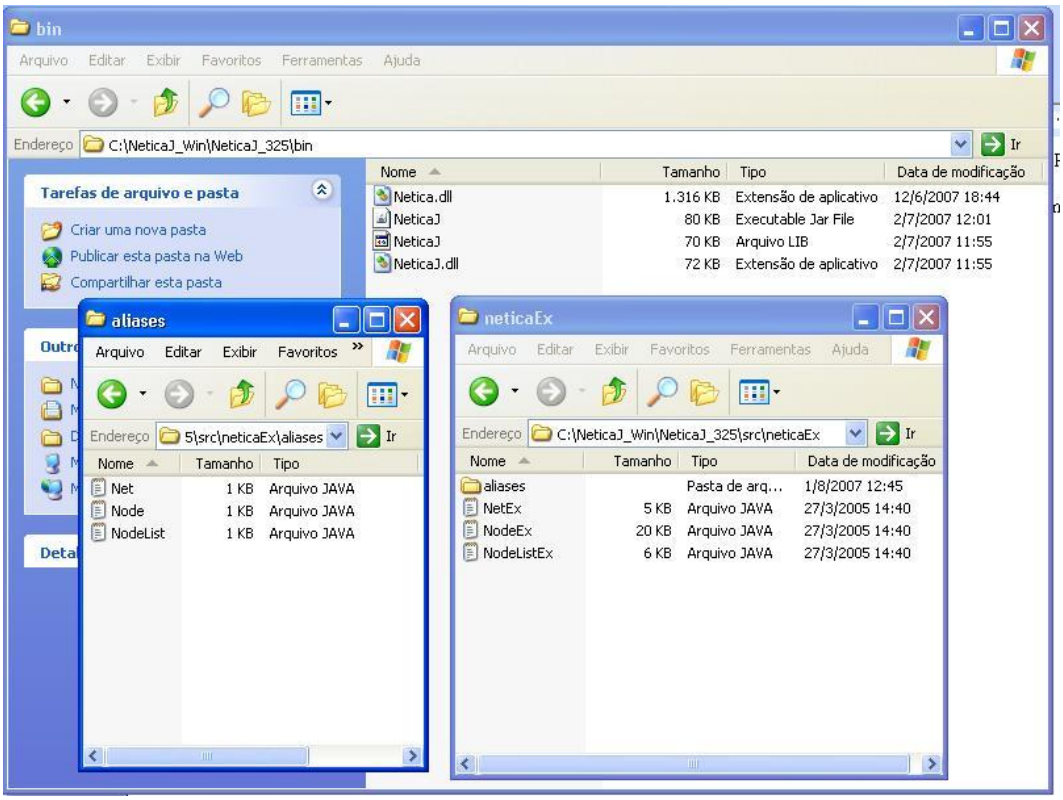

Figura 49 . Arquivos disponíveis no pacote Netica J

Depois que se tem todo o pacote Netica Java API em seu computador é necessário configurar seu ambiente de variáveis do sistema operacional, sendo que você deve colocar o caminho que se encontra a biblioteca Netica Java API na variável Path do seu sistema operacional seguido de ponto e vírgula (;). Como esta aplicação foi desenvolvida no sistema operacional Windows XP, veja a seguir como deve ser a configuração do seu ambiente, conforme ilustração da Figura 50.

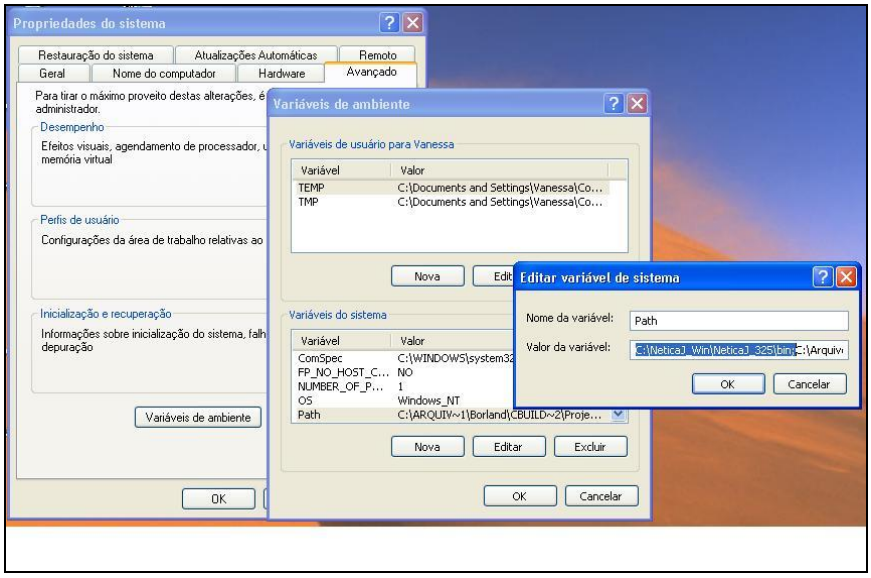

Figura 50 . Ambiente de variáveis para a configuração do Path

Depois disso no ambiente de programação NetBeans 5.5 é necessário adicionar a biblioteca no projeto a ser desenvolvido. Nas propriedades do projeto em desenvolvimento se deve adicionar a jar do Netica, ou seja, a NeticaJ.jar e no ambiente de desenvolvimento antes de iniciar o código fonte é necessário fazer todos os imports da API, ou seja, os imports a seguir: norsys.netica.Environ; norsys.neticaEx.aliases.Net e norsys.netica.*.

Quando já tiver sido feito estes procedimentos já pode começar a fazer o código fonte desejável com interface interativa usando componentes e as funções necessárias para a integração. A figura abaixo ilustra como e onde dever se adicionada a biblioteca e os imports no NetBeans.

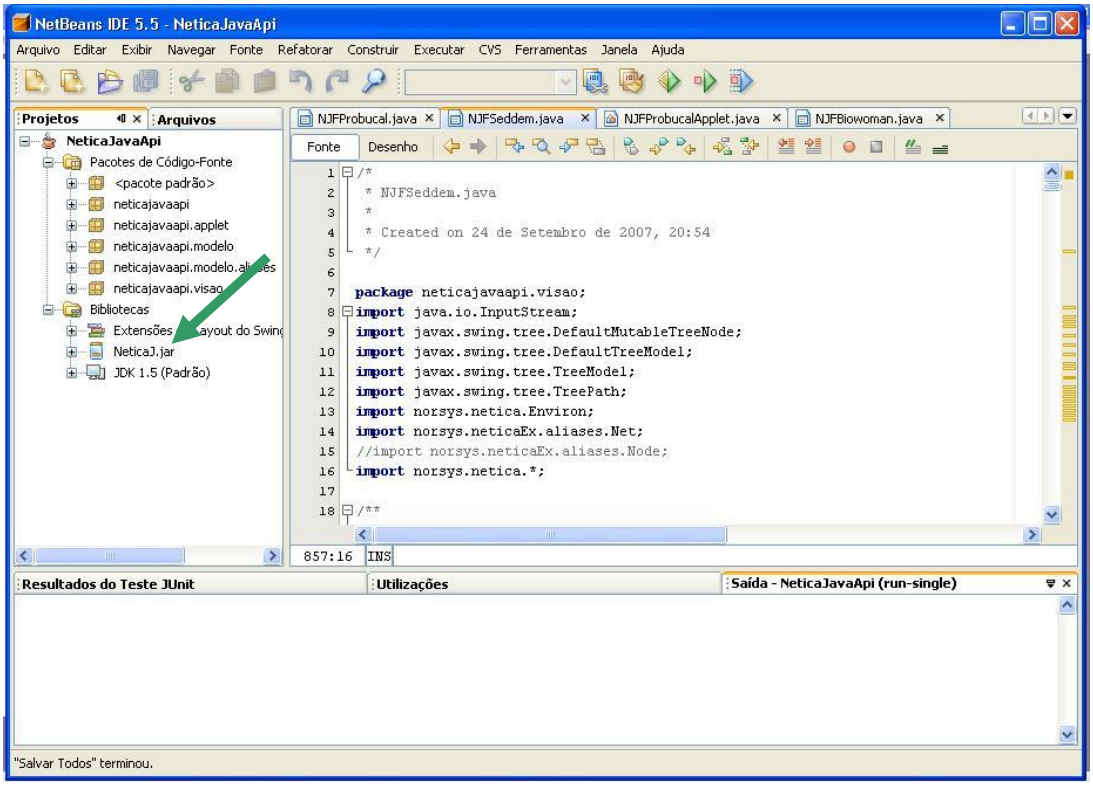

Figura 51 . Ambiente do NetBeans 5.5.

Na figura 51 sé ilustrado o ambiente NetBeans 5.5 onde é adicionada a biblioteca da Netica Java API.

APENDICE B

MÉTODOS DA CLASSE ENVIRON

A classe Environ é o ambiente onde são realizadas todas as operações do Netica, ela é um objeto do pacote *norsys.netica.Environ.*

Tabela 14 – Métodos da classe Environ

Tipo de Retorno / Assinatura	*Método / Descrição*
void	**Finalize ()** Fecha o ambiente criado pelo Netica e libera todos os seus recursos alocados na memória.
int	**getArgumentChecking()** Retorna o argumento verificando qual o nível da classe atual , que é o grau em que Netica controla as chamadas de argumentos.
char	**getCaseFileDelimChar()** Agrupa os caracteres para utilizar como delimitação na construção de processos.
static **Environ**	**getDefaultEnviron()** Retorna o último ambiente Environ construído.
double	**getMaxMemoryUsage()** È obrigatório a utilização do método ***getMemoryUsageLimit()*** para o funcionamento deste.
double	**getMemoryUsageLimit()** Retorna o limite atual que é permitido alocar na memória para o

	ambiente.
char	**getMissingDataChar()** Obtém o caracter usado para indicar dados ausentes ao criar processos.
int	**getVersion ()** Retorna o número da versão do Netica multiplicado por 100.
java.lang.String	**getVersionString ()** Retorna uma String contendo o número da versão, um espaço, um código para o tipo de máquina ou sistema operacional que ele está executando, uma vírgula, o nome do programa, e, por último, um código indicando a construção de algumas informações (entre parênteses).
void	**setArgumentChecking (int setting)** Define em que nível o Netica deve verificar os argumentos passados.
void	**setCaseFileDelimChar (char newChar)** Define qual o caracter a ser utilizado como delimitador de dados em processos criados pelo Netica.
void	**setMaxMemoryUsage (double maxMem)** È obrigatório a utilização do método ***setMemoryUsageLimit (maxMem)*** para o seu funcionamento.
void	**setMemoryUsageLimit (double maxMem)** Método que ajusta a quantidade de memória alocada para o ambiente Netica, este pode ser chamado a qualquer momento.
void	**setMissingDataChar (char newChar)**

	Define o caracter a ser utilizado para indicar os dados ausentes no arquivo que o Netica criou.

MÉTODOS DA CLASSE NET

A classe Net é o ambiente onde são realizadas todas as operações do Netica, ela é um objeto do pacote *norsys.netica.net*

Tabela 15 – Métodos da classe Net

Tipo de Retorno / Assinatura	*Método / Descrição*
void	**absorb (NodeList nodeList)** É obrigatório renomear para o uso do método *absorbNodes.*
void	**absorbNodes (NodeList nodeList)** Absorve os estados dos *nós*, a fim de retirar da rede, mais as probabilidades a priori continuam inalteradas, retirando somente os nós do ambiente.
void	**compile()** Compila a RB e atualiza a crenças das probabilidades (propagação de valores)
NodeList	**copyNodes (NodeList nodeList)** Duplica os *nós* em *NodeList* e coloca os duplicados na RB.
void	**delete()** Obrigatório a utilização do método *finalize*.

NodeList	**duplicateNodes (NodeList nodeList)** Obrigatório a utilização do método *copyNodes.*
void	**finalize()** Finaliza a rede e libera todos os seus recursos utilizados (por exemplo, libera a memória alocada), incluindo todas as suas subestruturas (por exemplo, os *nós*).
int	**generateRandomCase(NodeList nodeList, int method, double timeout)** Gera um caso randômico por simulação.
static java.util.Vector	**getAllNets(Environ env)** Recupera do *env* uma coleção de todas as redes definidas no mesmo Environ.
java.lang.String	**getAllNodesets(boolean includeSystem)** Retorna uma string, que é um conjunto de todos os nós definidos para esta rede, separados por vírgula, por ordem de prioridade, sendo que a de mais alta prioridade aparece em primeiro lugar.
int	**getAutoUpdate()** Retorna a rede atual com atualização de valores.
static java.lang.String	**getConstructorClass()** Recupera o classname da classe que o Netica-J utiliza quando cria uma nova rede automaticamente.
NodeList	**getElimOrder()** Retorna uma lista de nós da rede na sua "ordem de eliminação" (que é utilizada para a triangulação da compilação desta rede) ou null quando não houver ordem de associação na rede.
java.lang.String	**getFileName()**

	Retorna o nome completo do arquivo, incluindo o caminho completo de ondea rede foi aberta ou lida.
double	**getFindingsProbability()** Retorna a probabilidade conjunta das probabilidades da rede até o momento (incluindo até os valores negativos de probabilidade).
double	**getJointProbability(NodeList nodeList, int[] nodeStates)** Retorna a probabilidade conjunta que cada nó na *nodeList* é do estado correspondente no *nodeStates*, levando em consideração as conclusões atuais da RB.
int[]	**getMostProbableConfig(NodeList nodeList)** Encontra a configuração mais provável, também conhecida como a mais provável explicação (MPE), para todos os nós da rede.
Node	**getNode(java.lang.String nodeName)** Retorna o nó da rede, que tem exatemente o mesmo nome utilizado na comparação.
NodeList	**getNodes()** Retorna uma lista de todos os nós da rede.
java.awt.Color	**getNodesetColor(java.lang.String nodeset)** Recupera a cor do *nodeset* na rede.
void	**getRelatedNodes(NodeList relatedNodes, java.lang.String relation, NodeList nodeList)** Encontra todos os nós que suportam o relacionamento *relation* com qualquer membro do *nodelist* e coloca na *relatedNodes*.
void	**readCase(long[] casePosn, Streamer file, NodeList nodeList,**

	long[] idNum, double[] freq) Obrigatório a utilização do método *readFindings*.
void	**readFindings(long[] casePosn, Streamer file, NodeList nodeList, long[] idNum, double[] freq)** Lê um conjunto de resultados (ou seja, um caso) de um arquivo contendo um ou mais casos.
int	**redoOperation(double toWhen)** Método que refaz uma operação desfeita pela undoLastoperation.
void	**reorderNodesets(java.lang.String nodesetOrder)** Reordena a ordem de prioridades dos nós na rede.
java.lang.String	**reportJunctionTree()** Retorna uma string contento um relaório da junção da árvore da rede, semelhante a que é produzida pelo Netica Application na operação "Report -> Junction Tree"
void	**retractFindings()** Retrai todos os resultados de todos os nós da rede, exceto nos nós "constante" (que utiliza o método ***Node.finding().clear()***
void	**reviseCPTsByCaseFile(Streamer file, NodeList nodeList, double degree)** Revê o CPTs dos nós para ter em conta os casos de um determinado arquivo.

APÊNDICE C

TUTORIAL DE FUNCIOMENTO DA APPLET

A *shell* Netica possui uma limitação no seu funcionamento para Web, para resolver esta limitação e para um correto funcionamento da applet foi estabelecido e criado uma única aplicação jar, onde esta necessita-se de todas as informações que executarão a applet, como por exemplo, a biblioteca do Netica, a biblioteca jFreeChart para o funcionamento do gráfico.

Foi criado um arquivo denominado buil.xml, onde este possui o código citado abaixo, neste código é possível visualizar toda a parte de realização da junção dos arquivos e criando somente um arquivo.jar.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- You may freely edit this file. See commented blocks below for -->
<!-- some examples of how to customize the build. -->
<!-- (If you delete it and reopen the project it will be recreated.) -->
<project name="NeticaJavaApi" default="default" basedir=".">
    <description>Builds, tests, and runs the project NeticaJavaApi.</description>
    <import file="nbproject/build-impl.xml"/>
    <!--

    There exist several targets which are by default empty and which can be
    used for execution of your tasks. These targets are usually executed
    before and after some main targets. They are:

      -pre-init:                 called before initialization of project properties
      -post-init:                called after initialization of project properties
      -pre-compile:              called before javac compilation
      -post-compile:             called after javac compilation
      -pre-compile-single:       called before javac compilation of single file
      -post-compile-single:      called after javac compilation of single file
      -pre-compile-test:         called before javac compilation of JUnit tests
      -post-compile-test:        called after javac compilation of JUnit tests
      -pre-compile-test-single:  called before javac compilation of single JUnit test
      -post-compile-test-single: called after javac compilation of single JUunit test
      -pre-jar:                  called before JAR building
      -post-jar:                 called after JAR building
      -post-clean:               called after cleaning build products

    (Targets beginning with '-' are not intended to be called on their own.)
```

Example of inserting an obfuscator after compilation could look like this:

```
    <target name="-post-compile">
        <obfuscate>
            <fileset dir="${build.classes.dir}"/>
        </obfuscate>
    </target>
```

For list of available properties check the imported nbproject/build-impl.xml file.

Another way to customize the build is by overriding existing main targets. The targets of interest are:

```
  -init-macrodef-javac:     defines macro for javac compilation
  -init-macrodef-junit:     defines macro for junit execution
  -init-macrodef-debug:     defines macro for class debugging
  -init-macrodef-java:      defines macro for class execution
  -do-jar-with-manifest:    JAR building (if you are using a manifest)
  -do-jar-without-manifest: JAR building (if you are not using a manifest)
  run:                      execution of project
  -javadoc-build:           Javadoc generation
  test-report:              JUnit report generation
```

An example of overriding the target for project execution could look like this:

```
    <target name="run" depends="NeticaJavaApi-impl.jar">
        <exec dir="bin" executable="launcher.exe">
            <arg file="${dist.jar}"/>
        </exec>
    </target>
```

Notice that the overridden target depends on the jar target and not only on the compile target as the regular run target does. Again, for a list of available properties which you can use, check the target you are overriding in the nbproject/build-impl.xml file.

```
-->

<target name="app.dist">
    <delete file="dist/app.jar" failonerror="false"/>
    <mkdir dir="dist/app"/>
    <unzip src="lib/NeticaJ.jar" dest="dist/app"/>
    <unzip src="lib/jcommon-1.0.0-pre2.jar" dest="dist/app"/>
    <unzip src="lib/jfreechart-1.0.0-pre2.jar" dest="dist/app"/>
    <copy todir="dist/app">
        <fileset dir="build/classes/" includes="**/*"/>
    </copy>
    <zip destfile="dist/app.jar">
```

```
        <fileset dir="dist/app" includes="**/*"/>
      </zip>
      <delete dir="dist/app"/>
      <signjar alias="app" keystore="assinatura" storepass="123456" keypass="123456"
jar="dist/app.jar"/>
   </target>

  <target name="criar.assinatura">
      <genkey dname="CN=app" alias="app" keystore="assinatura" storepass="123456"
keypass="123456"/>
   </target>

</project
<?xml version="1.0" encoding="UTF-8"?>
<!-- You may freely edit this file. See commented blocks below for -->
<!-- some examples of how to customize the build. -->
<!-- (If you delete it and reopen the project it will be recreated.) -->
<project name="NeticaJavaApi" default="default" basedir=".">
   <description>Builds, tests, and runs the project NeticaJavaApi.</description>
   <import file="nbproject/build-impl.xml"/>
   <!--

   There exist several targets which are by default empty and which can be
   used for execution of your tasks. These targets are usually executed
   before and after some main targets. They are:

     -pre-init:                 called before initialization of project properties
     -post-init:                called after initialization of project properties
     -pre-compile:              called before javac compilation
     -post-compile:             called after javac compilation
     -pre-compile-single:       called before javac compilation of single file
     -post-compile-single:      called after javac compilation of single file
     -pre-compile-test:         called before javac compilation of JUnit tests
     -post-compile-test:        called after javac compilation of JUnit tests
     -pre-compile-test-single:  called before javac compilation of single JUnit test
     -post-compile-test-single: called after javac compilation of single JUunit test
     -pre-jar:                  called before JAR building
     -post-jar:                 called after JAR building
     -post-clean:               called after cleaning build products

   (Targets beginning with '-' are not intended to be called on their own.)

   Example of inserting an obfuscator after compilation could look like this:

      <target name="-post-compile">
         <obfuscate>
            <fileset dir="${build.classes.dir}"/>
         </obfuscate>
      </target>
```

For list of available properties check the imported
nbproject/build-impl.xml file.

Another way to customize the build is by overriding existing main targets.
The targets of interest are:

-init-macrodef-javac: defines macro for javac compilation
-init-macrodef-junit: defines macro for junit execution
-init-macrodef-debug: defines macro for class debugging
-init-macrodef-java: defines macro for class execution
-do-jar-with-manifest: JAR building (if you are using a manifest)
-do-jar-without-manifest: JAR building (if you are not using a manifest)
run: execution of project
-javadoc-build: Javadoc generation
test-report: JUnit report generation

An example of overriding the target for project execution could look like this:

```
    <target name="run" depends="NeticaJavaApi-impl.jar">
        <exec dir="bin" executable="launcher.exe">
            <arg file="${dist.jar}"/>
        </exec>
    </target>
```

Notice that the overridden target depends on the jar target and not only on
the compile target as the regular run target does. Again, for a list of available
properties which you can use, check the target you are overriding in the
nbproject/build-impl.xml file.

-->

```
  <target name="app.dist">
      <delete file="dist/app.jar" failonerror="false"/>
      <mkdir dir="dist/app"/>
      <unzip src="lib/NeticaJ.jar" dest="dist/app"/>
      <unzip src="lib/jcommon-1.0.0-pre2.jar" dest="dist/app"/>
      <unzip src="lib/jfreechart-1.0.0-pre2.jar" dest="dist/app"/>
      <copy todir="dist/app">
          <fileset dir="build/classes/" includes="**/*"/>
      </copy>
      <zip destfile="dist/app.jar">
          <fileset dir="dist/app" includes="**/*"/>
      </zip>
      <delete dir="dist/app"/>
      <signjar alias="app" keystore="assinatura" storepass="******" keypass="******"
jar="dist/app.jar"/>
    </target>

  <target name="criar.assinatura">
```

```
      <genkey dname="CN=app" alias="app" keystore="assinatura" storepass="******"
keypass="******"/>
   </target>

</project
```

No código destacado é possível visualizar toda a parte de deleção do arquivo anterior, criação do arquivo jar que receberá todos os arquivos necessários pra o funcionamento da applet, a compactação das biblitecas utilizadas para o desenvolvimento, a cópia deste arquivo para o diretório buil/classes, assinatura da applet garantindo que a mesma não é mal intencionada em relação ao usuário.

APÊNDICE D

Artigo do projeto de pesquisa.

Método para Integração de uma Base de Conhecimentos de um Sistema Especialista Probabilístico Utilizando a Netica Java

Vanessa Felisberto Bilésimo, Priscyla Waleska Targino de Azevedo Simões[2], Fabrício Giordani[2], Merisandra Côrtes de Mattos[2]

[1]Acadêmica do curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias - Universidade do Extremo Sul Catarinense (UNESC) – Criciúma, SC - Brasil

[2]Professor(a) do curso de Ciência da Computação - Unidade Acadêmica de Ciências, Engenharias e Tecnologias - Universidade do Extremo Sul Catarinense (UNESC) – Criciúma, SC - Brasil

vanessabilesimo@gmail.com, pri@unesc.net, fgiordani@gmail.com, mem@unesc.net

Abstract. *In some areas of application, the reasoning for the solution of a specific problem is not exact, and is accompanied by some degree of uncertainty, as in some medical diagnoses, and to address this issue, among other techniques emerged the Systems Specialists Probabilistics, whose Knowledge Base is represented through networks Bayesians. In this context, the goal of this research is to implement a method to integrate a Knowledge Base System of a Specialist Probabilistic using the Netica Java API. The methodology of development began with the study of the classes and methods provided by Netica Java API, followed by the choice of some networks Bayesians for the use of case study, ending with the implementation and documentation of the system developed. This search resulted in new versions of System Support Specialist for the Diagnosis of Diseases Exantematicas Maculopapulosas with Compulsory Eruption, System Specialist Probabilistic for Prognosis of Oral Diseases, Biowoman-Dynamic Knowledge Base Systems for Experts Probabilistic developed previously at the University of the Far South Catarinense. These applications have been implemented in the programming environment NetBeans IDE 5.5 through the use of Netica Java API. Each interface offered resources of inference, module of explanation, aid, and graphical analysis of diagnostic hypothesis. It can be concluded through this research that the results were satisfactory, considering that the method of integration based on Netica Java API and applets developed met the expectations of experts of the field of application*

Resumo. *Em alguns domínios de aplicação, o raciocínio para a solução de um determinado problema não é exato, sendo acompanhado por certo grau de incerteza, como acontece em certos diagnósticos médicos e, para tratar desta questão, entre outras técnicas surgiram os Sistemas Especialistas Probabilísticos, cuja Base de Conhecimento é representada por meio de Redes Bayesianas. Neste contexto, o objetivo desta pesquisa é aplicar um método para integração de uma Base de Conhecimento de um Sistema Especialista Probabilístico utilizando a Netica Java API. A metodologia de desenvolvimento iniciou com o estudo das classes e métodos disponibilizados pela Netica Java API, seguindo-se com a escolha de algumas Redes Bayesianas para a utilização do estudo de caso, finalizando-se com a implementação e documentação do sistema desenvolvido.*

Esta pesquisa resultou em novas versões do Sistema Especialista para o Apoio ao Diagnóstico de Doenças Exantemáticas Maculopapulosas com Erupção Obrigatória, Sistema Especialista Probabiístico para Prognóstico de Doenças Bucais, Biowoman – Base de Conhecimento Dinâmica para Sistemas Especialistas Probabilístico desenvolvidos anteiormente na Universidade do Extremo Sul Catarinense. Estas aplicações foram implementadas no ambiente de programação NetBeans IDE 5.5 por meio da utilização da Netica Java API. Cada interface ofereceu recursos de inferência, módulo de explicação, de ajuda, e de análise gráfica das hipótese diagnósticas. Pode-se concluir por meio desta pesquisa que os resultados obtidos foram satisfatórios, considerando-se que o método de integração baseado na Netica Java API e as applets desenvolvidas atenderam as expectativas dos especialistas do domínio de aplicação.

1. Introdução

Os Sistemas Especialistas Probabilísticos (SEP) são implementados e desenvolvidos com a finalidade de atuar em diversas áreas, desde a resolução de problemas até situações que envolvem maior complexidade como o diagnóstico de uma doença (TIBIRIÇA, NASSAR, 2003).

Nestes sistemas, o raciocínio probabilístico, fundamentado na teoria da probabilidade, é utilizado para modelagem de problemas que apresentam incerteza por aleatoriedade, ou seja, raciocínio cujo resultado mesmo em condições normais de experimentação, varia de uma observação para outra, dificultando desta maneira a previsão de um resultado futuro (LUNA, 2004).

Em alguns domínios de aplicação, o raciocínio para a solução de um determinado problema não é exato, sendo acompanhado por um certo grau de incerteza, como acontece em certos diagnósticos médicos (NASSAR, 2006), e, para tratar desta questão, entre outras técnicas surgiram os modelos probabilisticos que caracterizam os sistemas especialistas probabilísticos, cuja BC é representada por meio de RB.

Segundo Luna (2004) existem motivos práticos para a utilização de redes bayesianas, pois essa forma de representação do conhecimento incerto permite analisar grandes quantidades de dados, para extrair conhecimentos úteis e, dessa forma, auxiliar os especialistas em seu processo de tomada de decisão em domínios de conhecimento como: saúde, indústria, computação, redes, entre outros.

Apesar da utilização de redes bayesianas na saúde, os SEP desenvolvidos na área médica costumam ser pouco utilizados no apoio à prática clínica, pois, conforme surgem novas informações sobre dado sinal, sintoma e prevalência de uma doença na população (informações que definem as probabilidades condicionais *a priori* da rede bayesiana), essas probabilidades condicionais precisam ser atualizadas, o que demanda tempo do engenheiro do conhecimento e do especialista do domínio de aplicação nesse processo continuado, e acaba tornando esses sistemas apenas resultados de pesquisa, desatualizados com o passar do tempo (MACHADO, 2006).

Nesse contexto, muitas aplicações que utilizam redes bayesianas para representação da base de conhecimento de SEP, utilizam a *shell* Netica e a Netica C API, pois essa s*hell* é muito aplicada nesses sistemas, por oferecer gratuitamente uma versão limitada, e por ser utilizada no apoio ao ensino de redes bayesianas na disciplinas de graduação e pós graduação do cursos de Ciência da Computação, além ser uma ferramenta didática, de fácil utilização e que permite boa interação com os especialistas do domínio de aplicação. (NASSAR, 2006).

A primeira API dessa *Shell* foi desenvolvida para ser utilizada na linguagem C, e, considerando o processo de modelagem de uma base de conhecimento por meio da Aquisição de Conhecimento (AC) não automatizada, oferecia recursos para gerenciamento: do ambiente Netica, de nós, de probabilidades condicionais, das relações de dependência entre os nós (*links*), de inferência, além de outras funcionalidades para aquisição de conhecimento automatizada.

Com o passar dos anos, os recursos dessa API foram atualizados, e seu fabricante desenvolveu uma nova biblioteca de funções, denominada Netica Java API, que apresenta as funções originais, e novos recursos que visam facilitar sua utilização, além de ser desenvolvida conforme a filosofia e metodologia do Java (NORSYS, 2006).

Para utilizar uma nova API da s*hell* Netica, não basta traduzir o código fonte inicial para uma nova linguagem, é necessário entender a filosofia e metodologia dessa nova API baseada no paradigma de programação orientada a objetos, além de explorar seus recursos na linguagem de programação escolhida para o desenvolvimento do sistema, para isso, o programador tem que possuir bom conhecimento da linguagem de programação, da API que será utilizada, além de bom embasamento na área de IA, no que se refere ao processo de construção de SEP.

A Netica Java API foi desenvolvida para SEP a serem integrados em Java, Cozman (2001) destaca grande potencial da linguagem Java na construção de SEP por oferecer quatro vantagens: por apresentar portabilidade; por ser adotado por *browsers* na internet; possibilita trabalhar e utilizar ferramentas como a s*hell* Netica, que permitem modelar sistemas sobre o domínio da incerteza e por ser uma linguagem orientada à objetos.

Nesse sentido, busca-se por meio dessa pesquisa, estudar e aplicar a metodologia de utilização da Netica Java API, no desenvolvimento de um protótipo sistema especialista probabilístico possibilitando uma contribuição na área de Inteligência Artificial (IA).

1. Sistemas Especialistas Probabilísticos

Os SEP são sistemas computacionais desenvolvidos para auxiliar na resolução de determinados problemas em diversas áreas do conhecimento que representam a incerteza por aleatoriedade. O conhecimento necessário para a solução dos problemas é adquirido por meio de um especialista do conhecimento (WILLIAMSON, 2005).

A presença da incerteza por aleatoriedade influencia no modo como um especialista toma suas decisões. Em uma aplicação esta incerteza pode estar presente tanto nos dados de entrada como na solução do problema. Em um caso médico, por exemplo, podem existir dois pacientes com sinais e sintomas similares, porém, seus diagnósticos são diferentes. Com isso, o médico nesta situação para prescrever o diagnóstico de determinado paciente deve racionar por meio de incerteza, baseando-se em probabilidades já ocorridas anteriormente com muita freqüência sob o domínio da incerteza (MARQUES e MARIN, 2002)

O SEP desenvolvido deve ser capaz de simular e tomar decisões que seriam realizadas por especialistas, sendo que estas decisões devem se comportar de forma semelhante à decisão do especialista (TIBIRIÇA; NASSAR, 2003; SAHEKI, 2005).

Sendo assim, a arquitetura de um sistema especialista probabilístico apresenta a Base de Conhecimento que é representada por meio de redes bayesianas, sendo o conhecimento representado esquematicamente na forma de um grafo acíclico

direcionado[14]. Os *nós* do grafo representam dois tipos de variáveis: as variáveis de entrada (sinais e evidências) e as variáveis de saída (conjunto de hipóteses diagnósticas) (HUDSON, COHEN, 1999).

A Aquisição de Conhecimento é a tarefa fundamental para a construção da base de conhecimento, sendo a parte mais sensível no processo de construção e atualização da BC de um SEP, seno assim, a aquisição de conhecimento é a maneira de reunir informações de um ou mais especialistas e organizá-las, ou seja, analisar, interpretar e precodificar o conhecimento para uma forma compreensível pela máquina.

O Motor de Inferência é o processador do conhecimento do SEP, ou seja, é o elemento mais importante, o núcleo do sistema, pois tem como principal função inferir conclusões e gerar novos fatos, sendo que sua interpretação do conhecimento é reduzida por meio de técnicas como a dedução e a lógica probabilísticas onde o usuário pode solicitar o conhecimento armazenado na BC também é vista como motor de inferência (MARQUES; MARIN, 2002).

O Especialista do Conhecimento do domínio de aplicação fornece o conhecimento da área do problema e deve possuir experiências e formação na área específica do domínio em que a base de conhecimento será construída. É a pessoa responsável em repassar as informações para o engenheiro do conhecimento.

Onde o Engenheiro do Conhecimento é a pessoa responsável pelo processo de aquisição de conhecimento. Ele é o especialista em representações do conhecimento e sua tarefa principal é auxiliar o especialista do domínio a articular o conhecimento necessário para o desenvolvimento do SEP, além de transformar as informações adquiridas em dados manipuláveis pelo computador (LUGER, 2004).

A Interface com o Usuário (IU) procura tornar o uso do sistema fácil e agradável, eliminando-se as suas complexidades, sendo que a interface deve ser bem projetada, baseando-se em técnicas interativas como menus, gráficos, cores, janelas, pois deve ser considerar o grau de familiarização com o usuário, considerando-se que estes podem ser especialistas no assunto ou iniciantes (SPIEGELHALTER, 2004).

Conforme Cozman (2002) descreve, uma das atividades mais difíceis de se solucionar pela Inteligência Artificial é a manipulação de incertezas, e nesse sentido, técnicas como o raciocínio probabilístico, estão sendo utilizadas para o apoio nesta área onde as informações são vagas e incompletas.

Por meio de representações gráficas das evidências probabilísticas é possível manipular a incerteza com base em princípios matemáticos fundamentados e modelar o conhecimento do especialista do domínio de forma intuitiva (RUSSEL; NORVIG, 2004).

Estas incertezas são modificadas periodicamente após observações de novos dados ou resultados gerados por meio dos fatos e regras. A operação que potencializa a medida das incertezas é conhecida como operação bayesiana sendo baseada na fórmula de Bayes[15] (RUSSEL; NORVIG, 2004).

2. Redes Bayesianas

A Rede Bayesiana que forma a base de conhecimento dos SEPs, é um modelo gráfico matemático que tem como estrutura um grafo direcionado acíclico que representa a

[14] Quando as arestas têm uma direção associada e indicada por setas na representação gráfica (LUGER, 2004).

[15] Teorema que calcula a probabilidade de uma hipótese dado uma nova evidência (RUSSEL; NORVIG, 2004).

distribuição de probabilidades conjuntas das variáveis que modelam o domínio do conhecimento (LADEIRA et al, 2003).

Quando um especialista cria a estrutura de uma rede bayesiana, é necessário estimar ou aprender as probabilidades associadas a cada variável (JENSEN, 2001).

À medida que o número de variáveis cresce em um modelo, dificulta ao especialista definir a relação de dependência entre elas, no entanto, é vantajoso utilizar um banco de dados para realizar a inferência na estrutura da rede bayesiana e suas probabilidades (WILLIAMSON, 2005).

A RB permite expressar as assertivas de independência de forma visual e de fácil percepção; representa e armazena uma distribuição conjunta de probabilidades condicionais de forma sucinta, explorando a esparcidade da rede no relacionamento entre as variáveis, ou seja, quantificando a intensidade das relações nas variáveis, tornando assim o processo de inferência eficiente computacionalmente (SPIEGELHALTER, 2004).

Os nós representam variáveis e os arcos significam a influência causal entre variáveis diretamente ligadas, sendo que a intensidade destas influências são expressas por meio das probabilidades condicionais (SAHEKI, 2005).

O processo de inferência em redes bayesianas mostra que, uma vez construída uma representação probabilística por meio do modelo de RBs, para a incerteza presente no relacionamento entre variáveis de um domínio de dados, uma das tarefas mais importantes consiste em obter estimativas de probabilidades de eventos relacionados aos dados, à medida que novas informações ou evidências sejam conhecidas (RUSSEL; NORVIG, 2005).

Para a construção de uma Rede Bayesiana os sistemas utilizam representações gráficas de dependências probabilísticas. Essa representação permite manipular a incerteza com base em princípios matemáticos fundamentados, além de modelar o conhecimento do especialista do domínio de uma forma intuitiva.

Para isso são utilizadas algumas ferramentas de apoio a construção dessas redes, e entre essas, a *shell* Netica (Norsys)[16] (FLORES; PEROTTO; VICCARI, 2003). A primeira API dessa *Shell* foi desenvolvida para ser utilizada na linguagem C, e, considerando o processo de modelagem de uma base de conhecimento por meio da Aquisição de Conhecimento (AC) não automatizada, oferecia recursos para gerenciamento: do ambiente Netica, de nós, de probabilidades condicionais, das relações de dependência entre os nós (links), de inferência, além de outras funcionalidades para aquisição de conhecimento automatizada. Esses recursos atualmente estão disponibilizados e atualizados para o ambiente Java (NORSYS, 2006).

A BC e o motor de inferência necessitam se comunicar com os demais módulos do SEP, para isso utiliza-se uma linguagem de programação escolhida para o desenvolvimento do sistema, chamadas a Application Programmer Interfaces (API) de uma s*hell* de Redes Bayesianas (RB) que, entre seus recursos, oferece o motor de inferência, que por meio do teorema de Bayes propaga e gera as probabilidades *a posteriori*, a partir das probabilidades *a priori* do domínio de conhecimento representado (LUNA, 2004).

A Rede Bayesiana (RB) é um modelo gráfico matemático que tem como estrutura um grafo direcionado acíclico que representa a distribuição de probabilidades conjuntas das variáveis que modelam o domínio do conhecimento incerto (LADEIRA; FLORES; VICCARI, 2002), (LADEIRA et al, 2003).

---

[16] Netica Bayesian Network Software from Norsys. **Norsys Software Corp**. Disponível em: http://www.norsys.com.

O processo de inferência em redes bayesianas mostra que, uma vez construída uma representação probabilística por meio do modelo de RBs, para a incerteza presente no relacionamento entre variáveis de um domínio de dados, uma das tarefas mais importantes consiste em obter estimativas de probabilidades de eventos relacionados aos dados, à medida que novas informações ou evidências sejam conhecidas (RUSSEL; NORVIG, 2005).

A importância da inferência bayesiana não é restrita somente ao cálculo de probabilidades interessantes ao surgimento de novas evidências, podendo-se também tirar proveito da sua funcionalidade no processo de aprendizagem de RB. Este processo pode ocorrer de duas maneiras: estrutura gráfica e parâmetros numéricos; sendo que estes processos iniciam por meio dos dados incompletos na amostra da base de dados, onde a inferência é utilizada para estimar estes valores faltosos (LUNA, 2004).

Para a construção de uma Rede Bayesiana os sistemas utilizam representações gráficas de dependências probabilísticas. Essa representação permite manipular a incerteza com base em princípios matemáticos fundamentados, além de modelar o conhecimento do especialista do domínio de uma forma intuitiva.

Para isso são utilizadas algumas ferramentas de apoio a construção dessas redes, e entre essas, a *shell* Netica (Norsys)[17] (FLORES; PEROTTO; VICCARI, 2003) .

3. *Shell* Netica

Atualmente no mercado existem muitas ferramentas para a construção de RB, desde as gratuitas até as comerciais. Dentre estas ferramentas destaca-se a *shell* Netica, desenvolvida pela empresa *Norsys Software Corporation* de Vancouver, Canadá. Esta *shell* auxilia a criação de RB possibilitando a integração com outros ambientes e linguagens de programação incluindo-se o desenvolvimento da interface com o usuário.

O fabricante Norsys relata que por se basear na inferência bayesiana, essa *shell* permite por meio das entradas, a quantificação do resultado em porcentagem de todas as variáveis relacionadas no sistema, ou seja, determinando a probabilidade de algo acontecer de acordo com os valores iniciais.

Atualmente, o fabricante deste software oferece algumas API's que permitem a integração entre o Netica e as seguintes linguagens de programação: C; C++; Visual Basic e Java, destacando-se a Netica Java API utilizada no estudo de caso desta pesquisa.

As bibliotecas e funções disponibilizadas pela *shell* Netica para o processo de integração com um ambiente ou linguagem de programação são várias, dentre elas pode-se destacar a Netica Java API, disponibilizada para o ambiente de programação Java e utilizada no estudo de caso desta pesquisa.

O Netica Java *API* dispõe de uma biblioteca com classes e métodos escritos na linguagem C. Estes métodos possibilitam a integração entre a *shell* Netica e algumas linguagens de programação como por exemplo, o Java que por meio por exemplo, do ambiente o NetBeans IDE é utilizado para criar a interface do usuário, onde é possível gerenciar as opções presentes na versão application dessa *shell* (Norsys, 2006).

Dentre as classe e métodos que a Netica Java API se destacam algumas utilizadas neste estudo de caso.

A classe Environ é utilizada para criar um novo ambiente[18] para a RB a ser construída. Para a utilização desta classe é necessário declarar uma variável do tipo da classe e uma

[17] Netica Bayesian Network Software from Norsys. **Norsys Software Corp**. Disponível em: http://www.norsys.com.

variável do tipo *string* onde coloca-se a licença[19] de utilização da API que se faz necessária somente quando a RB possuir mais de 16 *nós*, caso contrário usa-se a definição *null* (NORSYS, 2006).

Segundo Norsys (2006), a classe *Net* é utilizada para criar uma nova RB. Esta classe possui vários construtores sobrecarregados que criam e executam a RB de forma diferente no ambiente atual. Entre os construtores existentes nesta classe destaca-se o *Net (Streamer)* utilizado para a leitura de um arquivo de RB. Na execução da leitura do arquivo, é retornado a referência da nova rede do arquivo ou uma exceção com a informação de que não foi possível ler o arquivo.

A classe Node representa o nós da RB, sendo assim, esta classe é utilizada para representar as ações que podem ser realizadas como a parte que trata da RB.

O método *getBelief()* é utilizado para retornar a probabilidade de cada estado de um determinado nó da RB.

Outro método disponibilizado na classe Node é o *enterFinding()* responsável pela propagação das probabilidades condicionais, dado uma nova evidência (NORSYS, 2006).

A classe NodeList contém um vetor utilizado para armazenar apenas os nós a partir de uma única rede. Se outros objetos forem inseridos, ou nós com nomes diferentes e de redes diferentes, uma exceção será lançada quando o NodeList for passado para a Netica (NORSYS, 2006).

Entre os métodos existentes na classe *NodeList* destaca-se o *getNode()*.

O fabricante Norsys (2006) define que o método *getNode()* é o responsável pela associação do *nós* declarados na RB com os *nós* existentes lidos de um arquivo.

O fabricante ainda disponibiliza um guia completo de instalação da *shell* e vários exemplos de aplicações, além de que o Netica Java API consegue executar todas as funções disponibilizadas no Netica C API.

Sendo assim, tendo-se como base os métodos disponibilizados na Netica Java API realizou-se a integração com o ambiente de programação NetBeans IDE 5.5.

4 . Estudo de Caso

Com o passar dos anos torna-se cada vez mais necessário que sistemas computacionais sejam desenvolvidos em ambientes e linguagens de programação para que possam ser executadas tanto em desktop, como na Web.

Esta pesquisa consiste na integração da *shell* Netica por meio da Netica Java API que auxilia a construção de Redes Bayesianas, com o ambiente de programação Java que por meio de *applets* permite que o sistema seja utilizado em várias plataformas e ser acessado pela internet.

Para atingir os objetivos propostos algumas etapas metodológicas foram seguidas e, dentre estas, a primeira indica o levantamento bibliográfico realizado por meio de livros, artigos, trabalhos de conclusão de curso, teses, dissertações, entre outros materiais científicos na área médica.

A problemática escolhida para o estudo de caso diz respeito a uma RB já desenvolvida na Universidade do Extremo Sul Catarinense em um trabalho de conclusão de curso denominado Sistema Especialista para o Apoio ao Diagnóstico de

---

[18] Espaço de memória alocado para a execução do gerenciamento da RB Norsys (2006).

[19] Atualmente a UNESC possui uma licença, para utilização da *shell* Netica e da API, com o fabricante Norsys onde uma licença de acesso ilimitado é disponibilizada.

Doenças Exantemáticas Maculopapulosas com Erupção Obrigatória (ANTUNES, 2002), (ANTUNES et al, 2004)...

Esta RB foi desenvolvida em 2002 pelo acadêmico Luciano Antunes do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, vinculado aos grupos de pesquisa em Inteligência Computacional Aplicada, e Informática Médica e Telemedicina tendo como especialista pelo médico e professor do curso de Medicina da UNESC Frank Traebert Júnior.

A escolha da utilização dessa RB se deu em virtude de oferecer uma atualização a este sistema, e considerando também sua utilidade para os acadêmicos do curso de Medicina da Universidade do Extremo Sul Catarinense no Laboratório de apoio ao ensino médico.

É importante ressaltar também conforme apresentado na justificativa desta pesquisa que muitos sistemas probabilísticos costumam ser pouco utilizados na prática em decorrência de tecnologia utilizada no desenvolvimento, tornando este sistema apenas uma fonte de referência com uma BC desatualizada.

Assim, busca-se com a utilização da RB do SEDDEM oferecer uma atualização de interface e probabilidades condicionais conforme necessidade do especialista e usuários a fim de que este sistema possa ser utilizado com maior freqüência no curso de Medicina da Universidade do Extremo Sul Catarinense e de outra instituições.

O SEDDEM escolhido para a realização da integração foi desenvolvido inicialmente no ambiente de programação C++ Builder[20], oferecendo uma interface desktop com as opções de inferência bayesiana e contando com um módulo de ajuda em texto livre.

Na atualização da interface realizou-se alguns ajustes de diagramação conforme ilustra a Figura 24, que foi desenvolvida no ambiente de programação NetBeans IDE 5.5.1, disponibilizando-se as informações deste sistema por meio de uma applet[21].

Este novo projeto de interface incluiu duas novas informações que não constavam na versão anterior, sendo o módulo de explicação desenvolvido por meio de árvores (mostrando cada opção que o usuário seleciona), sendo esta um componente essencial na arquitetura de um SEP e um gráfico em modelo de pizza para uma melhor visualização do usuário em relação as hipóteses diagnosticadas geradas pelo SEP.
Para o desenvolvimento do módulo de explicação utilizou-se o estudo e padronização realizado por Goulart (2007) que trata de um Trabalho de Conclusão de Curso do curso de Ciência da Computação da UNESC, que teve como objetivo implementar um agente assistente no sistema de apoio ao ensino e diagnóstico etiológico de lombalgia. Nesta pesquisa desenvolveu-se uma forma de representação do raciocínio por meio de árvores.

O módulo de ajuda foi reestruturado por meio de *links* html, facilitando a sua utilização com mais aplicabilidade na busca de ajuda que o usuário realiza.

Depois de realizar a atualização de interface com o usuário e inclusão do módulo de explicação passou-se à etapa da utilização da Netica Java API para a comunicação do ambiente NetBeans 5.5 com a RB desenvolvida na s*hell* Netica.

O ambiente para a utilização da RB do SEDDEM é referenciado da variável *env* é do tipo *Environ* e recebe *null*, logo depois esta variável *env* recebe uma referência ao novo ambiente com parâmetro *null* que representa não necessitar da licença.

A utilização da RB ocorre por meio de uma variável declarada do tipo Net com o nome net. Declarando-se a variável, realiza-se a abertura do arquivo por um

[20] Disponível em: http://www.borland.com/
[21] Um programa escrito em Java que possui uma página html (http://java.sun.com/applets/).

construtor da classe Net, onde a RB é aberta recebendo os parâmetros *Base.dne*(arquivo da RB), um nome a RB, neste caso SEDDEM, e o seu endereço de ambiente criado por meio da classe Environ, neste caso env.

Para realizar a inferência bayesiana é necessária a criação ou associação dos *nós* já existentes. Na integração do SEDDEM realizou-se a associação dos *nós* existentes com os *nós* endereçados na integração, sendo realizada por meio do método GetNode que associa o *nó* endereçado com o *nó* do arquivo

Com os *nós* e as respectivas probabilidades a *priori* já associada as variáveis, à próxima etapa da integração visa compilar a RB.

O método *compile()* da classe Net realiza a propagação das probabilidades a *priori*.

A propagação das probabilidades condicionais são realizadas a cada evidência escolhida pelo usuário. Essa lógica é realizada por meio do método *enterFinding* que propaga os valores na inferência conforme a evidência escolhida.

A finalização da RB é realizada por meio do método *finalize* da classe Net onde são liberados todos os recursos de alocação de memória utilizados durante o processo de execução do sistema.

Como resultado desta pesquisa foram oferecidas novas versões para os sistemas SEDDEM, PROBUCAL e BIOWOMAN com portabilidade de execução na internet.

Conforme a opinião do especialista, a nova versão do SEDDEM será melhor utilizada em suas aulas no curso de medicina da UNESC.

Sendo que o software desenvolvido se comportou conforme a expectativa do especialista.

Para atender esta característica foram desenvolvidas algumas applets.

Esta applet encontra-se no site do grupo de pesquisa em Informática Médica e Telemedicina denominado Kiron (http://www.kiron.unesc.net), que tem como objetivo por meio do desenvolvimento científico-tecnológico, desenvolver projetos de pesquisa nos quais são utilizados elementos dos domínios da Medicina e da Computação para suporte aos profissionais de saúde no atendimento ao público, além de criar um ambiente de pesquisa em Informática Médica na região sul do estado de Santa Catarina para gerar e difundir tecnologias aplicadas à área médica

Além do SEDDEM também foi possível realizar a integração de mais dois SEP da área da saúde já desenvolvidos na UNESC.

Estes sistemas foram desenvolvidos por meio dos mesmos critérios e padrões de implementação adotados no SEDDEM.

Realizou-se a integração do PROBUCAL e BIOWMAN descritos detalhadamente nos itens a seguir.

O PROBUCAL é um sistema especialista probabilístico, desenvolvido para prognosticar algumas doenças bucais como: cárie, gengivite, periodontite e câncer, além de verificar a possibilidade da inexistência das mesmas (ROSA, 2002), (ROSA et al, 2003).

Sua RB foi desenvolvida em 2002 pela acadêmica Lílian Regina Junkes da Rosa como trabalho de conclusão do curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, vinculado aos grupos de pesquisa em Inteligência Computacional Aplicada, e Informática Médica e Telemedicina tendo como especialista a odontóloga Maria Emília Moreira Wessler Philippi.

O SEP BIOWOMAN que objetiva auxiliar o profissional médico no diagnóstico de algumas doenças relacionadas à leucorréia, encontradas no trato genital

inferior da população feminina (RODRIGUES, 2002), (MACHADO, SIMÕES, MATTOS, 2006).

Esta RB foi desenvolvida em 2002 pela acadêmica Ana Carolina Braga Rodrigues como trabalho de conclusão do curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, vinculado aos grupos de pesquisa em Inteligência Computacional Aplicada, e Informática Médica e Telemedicina tendo como especialista a médica e professora do curso de Medicina da UNESC Sandra Manenti.

Sendo estes os resultado obtidos pode afirmar que a integração entre a Netica Java API e o ambiente de programação NetBeans IDE 5.5 foi de grande relevância para a conclusão desta pesquisa.

5 . Conclusão

Com a realização desta pesquisa é possível afirmar que a área de inteligência artificial na saúde é um campo que cresce a cada dia com novas pesquisas e novas descobertas em todo o mundo.

O objetivo principal desta pesquisa foi concluído pela integração da Netica Java API com o ambiente de programação NetBeans IDE 5.5, considerando-se que durante este processo foram compreendidos os principais recursos disponibilizados pela API da s*hell* Netica para o ambiente Java, que resultaram nas novas versões dos sistemas SEDDEM, PROBUCAL, BIOWOMAN.

Com o intuito de facilitar futuras pesquisas na área de conhecimento referente ao Netica Java API ofereceu-se uma documentação sobre a utilização desta API no ambiente de desenvolvimento NetBeans IDE 5.5.

Teve-se como limitação da pesquisa, que applet pode ser executada de forma satisfatória no navegador Internet Explorer em virtude de algumas configurações de segurança do demais navegadores. Ainda nesse contexto, para o correto funcionamento neste browser é necessário realizar o download dos arquivos NeticaJ.dll e Netica.dll necessários para a execução da API da *shell* Netica nas applets.

Referências

Antunes, Luciano. *SEDDEM* - Sistema Especialista para o Apoio ao Diagnóstico de Doenças Exantemáticas Maculopapulosas com Erupção Obrigatória. 2002. 76 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2002.

Antunes, Luciano et al. Apoio ao Diagnóstico de Doenças Exantemáticas Maculopapulosas por meio do Raciocínio Probabilístico. In: CONGRESSO BRASILEIRO DE INFORMÁTICA EM SAÚDE, 9., 2004, Criciúma. Anais eletrônicos... Criciúma: UNESC, 2004. Disponível em: http://www.sbis.org.br/cbis9/arquivos/845.pdf Acesso: 20 out. 2007.

Cozman, Fabio Gagliardi. JavaBayes. Disponível em: http://www.cs.cmu.edu/afs/cs/user/fgcozman/www/home.html. Acesso: 20 out. 2006

Flores, Cecília Dias; Perotto, Filipo; Viccari, Rosa Maria. Sistemas Baseados em Conhecimento para a Área da Saúde. Programa de Pós Graduação em Computação – Universidade Federal do Rio Grande do Sul. 2003, Porto Alegre.

GOULARTE, Jackson Pirola. **Agente Assistente no Sistema de Apoio ao Ensino e Diagnóstico Etiológico de Lombalgia.** 2007. 100f. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2007.

Hugin Expert. Disponível em: http://www.hugin.com/. Acesso: 07 dez. 2006.
LADEIRA, Marcelo; FLORES, Cecília Dias; VICCARI, Rosa Maria. Tratamento Eficiente da Incerteza em Sistemas de Apoio à Decisão. Departamento de Ciência da computação – Universidade de Brasília. 2002, Brasília

Ladeira, Marcelo et al. Ferramenta Aberta e Independente de Plataforma para Redes probabilísticas. Departamento de Ciência da computação – Universidade de Brasília. 2003, Brasília.

Luna, José Eduardo Ochoa. Algoritmo em para Aprendizagem de Redes Bayesianas a Partir de Dados Incompletos. 2004. 118 f. Dissertação – Departamento de Computação e Estatística, Universidade Federal de Mato Grosso do Sul, Campo Grande, Mato Grosso do Sul, 2004.

Marques, Isaac R.; Marin, Heimar F. Sistema de Apoio à Decisão em enfermagem.. Programa de Pós-Graduação em Ciências da Computação – Universidade Federal de Santa Catarina. 2002, Florianópolis

Nassar, Sílvia Modesto. Tratamento de Incerteza: Sistemas Especialistas Probabilísticos. Disponível em: http://www.inf.ufsc.br/~silvia/disciplinas/sep/MaterialDidatico.pdf: Acesso: 30 out. 2006.

Netica Bayesian Network Software from Norsys. Norsys Software Corp. Disponível em: http://www.norsys.com. Acesso: 10 out. 2006.

RODRIGUES, Ana Carolina Braga. **Biowoman** – Base de Conhecimento Dinâmica para Sistemas Especialistas Probabilísticos. 2002. Trabalho de Conclusão do curso (Graduação) - Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2002.

ROSA, Lilian Regina Junkes da . **Sistema Especialista Probabilístico Para Prognóstico De Doenças Bucais**. 2002. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense, Criciúma, Santa Catarina, 2002.

ROSA, Lilian Regina Junkes et al. **Aplicação do Raciocínio Probabilístico no Desenvolvimento de um Sistema Especialista para Apoio ao Prognóstico de Doenças Bucais**. Revista de Iniciação Científica (Criciúma), v. 1, p. 101-114, 2003.

Russell, Stuart J.; Norvig, Peter. Inteligência artificial. Rio de Janeiro: Elsevier, 2004

Saheki, André Hideaki. Construção de uma Rede Bayesiana Aplicada ao Diagnóstico de Doenças Cardíacas. 98 f. Dissertação – Título de Mestre em Engenharia, Escola Politécnica da Universidade de São Paulo, São Paulo, São Paulo, 2005.

Simões, Priscyla W. T. A. *SACI* - Sistema Especialista Probabilístico para Avaliação do Crescimento Infantil. Dissertação – Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, 2001.

Tibiriçá, Carlos A. G.; NASSAR, Sílvia M. Desenvolvimento de uma Abordagem Híbrida Difuso-Probabilística para Modelagem de Incertezas. Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, 2003.

Williamson, Jon. Bayesian nets and causality: philosophical and computational foundations. New York: Oxford University Press, 2005.