Implementação da Revisão Sistemática de Sintomas em Sistemas Móveis Utilizando Redes Bayesianas.

Alexandre B. José[3], Talles Marcelo G. de A. Barbosa[1, 3], Iwens G. Sene Jr[1, 3], Hervaldo S. Carvalho[2], Adson F. da Rocha[1], Francisco A. de O. Nascimento[1], Liana S. da S. Castro[1]

[1] Departamento de Engenharia Elétrica – Universidade de Brasília (UnB)
[2]Faculdade de Medicina – Universidade de Brasília (UnB)
Campus Universitário - Asa Norte - 70910-090 - Brasília - DF – Brasil

[3]Departamento de Computação – Universidade Católica de Goiás (UCG)
Av. Universitária 1.440 - Setor Universitário - 74605-010 - Goiânia - GO - Brasil

{abjucg@gmail.com}, {talles, iwens, carvalho, adson, assis}@unb.br,
{talles, iwens}@ucg.br, {liana_sena@yahoo.com.br}

Abstract: *The purpose of this work is to program a module capable of executing the Objective Symptoms Review using Bayesian Networks in mobile systems like cell phones. As result, is expected a more efficient system for the user's response time metric.*

Resumo: *O objetivo deste trabalho é implementar um módulo capaz de executar a Revisão dos Sistemas utilizando Redes Bayesianas em sistemas móveis tipo celular. Como resultado, busca-se um sistema mais eficiente em tempo de resposta ao usuário.*

1. Introdução

A evolução tecnológica relacionada aos dispositivos computacionais torna possível que vários desses sejam inseridos no espaço pessoal do usuário. Isso facilita o monitoramento humano, pois permite com que informações sobre o ambiente assim como sobre funcionamento do próprio corpo possam ser captadas a qualquer momento e em qualquer lugar [Billinghurst 2002] [Carvalho et al. 2003] [Devaul et al. 2003] [Mann 2001].

A Revisão dos Sistemas (RS) ou também conhecido Interrogatório Sintomatológico é um processo em que o médico levanta possibilidades e reconhece enfermidades que não guardam relação com o quadro sintomatológico registrado na História da Doença Atual (HDA). A única maneira de realizar uma boa RS é seguir um esquema rígido, constituído de um conjunto de perguntas que correspondem a todos os sintomas indicativos de alterações dos vários aparelhos do organismo [Porto 2001].

Como ferramenta computacional, a teoria de decisão Bayesiana é usada para gerar um modelo probabilístico, pois ela permite determinar probabilidades condicionais de uma evidencia a priori. Redes Bayesianas podem ser criadas para lidar com problemas que possuem algum nível de incerteza, desde o estabelecimento de

diagnósticos de doenças, seleção de tratamentos otimizados até na prescrição de tratamentos médicos [Pearl & Russel 2000] [Carneiro & Silva 1999].

Assim o uso de um *software* capaz de executar o Interrogatório Sintomatológico de uma maneira que não sobrecarregue o usuário do sistema com inúmeras perguntas, pode acarretar um ganho em termos de tempo de interação entre homem e sistema, e conseqüentemente diminuir o consumo de energia, ocasionando uma melhora na quantidade de tempo em que o sistema pode continuar em operação.

Contudo, o objetivo deste trabalho é verificar a possibilidade de implementar de um módulo capaz de executar a Revisão dos Sistemas utilizando Redes Bayesianas. Como resultado, busca-se um sistema mais eficiente em tempo de resposta ao usuário.

Na seção 2 será mostrada a técnica de coleta de evidências utilizada. A seção 3 descreve a maneira com a Rede Bayesiana foi modelada. Na seção 4 tem-se uma descrição das classes aqui apresentadas junto com uma aplicação real. E finalmente na seção 5 as conclusões obtidas na realização deste trabalho.

2. Coletando evidências através do caminhamento em grafos

Coletar evidências em uma Rede Bayesiana pode ser considerado como um caminhamento pelo grafo que representa a rede. Durante o trajeto é possível realizar a coleta de evidências que são constantemente confirmadas pelos usuários do sistema.

O caminhamento em profundidade foi adotado como metodologia para a realização deste trabalho.

2.1 Caminhamento em Profundidade

Tal técnica de caminhamento consiste em visitar todos os nós de um ramo até atingir os nós terminais, repetindo o processo em todos os ramos [Forbellone & Eberspächer 1998].

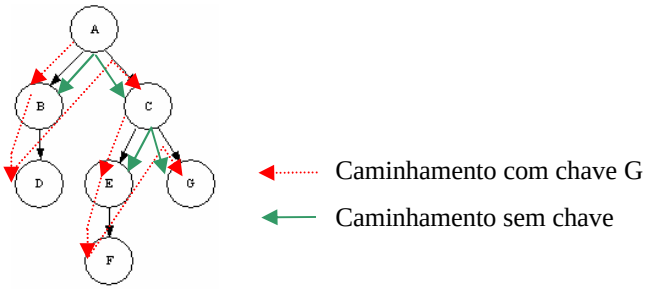

Figura 1 – Caminhamento em Profundidade

A técnica original de caminhamento em profundidade procura alguma chave de pesquisa pré-definida. Como o intuito das classes que serão aqui discutidas é permitir que redes que representem questionários, sejam transportadas para a execução em sistemas computacionais, tem-se a percepção de que, quando se realiza algum questionário, as chaves de buscas verdadeiras só são encontradas no final do conjunto de questões. Portanto é necessário que durante todo o percurso pelo grafo o usuário do sistema seja indagado sobre qual das próximas evidências condizem com a situação real.

Logo para o seu uso no grafo de interrogatórios faz-se necessária a execução de perguntas referentes aos nós imediatamente posteriores a pergunta anterior. Ex: Caso a pergunta C da Figura 1 tenha sido feita, será necessário realizar a pergunta referente aos nós E e G, pois o algoritmo original não prevê qualquer tipo de heurística para escolher a pergunta mais apropriada, ou seja, é dependente de uma escolha feita pelo usuário.

A complexidade computacional do algoritmo de caminhamento em profundidade, especificamente como discutido acima é O(n), pois o número de iterações do algoritmo fica restrito ao número de nós encontrados em seu trajeto, sendo que o grafo utilizado para o caminhamento não possui arestas cíclicas.

3. Ferramenta para a Modelagem de Redes Bayesianas: Microsoft Belief Networks (MBN)

O MBN é um *software* para a plataforma Windows que dá suporte a criação, manipulação e avaliação de Redes Bayesianas. O software representa a rede graficamente através de um diagrama, onde as variáveis são exibidas como elipses, os nós, e as dependências condicionais são exibidas como arcos entre as variáveis [Microsoft 2004]. A ferramenta permite que a estrutura gráfica gerada seja armazenada em um arquivo com a extensão .dsc, sendo este o primeiro formato utilizado pela ferramenta e mantido por questões de compatibilidade. Uma das desvantagens desse *software* é que o mesmo não gera o modelo em forma de estrutura de dados para nenhuma linguagem específica, usa um formato proprietário no arquivo .dsc, ou seja, um arquivo de texto com *tokens* específicos para a MBN. Sendo assim foi necessária a implementação de um pequeno tradutor que será descrito na subseção 4.1.

3.1 Modelando o Interrogatório Sintomatológico

O Interrogatório Sintomatológico é um conjunto de questões pertinentes à saúde do paciente, que tem como objetivo auxiliar o médico na realização da anamnese. As perguntas que compõem o interrogatório são organizadas de forma hierárquica, ou seja, permitem uma abordagem *top-down*, visão genérica (macro) e depois visão mais específica (micro), do problema vivido pelo paciente. O mais importante neste tipo de hierarquia é que a mesma pode ser facilmente representada sob a forma de um grafo.

A seguir tem-se um pequeno trecho do Interrogatório Sintomatológico, ilustrando como o mesmo pôde ser transformado em um grafo. O Interrogatório Sintomatológico utilizado foi o mesmo apresentado em [Barbosa et al. 2004].

```
Cabeça
   3.1.Onde fica? no rosto (inclui a testa e o ouvido) resto da
       cabeça
      3.1.1. Tem dor de cabeça?   Sim    Não
         3.1.1.1. Começou: de repente gradualmente
         3.1.1.2. Intensidade: fraca moderada intensa muito
                intensa
         ...
         3.1.1.3.exercício posição e/ou movimentos nenhuma
                alternativa anterior
         3.1.1.4. Está associado com: distúrbios visuais ânsia de
                vômito vômito
```

Figura 2 – Parte do Interrogatório Sintomatológico

Na Figura 2 observa-se a estrutura de tópicos do interrogatório, evidenciando sua natureza hierárquica, vale lembrar que os itens em sublinhado são as evidências que podem ou não ser observadas pelo paciente. Assim o grafo só representa as perguntas, sendo que as evidências ficam inseridas dentro de cada nó do grafo. Uma característica importante do interrogatório é que o mesmo só gera um antecessor para cada nó, ou seja, cada nó filho ou conjunto de nós filhos no mesmo nível irão possuir apenas um nó predecessor (pai). Garantindo assim um único trajeto possível até um nó folha.

4. Implementação das Classes

As classes componentes deste trabalho podem ser vistas na Figura 3. Observe que é possível criar sua própria técnica de caminhamento pelo grafo, ou seja, não é obrigatório utilizar nenhum objeto da classe `BProfundidade`, deixando assim o pesquisador livre para implementar o tipo de caminhamento que melhor se encaixa no seu objeto de estudo. No entanto tal fato requer que o programador passe então a ter conhecimento das interfaces das classes `GAD` e `No` para que possa tirar proveito de suas funcionalidades.

Surgem, portanto, dois tipos de abordagens no uso de tais classes. Na primeira a rede que representaria algum tipo de questionário pode ser implementada no formato dsc e então utilizada de maneira direta pelas classes. Outra opção seria o uso de um tipo diferente de caminhamento, que deverá ser implementado em outra classe, e conseqüentemente permitindo um novo controle na lógica de execução das perguntas.

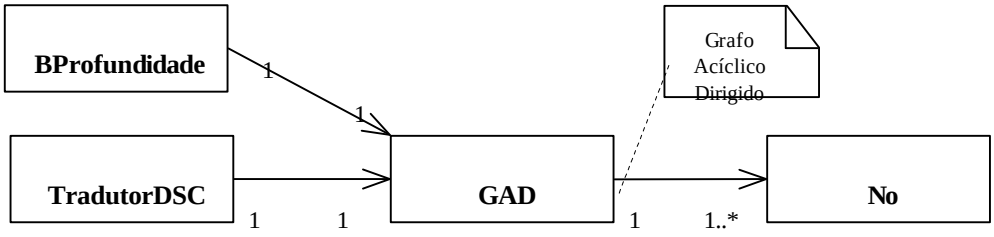

Figura 3 – Diagrama de Classes TradutorDSC e BProfundidade

4.1 O TradutorDSC

O tradutor criado é bastante simplificado, pois deverá ser portado para execução em ambientes com recursos computacionais restritos, como celulares e *PDA*.

A classe `TradutorDSC` assume que algumas características importantes devem existir na rede que se deseja traduzir: todo nó deve possuir um único nó antecessor (Pai); as matrizes de probabilidades no arquivo .dsc não podem ultrapassar o comprimento da linha, ou seja, não pode haver um matriz com quebra de linha; linhas Type, Position e Default são ignoradas pelo tradutor. Sendo que *Type* define a quantidade de evidências que podem ser observadas em um nó. O tradutor simplesmente começa a leitura das evidências quando encontra o *token* **`{`** e finaliza a leitura das mesmas quando encontra o *token* **`};`**.

Sempre que um novo objeto da classe `TradutorDSC` é instanciado, o mesmo se encarrega de instanciar um objeto da classe `GAD`, descrita na subseção 4.3. Iniciando então a leitura do arquivo .dsc, quando o tradutor encontra uma variável então é instanciado um objeto `No` que é em seguida associado ao objeto `GAD` criado anteriormente. Logo que as evidências de um determinado nó são encontradas, passam a

fazer parte das características do nó, assim como suas probabilidades. A Figura 4 ilustra a seqüências de ações executadas pelo tradutor.

O tradutor possui um único método chamado `ListarArquivo()` que é responsável por iniciar toda a execução da leitura do arquivo .dsc e montagem das estruturas de dados. Assim com o uso do `TradutorDSC` é possível portar a estrutura gerada pelo Microsoft Belief Networks para a linguagem *JAVA*.

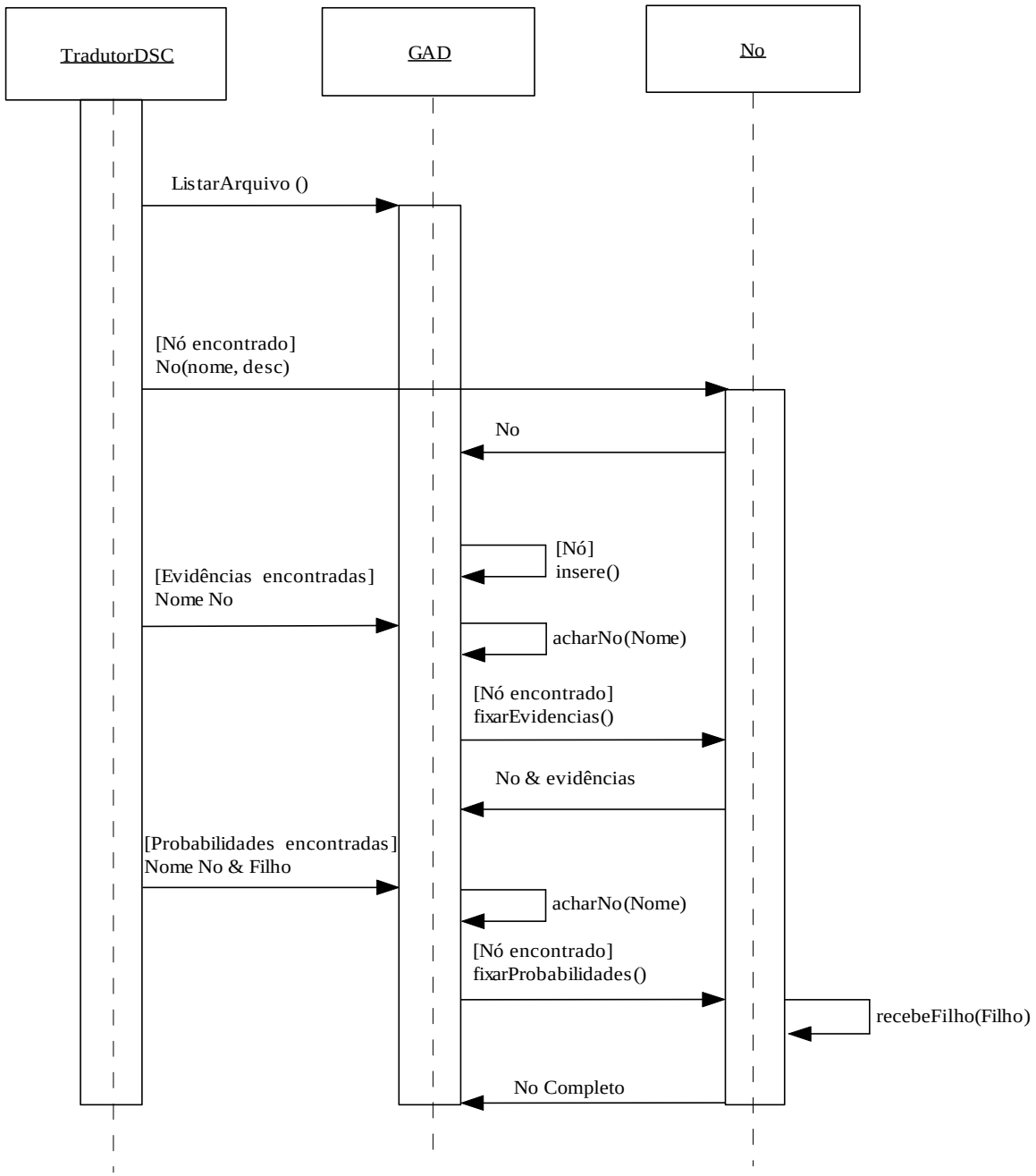

Figura 4 – Diagrama de Seqüência do TradutorDSC

4.2 A Classe No

A classe `No` é responsável por gerar todas as estruturas de dados e métodos para a criação e manutenção dos nós em memória. Sua descrição gráfica pode ser vista na Figura 5.

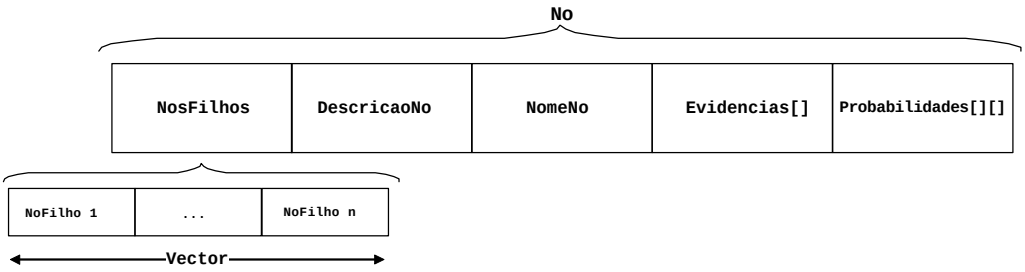

Figura 5 – Estrutura de Dados da Classe No

Toda a classe `No` pode ser vista na Figura 6 (versão para *J2ME*). O método de maior utilidade para o `TradutorDSC` é o `recebeFilho()`. Sendo que os demais métodos podem ser utilizados pelo programador para dar maior funcionalidade ao seu sistema. Por exemplo uma chamada há `noInstanciado.listarFilhos()` irá retornar um objeto `ChoiceGroup` que pode ser imediatamente associado a algum formulário para exibição na tela do dispositivo e conseqüentemente permitindo uma escolha de alguma evidência contida no nó. O mesmo raciocínio é valido para o método `imprimeEvidencias()`.

```
 1  import javax.microedition.lcdui.*;
 2  import java.util.*;
 3
 4  public class No
 5  {
 6      public Vector NosFilhos;
 7      public String DescricaoNo;
 8      public String NomeNo;
 9      public String Evidencias[];
10      public float Probabilidades[][];
11
12      public No(String Desc,String NomeN)
18      public ChoiceGroup imprimeEvidencias()
31      public int quantasEvidencias()
35      public int quantosFilhos()
39      public boolean temFilhos()
43      public void recebeFilho(No ref)
47      public void removeFilho(No ref)
51      public int ondeEsta(No ref)
55      public void fixarEvidencias(String evi[])
59      public void fixarProbabilidades(float p[][])
63      public ChoiceGroup listarFilhos()
80
81  }
```

Figura 6 – Especificação da Classe No

4.3 A Classe GAD

Responsável pela manutenção e criação de Grafos Acíclicos e Dirigidos na memória, é basicamente uma classe que assimila referencias para objetos do tipo `No`, criando assim uma coleção de nós que juntos formam um grafo, Figura 7.

É uma classe de apóio a execução do `TradutorDSC` com dois métodos de essencial importância: `Insere()` e `acharNo()`, permitindo que nós referentes a um

determinado `GAD` sejam inseridos e também pesquisados para garantir a localização correta dos ponteiros durante a execução dos algoritmos.

```
1  import java.util.*;
2
3  public class GAD
4  {
5      private Vector No;
6      private No x;
7
8      public GAD()
12     public void Insere(No ref)
16     public int ContaNos()
20     public void imprimeGAD()
29     public No acharNo(String chave)
44 }
```

Figura 7 – Especificação da Classe GAD

4.4 A classe BProfundidade

A classe que implementa o caminhamento em profundidade no grafo é defina de modo que possa operar sobre Redes Bayesianas que representem um conjunto de perguntas e respostas sobre algum assunto. Para este trabalho a rede utilizada representa o conjunto de questões referentes ao estado de saúde de pessoas, que está contido no Interrogatório Sintomatológico. A classe `BProfundidade` possui apenas um único método chamado `Buscar()` que inicia o questionamento, tal classe assume que já existe uma instância da classe `TradutorDSC`.

A Figura 8 mostra um exemplo de como a classe pode ser utilizada em outro domínio que não seja o do Interrogatório Sintomatológico. No caso foi utilizado um pequeno questionário sobre defeitos no *hardware* de um computador pessoal e seus periféricos.

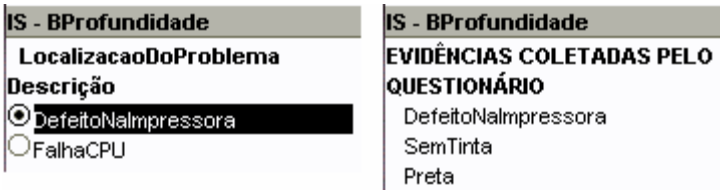

Figura 8 – Exemplo de uso da classe BProfundidade em outro domínio

4.5 O Protótipo

O protótipo possui uma versão para *desktop*, e uma versão para dispositivos móveis, como pode ser visto pela Figura 9. O uso do Interrogatório Sintomatológico como rede representativa de um questionário, confirma a utilidade das classes descritas acima na forma de uma aplicação real.

Os algoritmos são capazes de fornecer uma relação de todas as evidências coletadas durante a execução do Interrogatório Sintomatológico, como mostrado na Figura 9. Permitindo, por exemplo, que o conjunto de evidências coletadas (respostas) possam ser enviadas para o médico do paciente, ou outro sistema que se encarregue de cuidar de tais informações.

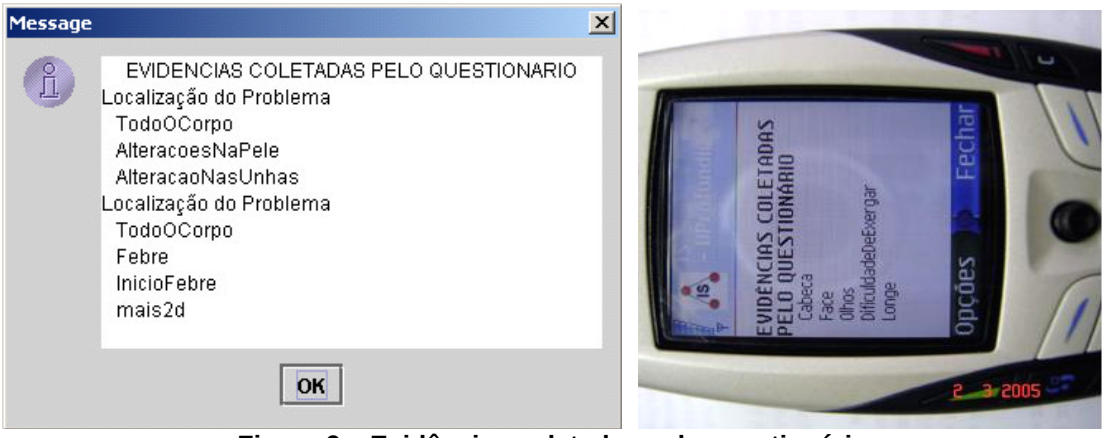

Figura 9 – Evidências coletadas pelo questionário

5. Conclusões

As classes definidas permitem uma abordagem estruturada para a construção de um sistema computacional encarregado de realizar questionamentos sobre determinado assunto. Viu-se que é possível separar a semântica dos questionários, através do uso de Redes Bayesianas, da implementação da lógica de execução das perguntas. O que pode resultar numa quantidade de linhas de código menor, sendo que se fosse utilizado um paradigma de programação convencional baseado em estruturas de seleção do tipo *IF..THEN..ELSE* ter-se-ia um aninhamento de comandos aproximadamente igual ao número de questões. O que não ocorre com a classe `BProfundidade`, que possui apenas treze comandos de seleção.

Para o usuário de tais sistemas, o algoritmo de caminhamento em profundidade, garante que não será preciso que o usuário tenha total visão do universo de questões sobre o assunto sendo avaliado. Diminuindo assim o tempo em que o usuário terá de ficar avaliando as possíveis opções.

Como trabalhos futuros, propõem-se o uso de uma nova técnica para caminhamento no grafo, que utilize métodos heurísticos para a seleção de um conjunto mais otimizado de perguntas, podendo acelerar o tempo de resposta do usuário ao sistema. O `TradutorDSC` deverá ser alterado para que o mesmo torne-se um tradutor genérico, sendo capaz de trabalhar com redes que possuam mais de um pai por filho. Permitindo assim o uso de Redes Bayesianas que não contemplem apenas modelos de questionários estruturados de forma hierárquica. Outra mudança importante é o suporte a outros formatos de arquivos além do .dsc, como por exemplo XML.

Agradecimentos

Aos alunos Érica Figueiredo e Henrique Branisso da Faculdade de Medicina da Universidade de Brasília (UnB) pelo apoio através da elaboração das perguntas referentes ao questionário para Revisão dos Sistemas.

Aos professores José Olímpio Ferreira e Olegário Correa da Silva Neto do Departamento de Computação da Universidade Católica de Goiás (UCG) pelas discussões e sugestões referentes aos algoritmos para caminhamento em grafos.

Referências

Barbosa, M. G. A. Talles; SENE, Jr. G. Iwens; CASTRO, S. S. Liana; BRANISSO, J. P. Henrique; FIGUEREDO, C. Érika; CARVALHO, S. Hervaldo; ROCHA, F. Adson; NASCIMENTO, A. O. Francisco. “Sistema Pessoal Móvel de Monitoração da Saúde: Algoritmo para Captura Inteligente de Sintomas”, CBIS, 2004.

Billinghurst, Mark. Wearable Appliances; “The Future of Wearable Computing”, Human Interface Technology Laboratory, University of Washington, 2002.

Carneiro, Alexandre L; Silva, Wagner T. “Introdução a redes Bayesianas”, Relatório de Pesquisa CIC/UnB – 09/99.

Carvalho H., Heinzelman W., Murphy A. and Coelho C. “A General Data Fusion Architecture”. Proceedings of the 6th International Conference on Information Fusion (Fusion 2003), June 2003.

Devaul, Rich; Sung, Michael; Gips, Jonathan; Pentland, Alex “Sandy”. “MIThril 2003: Applications and Architecture”, Media Laboratory, Massachusetts Institute of Technology, 2003.

Forbellone, André L. V.; Eberspächer, Henri F., “Lógica de Programação – A Construção de Algoritmos e Estruturas de Dados”, Ed. Makron Books, 1998.

Mann, Steve. “Wearable Computing: Toward Humanistic Intellingence”, IEEE Intelligent Systems, University of Toronto, 2001.

Microsoft Research, http://research.microsoft.com/adapt/MSBNx/, novembro 2004.

Pearl, J; Russel, S. “Bayesian Networks”, Technical Report, R-277, Handbook of Brain Theory and Neural Networks, MIT Press, Novembro 2000.

Porto, Celmo C. “Semiologia Médica”, Editora Guanabara Koogan, Terceira Edição, 2001.