Marcos Oliveira Junqueira

MobileX-DAVIS: um Sistema Especialista Móvel para o Auxílio ao Diagnóstico de Pacientes com Doenças nas Vias Aéreas Superiores e Inferiores

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS
MINAS GERAIS - BRASIL
2006

Marcos Oliveira Junqueira

MobileX-DAVIS: um Sistema Especialista Móvel para o Auxílio ao Diagnóstico de Pacientes com Doenças nas Vias Aéreas Superiores e Inferiores

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de Concentração:
Inteligência Artificial

Orientador:
Prof. Heitor Augustus Xavier Costa

LAVRAS
MINAS GERAIS – BRASIL
2006

Ficha Catalográfica preparada pela Divisão de Processos Técnico da Biblioteca Central da UFLA

Junqueira, Marcos Oliveira

MobileX-DAVIS: um Sistema Especialista Móvel para o Auxílio ao Diagnóstico de Pacientes com Doenças nas Vias Aéreas Superiores e Inferiores /Marcos Oliveira Junqueira. Lavras – Minas Gerais, 2006. 71p : il.

Monografia de Graduação –Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. Informática. 2. Inteligência Artificial. 3. Dispositivos Móveis. I. JUNQUEIRA, M. O. II. Universidade Federal de Lavras. III. MobileX-DAVIS: um Sistema Especialista Móvel para o Auxílio ao Diagnóstico de Pacientes com Doenças nas Vias Aéreas Superiores e Inferiores.

Marcos Oliveira Junqueira

MobileX-DAVIS: um Sistema Especialista Móvel para o Auxílio ao Diagnóstico de Pacientes com Doenças nas Vias Aéreas Superiores e Inferiores

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 22/09/2006

_____

Prof. Luiz Henrique Andrade Correia

_____

Prof. Sérgio Martins de Souza

_____

Prof. Heitor Augustus Xavier Costa
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL

“A diferença fundamental entre o homem comum e o guerreiro,
é que o guerreiro encara tudo como desafio, enquanto o
homem comum encara tudo como bênção ou maldição.”

Carlos Castaneda.

Agradeço a todos os parentes, amigos, colegas e professores que me ajudaram nessa caminhada. Em especial à Jussara Marques Ferreira e Tatiane Stevan.

RESUMO

Neste trabalho, é proposto um Sistema Especialista Móvel que pode auxiliar no diagnóstico de doenças das vias aéreas superiores e inferiores, o MobileX-DAVIS. O sistema possui uma máquina de inferência que utiliza o método de Mamdani. Para que o sistema possua mobilidade, é implementado utilizando tecnologias como *WebServices* e *Java ME* para *Palm*. A resposta do sistema é um conjunto *fuzzy*, ou seja, o conjunto nem sempre responde qual doença o paciente possui, mas fornece a distribuição de possibilidades do paciente no conjunto de doenças. O Sistema pode ser expandido de maneira que possa atender a outras classes de doenças e até mesmos outros tipos de problemas que não estejam relacionados à área médica.

ABSTRACT

In this work, it is proposed a Mobile Specialist System that can aid in the diagnosis of diseases of the superior and inferior aerial ways, the MobileX-DAVIS. The system have an inference machine that uses the method of Mamdani. For the system to have mobility, it is implemented using technologies like WebServices and Java ME for Palm. The answer of the system is a group fuzzy, in other words, the group not always answers which disease the patient have, but supplies the distribution of the patient's possibilities in the group of diseases. The System can be expanded so that he can assist to other classes of diseases and to same other types of problems that are not related to the medical area.

Sumário

Resumo....i
Abstract....i
Lista de Figuras....iv
Lista de Tabelas....vi
1 Introdução....1
1.1 Motivação....2
1.2 Objetivo....2
1.3 Estrutura do Trabalho....3
2 J2ME....5
2.1 Considerações Iniciais....5
2.2 Configurações....7
2.2.1 *Connected Limited Device Configuration* (CLDC)....8
2.2.2 *Connected Device Configuration* (CDC)....9
2.3 *Profiles*....11
2.4 Pacotes Adicionais....12
2.5 Considerações Finais....13
3 *WebServices*....14
3.1 Considerações Iniciais....14
3.2 Arquitetura....15
3.3 XML-RPC....16
3.4 SOAP....17
3.5 WDSL....20
3.6 UDDI....22
3.7 Considerações Finais....22
4 Sistemas especialistas....24
4.1 Considerações Iniciais....24
4.2 Estrutura de um Sistema Especialista....25
4.2.1 Base de Conhecimento....25
4.2.2 Máquina de Inferência....25
4.2.3 Interface com o Usuário....27
4.3 Representação do Conhecimento....28
4.4 Técnicas Utilizadas no Processo de Inferência....29
4.4.1 Busca em Largura....29
4.4.2 Busca em Profundidade....30
4.4.3 Heurísticas....31
4.4.4 *Backtracking*....32
4.5 Máquina de Inferência - Método de Mamdani....33
4.6 Classificação dos Sistemas Especialistas....34
4.7 Alguns Sistemas....34
4.7.1 MYCIN....34
4.7.2 NIACIN....36
4.7.3 Sistemas Especialistas em Cardiologia - SEC....38
4.8 Considerações Finais....40
5 Sistema MobileX-DAVIS....41
5.1 Considerações Iniciais....41

5.2 Arquitetura....41
5.3 A Base de Regras do MobileX-DAVIS....43
5.4 Implementação da máquina de inferência....45
5.5 Modelagem UML....47
5.5.1 Módulo *WebService*....47
5.5.2 Módulo Cliente....49
5.6 Apresentação do MobileX-DAVIS....50
5.6.1 Configurando a URL do *WebService*....52
5.6.2 Inserir Paciente....52
5.6.3 Efetuando Diagnóstico do Paciente....53
5.6.4 Consultar Histórico do Paciente....54
5.7 Considerações Finais....55
6 Considerações Finais....56
6.1 Conclusões....56
6.2 Contribuições....56
6.3 Trabalhos Futuros....57
Referências Bibliográficas....58

LISTA DE FIGURAS

Figura 2.1: Hierarquia da Arquitetura. As camadas *Foundation Profile* (FP), *Personal Profile* (PP), *Personal Basis Profile* (PBP) e *Mobile Information Device Profile* (MIDP) representam modelos de implementações da camada profile da arquitetura. A *Connected Limited Device Configuration* (CLDC) e a *Connected Device Configuration* (CLDC) representando as diferentes configurações entre os dispositivos de pequeno porte. E os pacotes adicionais que oferecem funcionalidades variadas ao Java ME (Fonte: [SUN, 2005 a])....6
Figura 3.1: Requisição XML-RPC. (Fonte: [CERAMI, 2002])....17
Figura 3.2: Resposta XML-RPC. (Fonte: [CERAMI, 2002])....17
Figura 3.3: Estrutura de um envelope SOAP (Fonte: [TIDWELL et al, 2001])....18
Figura 3.4: Estrutura de uma mensagem contendo uma chamada RPC. (Fonte: [CERAMI, 2002])....19
Figura 3.5: Estrutura de uma mensagem contendo um documento. (Fonte: [CERAMI, 2002])....19
Figura 3.6: Estrutura de um arquivo WDSL. (Fonte: [CERAMI, 2002])....20
Figura 3.7: Tipos de operações suportadas pelo *WebServices* e seus respectivos passos. (Fonte: [CERAMI, 2002])....21
Figura 4.1: Exemplo de *forward-chaining*. (Fonte: [LUGER, 1998])....26
Figura 4.2: Exemplo de *backward-chaining*. (Fonte: [LUGER, 1998])....27
Figura 4.3: Árvore de estados com nós em azul representando a seqüência da busca em largura e em amarelo a produção inicial....30
Figura 4.4: Árvore de estados com nós em azul representando a seqüência da busca em profundidade....31
Figura 4.5: Processo de *Backtracking*. Nós contendo ramificações são marcados para possíveis retrocessos na árvore....32
Figura 4.6: Método de Mamdani com composição *max-min* (Fonte: [LOPES et al, 2005])....33
Figura 4.7: Função de pertinência (Fonte: [LOPES et al, 2005])....33
Figura 5.1: Arquitetura de sistemas baseados em regras *fuzzy* (Fonte: [LOPES et al, 2005]).....42
Figura 5.2: Arquitetura do sistema MobileX-DAVIS....43
Figura 5.3: Métodos diagnóstico e possibilidade que atuam como máquina de inferência. 46
Figura 5.4: Diagrama de casos de uso do sistema....47
Figura 5.5: Diagrama de classes para o módulo *WebService*....48
Figura 5.6: Diagrama de seqüência para o módulo *WebService*....49
Figura 5.7: Diagrama de classes para o módulo cliente....50
Figura 5.8: Diagrama de seqüência para o módulo cliente....50
Figura 5.9: MobileX-DAVIS já instalado no *Palm*....51
Figura 5.10: Tela inicial do MobileX-DAVIS....51
Figura 5.11: Alterando a URL do *WebService*....52
Figura 5.12: Inserindo paciente....52
Figura 5.13: Paciente inserido....52
Figura 5.14: Informando código do paciente....53
Figura 5.15: O paciente é exibido na tela....53
Figura 5.16: Informando grau dos sinais e dos sintomas....54

Figura 5.17: Sugestão de diagnóstico do sistema.....54
Figura 5.18: Histórico do paciente Joãozinho.....55

LISTA DE TABELAS

Tabela 5.1: Relação *fuzzy* sintomas x doenças (Fonte: [LOPES et al, 2005])..... 45
Tabela 5.2: Relação *fuzzy* pacientes x sintomas (Fonte: [LOPES et al, 2005])..... 45

1 INTRODUÇÃO

Este trabalho tem por finalidade a construção de um sistema especialista móvel que simula a atuação de um médico no diagnóstico de pacientes que apresentam sinais e sintomas de doenças das vias aéreas superiores e inferiores[1] (Pneumonia, bronquite, rinite, sinusite, gripe, laringite e amigdalite), a partir de informações dadas por especialistas da área médica.

O diagnóstico médico pode ser uma tarefa complicada, de certa maneira, ela é uma comparação, onde o médico confronta os dados reunidos com as informações disponíveis a respeito das diversas doenças existentes.

Durante a consulta médica, o médico deve transmitir ao paciente conforto e confiança, o que não depende do sistema especialista, tornando-se responsabilidade única e exclusiva do médico.

Um sistema móvel se torna muito interessante, pois a mobilidade é uma característica inerente ao ambiente médico. Pode ser necessário ao médico locomover-se até a residência do paciente para realizar a consulta. Porém, essa mobilidade é restringida pelo alcance das redes sem fio atuais.

Em um Sistema Especialista, tipicamente concentram-se as atividades de aquisição de conhecimento em domínios muito estreitos. A lista de aplicações em potencial para os Sistemas Especialistas é muito longa, mas podem ser enquadradas em três categorias [BITTENCOURT, 2001] :

i. Manufatura;
ii. Finanças;
iii. Serviços (educação, engenharia, meteorologia, medicina, militar, telecomunicações, etc.).

A excelência em qualquer uma destas categorias depende de fatores ligados à natureza do conhecimento envolvido, sua confiabilidade, sua integridade, sua ambigüidade e sua estabilidade. Em uma situação ideal, é interessante ter uma aplicação na qual o conhecimento seja restrito a um domínio muito estreito e as decisões sejam determinadas inteiramente por fatores mensuráveis que não mudam com o tempo. Dessa forma, os Sistemas Especialistas de elaboração mais fácil são aqueles em que as regras são

1 Faringe, laringe, epiglote, traquéia, bronquíolo, brônquio, alvéolos [MERCK, 2006].

conhecidas antecipadamente e são determináveis. O raciocínio deste ideal é quanto mais sólido for o entendimento do domínio do especialista, mais confiáveis serão as decisões do sistema.

No mundo real, entretanto, quase todo especialista humano toma decisões baseadas em uma situação em que os dados são, de certa forma, incompletos, não confiáveis, ambíguos e dinâmicos. É para esse tipo de situação que os Sistemas Especialistas são aplicáveis.

Entretanto, por melhor que seja a maneira como um Sistema Especialista lida com a incerteza em sua base de conhecimentos, esses sistemas ainda são fundamentalmente dedutivos e, desse modo, deixam muito a desejar quando se trata de raciocinar de formas não dedutivas. Por essa razão, o domínio de conhecimento ideal a ser modelado deve envolver pouca ou nenhuma porção de intuição e bom senso.

1.1 Motivação

Tipicamente, os problemas que podem ser solucionados por um Sistema Especialista são do tipo que seriam atendidos por um profissional especialista no domínio do problema, que normalmente é bem específico, como por exemplo, diagnosticar doenças das vias aéreas superiores e inferiores.

Através do conhecimento de um sistema para atendimento hospitalar existente, porém de difícil manutenção, surgiu a idéia de desenvolver um sistema especialista. Este sistema, tendo como entrada o grau dos sintomas ou sinais de um paciente com alguma complicação nas vias aéreas superiores ou inferiores, retorna uma tupla[2] com a possibilidade de cada uma das doenças estar infligindo o paciente.

Sendo uma das prioridades do Governo Federal acabar com as filas no serviço público, sobretudo nos hospitais, postos de saúde e da Previdência Social, este sistema especialista visa agilizar o atendimento de um paciente, através da automação do diagnóstico, agilizando o processo de atendimento e diminuindo as filas e o tempo de espera.

1.2 Objetivo

O objetivo deste trabalho é permitir, através das tecnologias de computação móvel, o diagnóstico de pacientes em locais onde o atendimento é lento, o serviço de saúde não

2 Série de valores separados por vírgulas e entre parênteses [ESPERANÇA, 2006].

dispõe de recursos humanos qualificados, que dispõem somente de um clínico geral e não de especialistas, que por alguma eventualidade o médico especialista não esteja presente.

Assim, o trabalho visa levar, mesmo que virtualmente, o conhecimento de um profissional médico aos locais onde este se faz necessário.

Como em muitos municípios do interior a falta de médicos especialistas é uma realidade, os enfermeiros ou os agentes de saúde, responsáveis pela população, não dispõem do conhecimento adequado para diagnosticar e prescrever medicações de forma correta. Isso pode colocar o paciente em um estado mais agravante. Em muitos casos, isso ocorre devido a não eficácia de um medicamento ou a uma falha no diagnóstico.

Além disso, devido à não presença do médico no local e, em alguns casos, à falta de acesso aos dados do paciente, o presente trabalho visa o desenvolvimento de um sistema especialista para auxiliar no diagnóstico do paciente. Este será responsável em auxiliar o médico na análise dos sintomas e definição da doença.

Tendo como alvo a rede pública de saúde, a diminuição dos custos faz parte do objetivo, assim, a implementação da proposta deste trabalho é baseada em software livre, utilizando tecnologia Java, na área de mobilidade do atendimento e na concepção do sistema especialista.

1.3 Estrutura do Trabalho

O capítulo 2 apresenta a Plataforma J2ME e as suas configurações CDC e CLDC, os *profiles Foundation Profile*, *Personal Basis Profile*, *Personal Profile* e *Mobile Information Device Profile* e os pacotes adicionais.

O capítulo 3 explica o que são os *WebServices*, apresentando sua arquitetura e os tipos XML-RPC, SOAP, WDSL e UDDI.

O capítulo 4 discute sobre sistemas especialistas. Inicialmente, é apresentada a estrutura de um sistema especialista: Base de Conhecimento, Máquina de Inferência e Interface com o Usuário. Além disso, esse capítulo apresenta a Representação do Conhecimento utilizando Lógica e Redes Semânticas, bem como técnicas utilizadas no processo de inferência, a saber: Busca em Largura, Busca em Profundidade, Heurísticas e *Backtracking*. Também é apresentado o método de Mamdani que, no MobileX-DAVIS, atua como máquina de inferência. Ainda neste capítulo, são apresentadas as classificações para os sistemas especialistas e alguns sistemas especialistas relacionados com o tema.

O capítulo 5 apresenta o funcionamento do MobileX-DAVIS, descrevendo a sua arquitetura, base de regras e implementação da máquina de inferência a partir do método de Mamdani. Além disso, é apresentada a modelagem do módulo *WebService* e do módulo cliente. Por fim é apresentado o módulo cliente em execução.

O capítulo 6 discorre sobre as conclusões e as contribuições do trabalho e sugere alguns trabalhos futuros.

2 J2ME

2.1 Considerações Iniciais

A plataforma Java criada pela Sun Microsystems define um ambiente de software independente da plataforma do hardware. Visando a portabilidade das aplicações, as plataformas especializadas foram desenvolvidas para atender desde aplicações *desktop* (*J2SE – Java 2 Standard Edition*) até aplicações distribuídas e de grande porte (J2EE – *Java 2 Enterprise Edition*), incluindo aplicações para micro dispositivos (J2ME – *Java 2 Micro Edition*) [SUN, 2006 c].

A plataforma Java™ Micro Edição (*Java ME*) é a plataforma de aplicação mais difundida para dispositivos móveis, pois fornece um ambiente robusto e flexível para as aplicações que funcionam em dispositivos portáteis, tais como celulares, *PDAs (Personal Digital Assistants*), decodificadores de TV e impressoras. A plataforma Java ME inclui interfaces de usuário flexíveis, um modelo de segurança robusto, grande quantidade de protocolos de rede internos e suporte extensivo para aplicações em rede e *offline* que podem ser carregadas dinamicamente. As aplicações baseadas nas especificações Java ME são escritas uma vez para vários dispositivos, contudo exploram potencialidades nativas de cada dispositivo [SUN, 2006 d].

Assim como *Java EE* e *Java SE*, *Java ME* inclui uma Máquina Virtual Java (*Java Virtual Machine - JVM*) e um conjunto padrão de Interfaces de Programação de Aplicativos (*Application Programming Interfaces - APIs*[3]*) Java* definido pela *Java Community Process* (JCP), por grupos de especialista cujos membros incluem os principais fabricantes de dispositivos, fabricantes de software e provedores de serviço [SUN, 2006 d].

Devido às diversas categorias de dispositivos, cada qual atendendo a um mercado vertical, a dificuldade de modelar uma única arquitetura que atenda todas as classes de dispositivos levou a *Java ME* a ser uma coleção de especificações ao invés de uma única e rígida plataforma. A especificação principal da plataforma *Java ME* define as estruturas básicas necessárias para a verticalização nas áreas de atuação de cada dispositivo. Assim como *Java EE* se adapta a servidores e *Java SE* a *desktops*, *Java ME* se adapta aos

3 Interface de programação de aplicativos que permite a comunicação entre programas ou entre um programa e o kernel (de um sistema operacional), estabelecendo as convenções e os parâmetros a serem seguidos [CERTISIGN, 2006].

diferentes tipos de dispositivos (decodificadores de sinal de TV, celulares e *PDAs*). Porém não se pode importar aplicações completas entre um ambiente e outro.

Visando atender a necessidade de verticalização, a arquitetura da plataforma é modelada baseada no padrão de camadas. Esta modularização permite fácil manutenção de seus componentes sem grandes impactos nas outras camadas. São definidas três camadas hierarquizadas nesta ordem: *configuration*, *profiles* e *optional packages*.

Cada camada da arquitetura é composta por especificações que definem modelos de implementações, onde cada modelo visa atender requisitos de uma determinada categoria de dispositivos baseado em seus recursos. Assim, existem hoje dois modelos de arquitetura que levam o nome da implementação da camada mais inferior (*configuration*) [SUN, 2005]. Esta camada define a abstração necessária do sistema nativo do dispositivo e o ambiente para o desenvolvimento de aplicações, como observado na Figura 2.1.

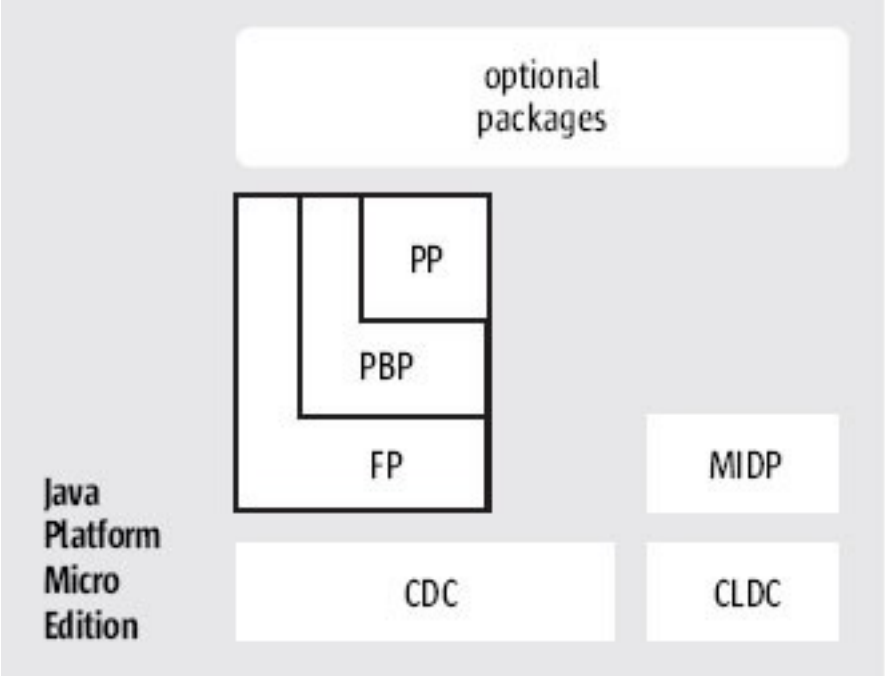

Figura 2.1: Hierarquia da Arquitetura. As camadas Foundation Profile (FP), Personal Profile (PP), Personal Basis Profile (PBP) e Mobile Information Device Profile (MIDP) representam modelos de implementações da camada profile da arquitetura. A Connected Limited Device Configuration (CLDC) e a Connected Device Configuration (CLDC) representando as diferentes configurações entre os dispositivos de pequeno porte. E os pacotes adicionais que oferecem funcionalidades variadas ao Java ME (Fonte: [SUN, 2005 a]).

2.2 Configurações

Uma configuração é a especificação que define o ambiente de software baseada em características comuns de uma gama de dispositivos como [TOPLEY, 2002]:

- Tipo e quantidade de memórias disponíveis;
- Tipo e velocidade de processamento;
- Tipo de conectividade disponível.

A especificação da camada configuração define apenas os requisitos mínimos da plataforma e, segundo a especificação do J2ME [SUN, 2005], deve ser totalmente implementada nos dispositivos de forma a oferecer um ambiente consistente. A função básica desta camada é prover uma abstração entre o ambiente de desenvolvimento e o sistema operacional.

Basicamente, uma configuração consiste de uma máquina virtual (JVM) e uma coleção de classes. Devido às particularidades e às limitações de cada dispositivo, existe a impossibilidade de migrar os aspectos e a funcionalidade da plataforma padrão *Java SE*. Assim, apenas um subconjunto da API (*Application Programming Interface*) original é importado.

A arquitetura J2ME define atualmente dois tipos de Configurações [SUN, 2005]:

- CLDC (*Connected Limited Device Configuration*): É a menor das duas configurações. Projetada para dispositivos com conexões de rede intermitentes, pouca memória e baixo nível de processamento. Dentre os dispositivos suportados por esta especificação, estão os celulares e os *PDAs* com recursos limitados (*Palms*, etc);
- CDC (*Connected Device Configuration*): Voltada para dispositivos mais robustos, com maior capacidade de processamento e armazenamento de dados. Possui uma máquina virtual e uma API mais sofisticada que as da CLDC e com mais similaridades à plataforma padrão *Java SE*. Entre os dispositivos suportados para esta especificação estão os *PDAs* (*Pocket PCs*) e *smart-phones.*

2.2.1 *Connected Limited Device Configuration* (CLDC)

A *Connected Limited Device Configuration* (CLDC) é a menor implementação da camada de configuração. Devido à isto, sua especificação abrange uma vasta gama de dispositivos. As restrições de hardware definem que o dispositivo deve ter pelo menos 160 KB de memória não volátil disponível para a máquina virtual e bibliotecas enquanto que é necessário 32 KB de memória volátil para execução da máquina virtual [SUN, 2006 b]. Além destas restrições, a especificação CLDC necessita que exista um sistema operacional responsável pelo gerenciamento do hardware e que este sistema forneça mecanismos para executar a JVM (*Java Virtual Machine*) que neste modelo recebe o nome de KVM (*Kilobyte Virtual Machine*), devido às modificações necessárias para implementação em micro dispositivos.

O modelo de aplicação definido pela CLDC se baseia no modelo padrão da arquitetura *Java SE*, onde uma classe deve conter o método *main* como sendo *public*, *static* e *void*, recebendo como argumento um *array* do tipo *String*. Este modelo de aplicação pode ser substituído por outros que podem ser implementados na camada logo acima chamada *Profiles* que será abordada adiante.

A especificação da CLDC não define a forma como as aplicações são gerenciadas pelo dispositivo, ou seja, ela é responsável apenas pela execução e pela liberação dos recursos após a execução. Assim, a forma de gerenciar as aplicações é de responsabilidade da implementação da camada, levando em consideração aspectos como: *download* (quando disponível), instalação, pesquisa de aplicações existentes no dispositivo e a seleção e a execução de uma determinada aplicação.

As questões sobre segurança são abordadas na especificação e nos níveis que são definidos de forma a garantir políticas refinadas. Assim, o modelo é estruturado em dois níveis de segurança [SUN, 2006 b], sendo estes:

- Baixo Nível (Máquina Virtual): Provê mecanismos para que classes mal formadas ou códigos maliciosos não interrompam a execução de maneira a danificar o dispositivo. Além disso, aplicações são impossibilitadas de sobrescrever, alterar ou apagar qualquer classe dos pacotes protegidos (biblioteca padrão) e de carregar classes que não estejam no seu próprio JAR[4]

4 JAR ou Java *ARchive* é um arquivo compactado usado para distribuir um conjunto de classes Java. É usado para armazenar classes compiladas e meta dados associados que podem constituir um programa

assegurando que uma aplicação não viole dados de outra aplicação;

- Nível de Aplicação: Define políticas de restrições onde apenas os recursos e as bibliotecas pré-estabelecidos podem ser acessados pela aplicação.

Com relação às classes oferecidas na biblioteca, a especificação define dois tipos [SUN, 2006 b]:

- Classes que pertencem à biblioteca *Java SE*;
- Classes específicas da CLDC, mas que são mapeadas em classes da *Java SE*.

Esta divisão ocorre pelo fato de muitas funções da plataforma padrão possuírem grande dependência do tipo de ambiente (*hardware*). Assim, algumas classes não foram incorporadas e outras foram modificadas de forma a atenderem aos requisitos da plataforma *Java ME*. A especificação restringe algumas alterações na implementação das classes importadas da Java SE no que diz respeito às assinaturas ou à inclusão de novos métodos e aspectos relacionados à semântica.

Além dessas características, a especificação define uma de conexão genérica visando atender às necessidades de interligação entre dispositivos por meio de alguma rede e acesso a periféricos ou sistemas de arquivos.

Esta estrutura é, basicamente, uma generalização dos pacotes *java.net* e *java.io* da plataforma padrão. Sua arquitetura é similar à arquitetura dos *drivers* de dispositivos utilizados em *desktops* e servidores. A classe *javax.microedition.io.Connector* define o meio de acesso aos protocolos de conexão e de entrada e saída, sendo a sua chamada baseada na especificação URI definida pela RFC-2396 [SUN, 2006 b].

Apesar de definir os mecanismos de acesso, a especificação não define implementações de protocolos, deixando este trabalho para a implementação dos *profiles*. Esta solução minimiza os impactos quando os protocolos são alterados, visto que as classes que implementam os protocolos são *linkadas* em tempo de execução.

2.2.2 Connected Device Configuration (CDC)

A *Connected Device Configuration* (CDC) foi projetada em torno de dois objetivos da *Java SE, a* compatibilidade e o suporte para diversos dispositivos. A compatibilidade da *Java SE* permite que os desenvolvedores aumentem seus investimentos na tecnologia *Java SE*, incluindo bibliotecas, ferramentas e técnicas. O suporte para dispositivos de recursos

[WIKIPÉDIA, 2006 c].

limitados permite que os vendedores de dispositivos ofereçam um ambiente de execução *Java* que possa suportar aplicações seguras, móveis e empresariais [SUN, 2005 a].

A CDC suporta a especificação completa da JVM, incluindo carregamento completo de classes e características das bibliotecas do núcleo. No nível da biblioteca, o CDC usa as bibliotecas modificadas da *Java SE*, cujas relações foram alteradas para atender às necessidades dos dispositivos de recursos limitados e cujas implementações são otimizadas para ambientes de pouca memória. Com o intuito de conservar recursos, algumas bibliotecas baseadas em Java SE tiveram suas interfaces modificadas, enquanto outras foram removidas inteiramente [SUN, 2005 a].

A CDC contém uma biblioteca derivada da tecnologia *Java SE*, mais adaptada às necessidades de dispositivos conectados. A biblioteca da CDC inclui classes de suporte às aplicações que os desenvolvedores de software têm usado por anos, desenvolvendo incontáveis aplicações para *desktop* e servidores. Enquanto a maioria das *APIs* da CDC são idênticas às do Java SE, algumas interfaces mudaram e as implementações de outras *APIs* foram ajustadas às necessidades dos dispositivos de recursos limitados. O resultado é uma biblioteca Java que permita aos desenvolvedores migrarem rapidamente seus códigos e técnicas da Java SE para a CDC [SUN, 2005 a].

É útil examinar como as *APIs* da CDC são organizadas, comparando o *Foundation Profile*, o *Personal Basis Profile* e o *Personal Profile*. Compreender as diferenças entre esses *profiles* é uma parte importante do desenvolvimento da aplicação da CDC. O *Foundation Profile* fornece um jogo básico de suporte a aplicações. Os outros *profiles* são construídos adicionando-se funcionalidades específicas ao *Foundation Profile* [SUN, 2005 a].

A JVM para a CDC é uma máquina virtual Java otimizada, desenhada especificamente para dispositivos de pequeno porte. A JVM da CDC adere às mesmas especificações da máquina virtual Java da Java SE, entretanto, seu desempenho, suporte a dispositivos, recursos de memória e características de confiabilidade são projetados em torno das necessidades dos dispositivos de pequeno porte [SUN, 2005 a].

2.3 *Profiles*

Definida logo acima da camada *configuration*, a camada *profiles* complementa as configurações provendo mais recursos e visando a verticalização de um segmento de mercado ou de uma categoria específica de dispositivo. Segundo [SUN, 2005], os *profiles* definem *APIs* de alto nível e podem trabalhar em conjunto com outros *profiles;* além disso, definem o modelo do ciclo de vida da aplicação, assim como interface com o usuário e acesso a características específicas do dispositivo.

Atualmente, existem quatro especificações de *profiles* que são: *Mobile Information Device Profile (MIDP)*, *Foundation Profile (FP)*, *Personal Profile (PP)* e *Personal Basis Profile (PBP)*. Cada uma provê determinadas características baseadas nas limitações de cada categoria de dispositivo.

O *Foundation Profile*, assim como o *Personal Profile* e sua implementação mais simples, *Personal Basis Profile*, tem como alvo dispositivos que implementam o modelo da camada de configuração CDC, o *Personal Profile* e o *Personal Basis Profile* trabalham em conjunto com o *Foundation Profile* que dá a base de acesso a eles. O MIDP não atua com outro *profile* sendo embasado no modelo da CLDC.

A funcionalidade disponibilizada pelos *profiles* provê características robustas às aplicações que vão desde conexões de redes seguras, utilizando protocolos como HTTPS (*Hypertext Transfer Protocol Secure*), até visualização de gráficos 3D e manipulação de arquivos de mídia como vídeo e áudio. A seguir é apresentada uma breve descrição dos *profiles* [SUN, 2005 a].

- **FP** (***Foundation Profile***) **-** é o perfil o mais básico do CDC. Em combinação com a biblioteca fornecida pelo CDC, o *Foundation Profile* fornece suporte básico a aplicações, tais como o suporte à rede e o suporte à E/S (entrada e saída). Em particular, não inclui nenhum suporte para gráficos ou serviços GUI[5];
- **PBP (*Personal Basis Profile*) -** fornece uma estrutura para construir componentes baseados em um conjunto de ferramentas GUI limitado baseado

5 GUI é um mecanismo de interação entre usuário e sistema de computador baseado em símbolos visuais, como ícones, menus e janelas [WIKIPÉDIA, 2006 d].

em AWT[6], suporte à execução de *JavaBeans*[7] e suporte a aplicações *xlet*[8]. Além disso, o *Personal Basis Profile* inclui toda a API do *Foundation Profile*;

- **PP (*Personal Profile*)** - fornece suporte total ao AWT, suporte a *applets*[9] e suporte limitado a *JavaBeans*. Além disso, o *Personal Profile* inclui toda a API do *Personal Basis Profile*. O *Personal Profile* representa também o trajeto da migração para a tecnologia pessoal Java;
- **MIDP (*Mobile Information Device Profile*)** - habilita aplicações de rede verdadeiramente em dispositivos de informação móveis. Para carregar uma aplicação MIDP, o usuário navega em uma lista de aplicações armazenada em um servidor *Web*. Depois que uma aplicação é selecionada, o dispositivo confere para ter certeza que pode executar a aplicação. Nesse caso, o dispositivo carrega a aplicação, verifica e compila o *bytecode* Java™ para executar no dispositivo. Uma vez instalado, aplicações MIDP podem ser facilmente atualizadas e removidas pelo usuário [SUN, 2006 g].

2.4 Pacotes Adicionais

Os pacotes opcionais estendem a plataforma Java ME acrescentando funcionalidade à tecnologia que inclui CLDC ou CDC e *profiles* associados. Criado para dirigir-se a exigências muito específicas de aplicações, os pacotes opcionais oferecem APIs para usar tecnologias existentes e emergentes tais como conectividade com base de dados, mensagens *wireless*, multimídia, gráficos 3D e *Web Services*. Como os pacotes opcionais são modulares, os fabricantes de dispositivos podem evitar o *overhead* de funções desnecessárias incluindo somente os pacotes que uma aplicação necessita realmente. Os pacotes opcionais podem ser executados virtualmente ao lado de qualquer combinação de configurações e de perfis [SUN, 2006 d].

O desenvolvimento e a solidificação da plataforma J2ME como padrão de ambiente para aplicativos em dispositivos móveis vêm crescendo nos últimos anos. A sua estrutura

6 AWT é uma coleção de classes para a construção de interfaces gráficas em Java [WIKIPÉDIA, 2006 e].
7 Componentes reusáveis de software que podem ser manipulados visualmente com a ajuda de uma ferramenta de desenvolvimento [WIKIPÉDIA, 2006 h].
8 São bem parecidos a Applets Java e são projetados para apoiar sessões de TV Digital [WIKIPÉDIA, 2006 f].
9 Applets são pequenos aplicativos escritos em Java que utilizam a JVM existente na máquina cliente ou embutida no próprio *browser* do cliente para interpretar seu *bytecode [WIKIPÉDIA, 2006 g]*.

de conectividade abre um leque de opções para acesso a informações através de uma rede. Por exemplo, com a utilização das classes definidas na especificação JSR-172 [JCP, 2006] (*Java Request Specification*), aplicativos integram-se facilmente com serviços (*Web Services*) publicados na *Web*, possibilitando a aplicação consumir serviços de diferentes naturezas.

Com base na especificação flexível e na utilização de pacotes adicionais para aumentar a conectividade dos aplicativos, a plataforma J2ME pode ser vista como solução para uma gama diversificada de aplicações além de possibilitar desenvolvimento de aplicações rápidas e portáveis.

2.5 Considerações Finais

Pôde-se notar que a plataforma *Java 2 Micro Edition* é uma coleção de tecnologias e de especificações que os desenvolvedores podem escolher e combinar para construir um ambiente de execução Java completo que atenda às exigências particulares de uma série de dispositivos e de mercados. Cada combinação é otimizada em memória, poder de processamento e capacidades de entrada e saída de uma categoria de dispositivos.

3 *WEBSERVICES*

3.1 Considerações Iniciais

Hoje em dia, a principal utilização da *World Wide Web* é o acesso a documentos e a aplicações por parte dos seres humanos. Este acesso é feito, basicamente, através de *browsers*. A utilização da *Web* como meio de interação entre aplicações de diferentes empresas podem automatizar e aumentar a eficiência dos processos entre as organizações. Porém, para que isso se torne uma realidade, aplicações desenvolvidas em plataformas heterogêneas necessitam de uma linguagem em comum [CERAMI, 2002]. Assim, segundo [CHAPPEL, JEWELL, 2002], *WebService* é um componente que implementa uma lógica de negócio, localizado em algum lugar na Internet, podendo ser acessado através de algum protocolo padrão. Este serviço pode ser simples, como fazer um *logon* em um *site*, ou tão complexo quanto realizar uma negociação empresarial.

Os *WebServices* são uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia, é possível que novas aplicações possam interagir com as existentes e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Os *WebServices* são componentes que permitem às aplicações enviar e receber dados em formato XML (*eXtensible Markup Language*). Cada aplicação pode ter a sua própria linguagem, que é traduzida para uma linguagem universal, o formato XML [WIKIPÉDIA, 2006].

Com base nesta definição, os *WebServices* diferem de outras tecnologias distribuídas como CORBA (*Common Object Request Broker Architeture*), *scripts* CGI (*Common Gateway Interface*), e RMI (*Remote Method Invocation*) devido, a sua fundamentação no XML e, por isso, possibilitam interoperabilidade entre aplicações de diferentes plataformas, ao contrário das outras tecnologias [REHM, 2005].

Segundo [REHM, 2005], um dos fatores que impulsionaram a evolução dos *WebServices* foi a programação orientada a objeto. Seguindo este conceito, cada *WebService* pode ser visto como um objeto que encapsula propriedades e métodos bem definidos. Deste modo, aplicações modulares podem ser compostas por vários *WebServices* distribuídos pela Internet. As aplicações em três camadas são um exemplo disso, onde cada camada pode estar em um servidor diferente [DEITEL et al, 2003].

Da união dos conceitos de orientação a objeto, da computação distribuída e da linguagem XML, algumas características do comportamento dos *WebServices* podem ser listadas, dentre elas [CHAPPEL, JEWELL, 2002]:

- Interoperabilidade entre plataformas: como foi explicitado anteriormente, os *WebServices* não dependem de plataformas;
- Fraco Acoplamento: um cliente não está vinculado diretamente ao serviço. Alterações em um dos itens (cliente/serviço) não necessariamente obrigam alterações no outro item;
- Habilidade de atender requisições de forma síncrona ou assíncrona;
- Baixa granularidade: o agrupamento de diversos métodos possibilita uma estrutura organizada e mais natural, na medida em que permite o acesso a determinadas partes da lógica do negócio concentradas em um único serviço.

Essas características fazem dos *WebServices* uma solução ideal para conectividade de aplicações heterogêneas através da Internet. Apesar de haver algumas incompatibilidades entre as implementações existentes, esses problemas não são decorrentes da arquitetura dos *WebServices*, mas das implementações [CERAMI, 2002].

3.2 Arquitetura

Segundo [REHM, 2005], a arquitetura dos W*ebServices* define padrões conceituais para interoperabilidade entre diferentes aplicações rodando em diferentes plataformas e escritas em diferentes linguagens de programação, ou seja, ela não define como deve ser implementado um *WebService*, nem como devem ser combinados; o que é definido na arquitetura dos *WebServices* são as regras de como os componentes se comunicam. Estas regras definem um *WebService* como sendo um sistema que suporta interoperabilidade entre máquinas, utilizando SOAP (*Simple Object Access Protocol*) ou XML-RPC (*XML Remote Procedure Call*) como protocolo de troca de mensagens, HTTP **(***HyperText Transfer Protocol***)** ou SMTP (*Simple Mail Transfer Protocol*) como protocolos de transporte e XML como linguagem. Partindo dessas características, tem-se nesta arquitetura uma estrutura capaz de interligar sistemas distribuídos de forma barata, sem a necessidade de modificações bruscas nas aplicações.

Apesar da arquitetura ser orientada a serviços, sua especificação define quatro modelos ou visões conceituais, onde aspectos importantes são abordados [W3C, 2005], a

saber:

- Modelo de Mensagens: conceitua apenas aspectos relacionados às mensagens e ao seu processamento. Especificamente, neste modelo, não estamos interessados com qualquer significado semântico do conteúdo de uma mensagem ou sua relação a outras mensagens. Porém o Modelo de Mensagens focaliza a estrutura de mensagens, na relação entre remetentes e receptores e como são transmitidas as mensagens;
- O Modelo de Serviços enfoca aspectos relativos ao relacionamento entre um agente (consumidor/provedor) e os serviços que ele solicita e utiliza;
- O Modelo de Políticas enfoca aspectos da arquitetura que relacionam as políticas por extensão, segurança e qualidade do serviço;
- O Modelo de Recursos enfoca aspectos da arquitetura que relacionam-se a recursos.

Baseado neste modelo conceitual, os *WebServices* têm o protocolo SOAP como padrão arquitetural do Modelo de Mensagens. Além do SOAP, o protocolo XML-RPC também provê mecanismos de comunicação, porém mais simplificado. Para a modelagem arquitetural do Modelo de Serviços, o padrão utilizado é a linguagem WSDL (*WebService Description Language*), que provê recursos para a descrição dos serviços e seus pontos de acesso de forma abstrata. Além desses protocolos, a arquitetura oferece o UDDI (*Universal Description, Discovery and Integration*) como padrão arquitetural do Modelo de Recursos, que visa criar um catálogo de recursos para pesquisas de serviços.

3.3 XML-RPC

O XML-RPC provê um mecanismo baseado em HTTP para execução de procedimentos remotos. De maneira simplificada, seu vocabulário consegue descrever a natureza das requisições e das respostas de forma a atender diversos sistemas. Sua especificação não provê mecanismos de suporte a objetos e os *arrays* (vetores) são a estrutura de dados mais complexa suportada [LAURENT et al, 2001].

A estrutura de uma chamada XML-RPC (Figura 3.1) é basicamente uma requisição HTTP utilizando o método POST (envia parâmetros da URL solicitada sem expor os dados) com conteúdo *text/xml* no *MIME Type*.

```
POST /xmlrpc HTTP 1.0
User-Agent:myXMLRPCClient/1.0
Host: 192.168.124.2
Content-Type: text/xml
Content-Length: 169

<?xml version="1.0"?>
<methodCall>
    <methodName>circleArea</methodName>
    <params>
        <param>
            <value>
                <double>2.41</double>
            </value>
        </param>
    </params>
</methodCall>
```

Figura 3.1: Requisição XML-RPC. (Fonte: [CERAMI, 2002]).

De maneira análoga, a resposta (Figura 3.2) contendo o valor de retorno ou uma mensagem de falha é uma resposta HTTP com o conteúdo XML.

```
HTTP/1.1 200 OK
Date: Sat, 06 Oct 2001 23:20:04 GMT
Server: Apache.1.3.12 (Unix)
Connection: close
Content-Type: text/xml
Content-Length: 124

<?xml version="1.0"?>
<methodResponse>
    <params>
        <param>
            <value>
                <double>18.24668429131</double>
            </value>
        </param>
    </params>
</methodResponse>
```

Figura 3.2: Resposta XML-RPC. (Fonte: [CERAMI, 2002]).

O fato do XML-RPC ser mais simplificado que o SOAP o permite atender uma gama maior de sistemas desenvolvidos em diferentes linguagem de programação.

3.4 SOAP

O SOAP é um protocolo de troca de mensagens que tem como finalidade principal a intercomunicação entre sistemas heterogêneos através da *Web*. Com sua estrutura baseada

em XML, sua especificação define apenas um envelope, para encapsular a informação. Além disso, o SOAP contém um conjunto de regras para tradução de tipos entre plataformas diferentes e um tipo de mensagem especial para reportar falhas [TIDWELL et al, 2001].

Por ser um protocolo de mensagens, o SOAP se adapta perfeitamente na arquitetura *request-response* do HTTP, sendo este um fator fundamental para o seu sucesso, pois o HTTP é amplamente utilizado. Porém, existem implementações baseadas em outros protocolos de comunicação como SMTP (*Simple Mail Transfer Protocol*).

De acordo com a especificação, o envelope (Figura 3.3) deve conter apenas um corpo (*Body*), o qual pode conter qualquer expressão XML bem formada. Além disso, um cabeçalho (*Header*) opcional pode conter diretivas para o processador de destino da mensagem, como regras para roteamento, entrega da mensagem e informações para autenticação e autorização em transações [TIDWELL et al, 2001].

Figura 3.3: Estrutura de um envelope SOAP (Fonte: [TIDWELL et al, 2001]).

Como dito anteriormente, o SOAP tem como finalidade intercomunicação entre sistemas heterogêneos. Por intercomunicação entende-se intercâmbio de documentos e execução de procedimentos remotos. Assim, o conteúdo de uma mensagem SOAP é composto por uma chamada RPC contendo o nome do método e seus parâmetros de entrada e saída (Figura 3.4) ou por um documento contendo informações estruturadas (Figura 3.5), ambos utilizando uma descrição XML.

```
<s:Envelope xml:s="http://www.w3.org/2001/06/soap-envelope">
    <s:Header>
        <m:transaction xmlns:m="soap-transaction" s:mustUnderstand="true">
            <transactionID>1234</transactionID>
        </m:transaction>
    </s:Header>
    <s:Body>
        <n:getQuote xmlns:n="urn:QuoteService">
            <symbol xsi:type="xsd:string">
                IBM
            </symbol>
        </n:getQuote>
    </s:Body>
</s:Envelope>
```

Figura 3.4: Estrutura de uma mensagem contendo uma chamada RPC. (Fonte: [CERAMI, 2002]).

Hoje em dia, existem diversas implementações do protocolo SOAP, a Microsoft adotou os *WebServices* como solução principal em ferramentas de desenvolvimento voltadas para sua plataforma .NET. Além dela, a Sun Microsystems e a IBM também possuem implementações amplamente utilizadas [REHM, 2005].

```
<s:Envelope xml:s="http://www.w3.org/2001/06/soap-envelope">
    <s:Header>
        <m:transaction xmlns:m="soap-transaction" s:mustUnderstand="true">
            <transactionID>1234</transactionID>
        </m:transaction>
    </s:Header>
    <s:Body>
        <n:purchaseOrder xmlns:n="urn:CrderService">
            <from>
                <person>Christopher Robin</person>
                <dept>Accounting</dept>
            </from>
            <to>
                <person>Pooh Bear</person>
                <dept>Honey</dept>
            </to>
            <order>
                <quantity>1</quantity>
                <item>Pooh Stick</item>
            </order>
        </n:purchaseOrder>
    </s:Body>
</s:Envelope>
```

Figura 3.5: Estrutura de uma mensagem contendo um documento. (Fonte: [CERAMI, 2002]).

Um ponto importante para o sucesso deste protocolo é a abstração da sua existência durante o desenvolvimento de clientes. Com algumas linhas de código, um cliente pode fazer uma requisição via SOAP deixando para a implementação da plataforma os detalhes referentes ao protocolo [CHAPPEL, JEWELL, 2002].

3.5 WDSL

A linguagem WSDL provê uma gramática concisa para especificação dos *WebServices*. Basicamente, é um contrato de implementação fornecido pelo provedor do serviço para que os consumidores tenham as informações necessárias para a implementação dos clientes [CERAMI, 2002].

Esse contrato é disponibilizado para os clientes em um formato de arquivo cuja extensão é *WSDL* (Figura 3.6). Neste arquivo, são especificados definições de protocolo de

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloService"
    targetNamespace="http://www.ecerami.com/wsdl/HelloService.wsdl"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmisoap.org/wsdl/soap/"
    xmlns:tns="http://www.ecerami.com/wsdl/HelloService.wsdl"
    xmlns:xsd="htt://www.w3.org/2001/XMLSchema">

    <message name="SayHelloRequest">
        <part name="firstName" type="xsd:string"/>
    </message>
    <message name="SayHelloResponse">
        <part name="greeting" type="xsd:string"/>
    </message>

    <portType name="Hello_PortType">
        <operation name="sayHello">
            <input message="tns:SayHelloRequest"/>
            <output message="tns:SayHelloResponse"/>
        </operation>
    </portType>

    <binding name="Hello_Binding" type="tns:Hello_PortType">
        <soap:binding style="rpc"
            transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="sayHello">
                <soap:operation soapAction="sayHello"/>
                <input>
                    <soap:body
                        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                        namespace="urn:examples:helloservice"
                        use="encoded"/>
                </input>
                <output>
                    <soap:body
                        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                        namespace="urn:examples:helloservice"
                        use="encoded"/>
                </output>
            </operation>
    </binding>

    <service name="Hello_Service">
        <documentation>WSDL File for HelloService</documentation>
        <port binding="tns:Hello_Binding" name="Hello_Port">
            <soap:address
                location="http://localhost:8080/soap/servlet/rpcrouter"/>
        </port>
    </service>
</definitions>
```

Figura 3.6: Estrutura de um arquivo WDSL. (Fonte: [CERAMI, 2002]).

transporte, *namespace* contendo especificação do envelope SOAP, estrutura das mensagens SOAP e o local onde se encontra o serviço [KEOGH, 2003].

Apesar de ser amplamente utilizado em cima da pilha do HTTP, os *web services* não atuam apenas com operações *request-response*. Há quatro tipos de operações que podem também ser especificadas no WSDL (Figura 3.7) [CERAMI, 2002], a saber:

- *One-Way* (unidirecional): o serviço recebe uma mensagem e executa um procedimento ou consome um documento;
- *Request-Response* (bidirecional): o serviço recebe uma requisição e, após o processamento, envia uma resposta contendo valores de retorno ou mensagens de falha. Este tipo de operação é o mais utilizado;
- *Solicit-Response* (bidirecional): a comunicação é iniciada pelo serviço que envia uma requisição e espera uma resposta*;*
- *Notification* (unidirecional): o serviço envia uma mensagem e encerra a comunicação.

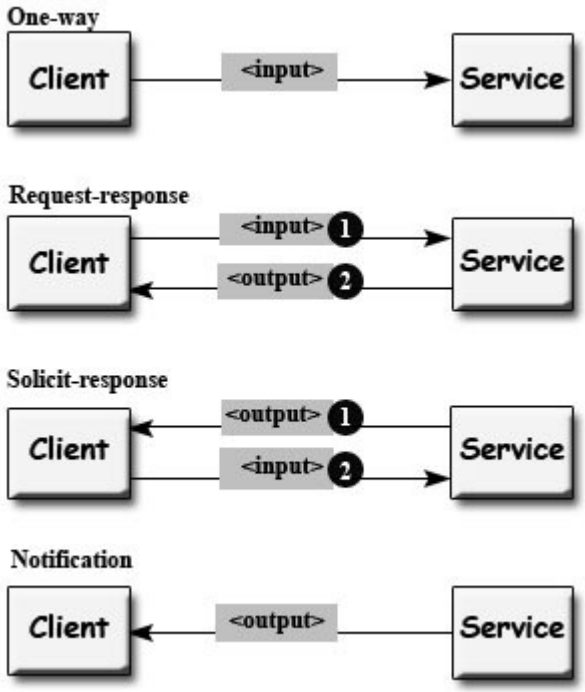

Figura 3.7: Tipos de operações suportadas pelo WebServices e seus respectivos passos. (Fonte: [CERAMI, 2002]).

Apesar da estrutura complexa, existem hoje diversos aplicativos para a criação automática dos arquivos WSDL; assim, basta especificar os pontos de acesso ao serviço e as características necessárias para a criação automática.

3.6 UDDI

Em um ambiente com muitos serviços, a manutenção e a obtenção de informações sobre os serviços disponíveis tornam-se muito complicadas. Visando catalogar e classificar os serviços de forma ordenada, o UDDI (*Universal Description, Discovery and Integration*) provê métodos padronizados para publicar e achar informações sobre *WebServices* [CHAPPEL, JEWELL, 2002].

Sua arquitetura pode ser dividida em duas partes: uma especificação técnica para construção de diretórios de negócio e um registro para empresas e seus respectivos *WebServices*, o qual pode ser visto como um catálogo de serviços [CERAMI, 2002].

Os dados disponibilizados pelo UDDI estão divididos em três categorias [CHAPPEL, JEWELL, 2002], [CERAMI, 2002]:

- Páginas Brancas: Disponibiliza informações sobre empresas, como nome, telefone e descrição do negócio;
- Páginas Amarelas: Lista os *WebServices* dentro de diferentes categorias, por exemplo, ramo do negócio;
- Páginas Verdes: Contém informações técnicas sobre os *WebServices.* Normalmente, aponta para uma especificação externa do serviço, desde uma interface WSDL até um *site* ou *e-mail*.

O uso desta tecnologia não se faz necessária para a utilização dos *WebServices*. Na verdade, ela confronta a idéia de sistemas distribuídos na medida em que centraliza o acesso às informações dos serviços.

3.7 Considerações Finais

Os *WebServices* permitem que a integração de sistemas seja realizada de maneira compreensível, reutilizável e padronizada. É uma tentativa de organizar um cenário cercado por uma grande variedade de aplicativos, fornecedores e plataformas.

Devido a sua fundamentação na linguagem XML, os *WebServices* permitem interoperabilidade entre aplicações de diferentes plataformas. Desenvolvida para o ambiente *web*, esta tecnologia possui os requisitos necessários para possibilitar sistemas interagirem entre si de forma rápida e simples, disponibilizando informações e serviços através de uma rede.

Acredita-se que no futuro as empresas irão listar seus *WebServices* em diretórios

públicos (UDDI), de onde poderão ser vendidos como serviços para outras empresas, instituições ou usuários comuns.

4 SISTEMAS ESPECIALISTAS

4.1 Considerações Iniciais

Sistemas especialistas são sistemas computacionais baseados em conhecimentos específicos de um determinado domínio e, por conseqüência, simulam a atuação de um especialista do domínio em questão. O conhecimento utilizado pelos sistemas especialistas possui um embasamento teórico aprofundado, além de heurísticas que se mostram eficazes na resolução de problemas [LUGER, 1998].

A grande vantagem dos sistemas especialistas em relação aos sistemas convencionais é sua orientação voltada para o conhecimento humano do domínio da aplicação. Um sistema especialista provido de grande base teórica sobre um determinado domínio pode solucionar diversos problemas onde um sistema desenvolvido de maneira convencional não seria capaz. Neste último, o conhecimento está codificado de maneira imutável e seqüencial podendo ser modificado apenas com alteração de código fonte que, dependendo do tamanho da aplicação, pode se tornar extenso e de difícil manutenção [REHM, 2005].

Além de outros ramos da inteligência artificial, como Redes Neurais e Algoritmos Genéticos terem dado grandes contribuições, o ramo de sistemas especialistas foi o primeiro a possibilitar soluções mais factíveis para problemas mais usuais. Dentre as áreas da inteligência artificial, a área dos Sistemas Especialistas foi a primeira a ter aplicação comercial [KRISHNAMOORTHY, RAJEEV, 1996].

Um dos primeiros sistemas especialistas desenvolvidos foi o DENDRAL, na universidade de *Stanford* em meados da década de 1960. Projetado para inferir estruturas moleculares de amostras, o DENDRAL utiliza heurísticas baseadas em experiências de especialistas químicos para contornar o problema de espaço devido ao tamanho das possíveis moléculas ser muito extenso [BITTENCOURT, 2001].

Outro sistema especialista de fundamental importância é o MYCIN [FEIGENBAUM, ENGELMORE, 1993]. Desenvolvido na década de 1970, também na universidade de *Stanford*, seu projeto influenciou a maneira como os sistemas especialistas seriam desenvolvidos. Seu objetivo inicial era diagnosticar e recomendar tratamentos para doenças infecciosas no sangue.

Além desses sistemas, o PLIDIS, para controle de emissão de poluentes, o ITA, para automação de tarefas de inspeção de robôs, e o PROSPECTOR, para auxiliar os geólogos na prospecção mineral, são outros exemplos das diversas áreas onde existem sistemas especialistas [CRIPPA, 2005], [BITTENCOURT, 2001].

4.2 Estrutura de um Sistema Especialista

Baseada em uma estrutura modular, a arquitetura dos sistemas especialistas muitas vezes provê reaproveitamento dos módulos entre diferentes soluções. Sua composição pode ser dividida em: base de conhecimento, motor de inferência e interface.

4.2.1 Base de Conhecimento

A Base de Conhecimento tem como objetivo principal representar conceitos significantes ao problema e suas relações dentro do domínio da aplicação em uma linguagem formal. Assim, a consistência destes conceitos e de seus relacionamentos é fundamental para o sucesso do sistema.

A Base de Conhecimento é composta pelo conhecimento geral necessário para solucionar problemas de um determinado domínio (base de regras), assim como dados do problema em questão (memória de trabalho).

Durante o processo de aquisição do conhecimento, a figura do engenheiro do conhecimento se faz altamente necessária. Ele é o responsável por extrair o conhecimento do especialista humano e organizá-lo de forma lógica e concisa para posterior inclusão na base. Somente após a aquisição do conhecimento, o modelo de representação e os métodos de busca são definidos.

Dentre os principais problemas encontrados na aquisição do conhecimento está a dificuldade dos especialistas em expressar de forma teórica determinados procedimentos adquiridos durante a vivência do trabalho.

4.2.2 Máquina de Inferência

Segundo [REHM, 2005], o Motor de Inferência é a base de raciocínio dos sistemas especialistas, em outras palavras, é o intérprete da base de conhecimentos. No motor de inferência, são definidos os métodos que processarão o conhecimento armazenado na base de conhecimento.

Nos sistemas com conhecimento baseado em regras, os métodos de inferência mais

comuns são: i) encadeamento progressivo (*forward-chaining*), orientado por dados, e ii) encadeamento regressivo (*backward-chaining*), orientado por objetivo, [CAWSEY, 2005]. Apesar de ambos obterem as mesmas soluções para os mesmos problemas, a eficiência do sistema pode ser comprometida dependo do método escolhido e do tipo do problema [LUGER, 1998].

A linha de resolução do *forward-chaining* (Figura 4.1) baseia-se nos dados do problema atual. Supondo estes dados verdadeiros, o motor de inferência compara as condições das regras de produção buscando as regras que atendam aos fatos do problema. Caso as condições de uma regra atendam a estes fatos, a ação desta regra é executada e colocada na área de memória do problema atual. Este processo se repete até que nenhum conhecimento adicional seja inferido e acrescentado ao problema ou o objetivo seja alcançado.

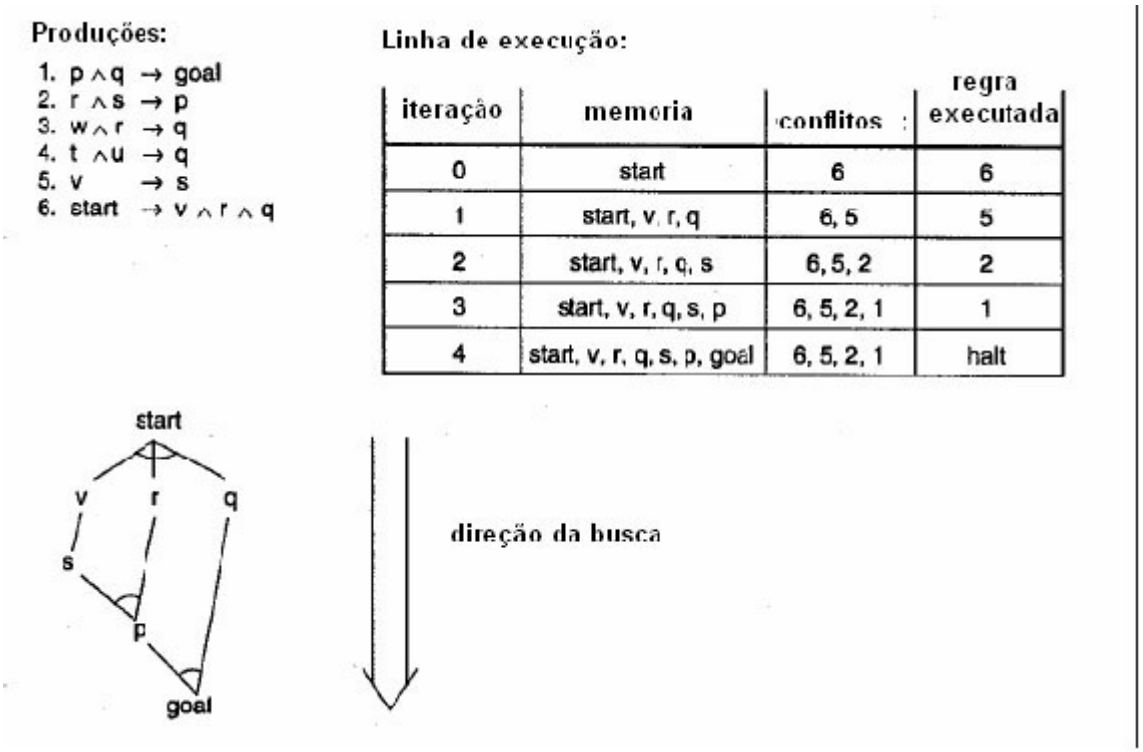

Figura 4.1: Exemplo de forward-chaining. (Fonte: [LUGER, 1998]).

Como visto na Figura 4.1, a busca ocorre partindo da ponto inicial (*start*) e executando a regra 6, é obtido *v*, *r* e *q*. Assim para *v* é executada a regra 5 obtendo *s*; agora como tem-se *r* e *s* pode-se executar a regra 2 obtendo-se *p*. Por fim encontra-se o objetivo (*goal*) a partir de *p* e *q* utilizando-se a regra 1 e encerra a busca.

No *backward-chaining* (Figura 4.2), o processo de inferência parte do objetivo ou

de prováveis objetivos que se quer encontrar. Sendo o objetivo uma ação de uma regra de produção, o sistema tenta validar quais condições devem ser verdadeiras para que este objetivo seja atendido. Nestas condições, são considerados os sub-objetivos e assim sucessivamente até que os dados iniciais do problema atendam às condições de um destes sub-objetivos ou nenhum conhecimento adicional possa ser inferido.

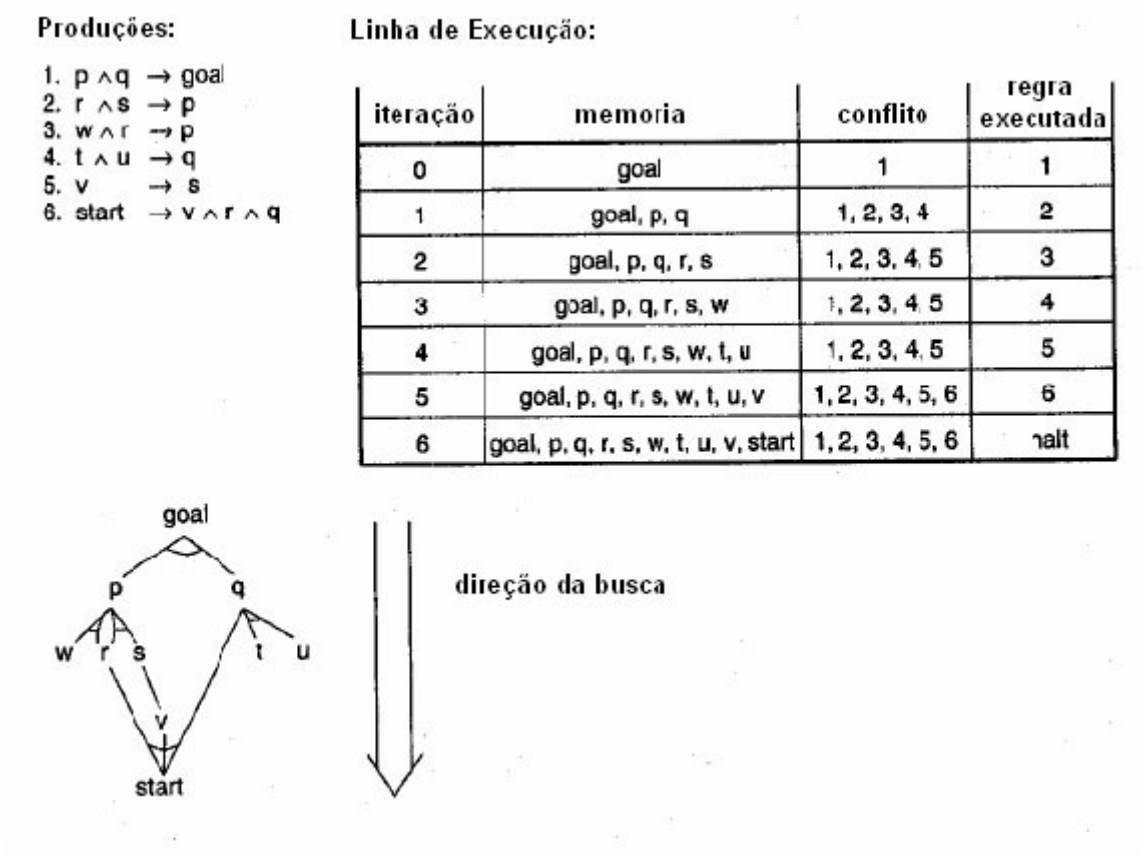

Figura 4.2: Exemplo de backward-chaining. (Fonte: [LUGER, 1998]).

Na Figura 4.2, a busca ocorreu partindo do objetivo (*goal*) até encontrar o ponto inicial (*start*). Assim, a partir da regra 1 obtém-se *p* e *q*; depois, a partir de *p* e da regra 2 obtém-se *r* e *s*; e a partir de *p* e da regra 3 obteve-se *w* e *r*; agora utilizando *q* e a regra 4 obtém-se *t* e *u*; e a partir da regra 5 e *s* obtém-se *v*. Finalmente, utilizando a regra 6, *v*, *r* e *q*, obtém-se o ponto inicial (*start*);

4.2.3 Interface com o Usuário

Por usuário deve-se entender não só usuários humanos, mas também outros sistemas que podem interagir com os sistemas especialistas. Para usuários humanos, questionários e interfaces gráficas ajudam na operação dos sistemas e na entrada de dados

mais consistente. A linguagem natural pode, algumas vezes, levar à ambigüidade. Para interface com outros sistemas, métodos ou repositório de dados podem ser utilizados. Cuidados com relação ao nível do usuário (iniciante/especialista) também são fatores importantes para o sucesso da aplicação; além disso, mensagens informativas, claras e de pedido de aguardo enquanto o sistema processa as informações são fundamentais para a aceitação por parte do usuário [BITTENCOURT, 2001].

4.3 Representação do Conhecimento

Seguindo as linhas de pesquisa da inteligência artificial, existem duas abordagens de sistemas especialistas: conexionista e simbólica [SAVARIS, 2002]. A linha conexionista visa a modelagem da inteligência humana através da simulação dos componentes do cérebro. Ela foi proposta em 1943 pelo neuropsicólogo McCulloch e o lógico Pitts criando a primeira idéia de redes neurais. A linha simbólica, que segue a tradição lógica, foi proposta por McCarthy e Newell e foi a responsável pelo sucesso dos sistemas especialistas [BITTENCOURT, 2001].

Os sistemas especialistas baseados na linha conexionista são voltados para aplicações onde o domínio do problema não é bem definido, onde existem muitas incertezas e muitas exceções às regras. Os sistemas baseados na linha simbólica são indicados para aplicações onde o domínio do problema é bem definido e não se deve trabalhar com incertezas de dados [BARRETO, 2002].

A principal diferença entre estas correntes de sistemas está na manipulação do conhecimento. Os sistemas baseados na linha conexionista não utilizam modelos de representação do conhecimento, um exemplo de sistemas deste tipo são os baseados nas redes neurais. Ao contrário, na linha simbólica, a característica principal é a utilização de técnicas de modelagem de uma base de conhecimento [REHM, 2005].

Ao longo do tempo, diversos modelos de representação do conhecimento para Sistemas Especialistas simbolistas foram propostos. Em [BITTENCOURT, 2001], estão as mais utilizadas:

- Lógica - é a base para a maioria dos formalismos de representação de conhecimento, seja de forma explícita ou disfarçada na forma de representações específicas que podem facilmente ser interpretadas como proposições ou predicados lógicos. Neste modelo, a representação do

conhecimento é feita na forma de conjuntos de instruções para solucionar um problema. Este tipo de representação é amplamente utilizado sendo o mais comum dentre os esquemas citados, pois se assemelha ao raciocínio humano que é baseado na lei de causalidade;

- Redes Semânticas - o conhecimento é estruturado na forma de grafos, onde cada nó representa um conceito e as arestas um relacionamento binário entre os nós. Esta abordagem possibilita a herança de características entre os conceitos. Este modelo visa reproduzir o comportamento da memória humana tomando como base experimentos onde o reconhecimento de elementos de classes mais abrangentes é mais lento do que o reconhecimento de elementos de classes mais restritas.

4.4 Técnicas Utilizadas no Processo de Inferência

Na resolução de problemas em domínios complexos, a árvore de busca da solução pode se tornar extensa, consumindo recursos, como tempo e espaço, de forma demasiada. Além disso, conflitos na escolha de regras a serem testadas podem ocorrer. Visando eliminar estes tipos de problemas, técnicas como busca em largura, busca em profundidade *backtracking* e heurísticas podem ser utilizadas no processo de inferência [KRISHNAMOORTHY, RAJEEV, 1996].

A busca em profundidade é mais eficiente que a busca em largura quando o número de soluções aceitáveis é grande. A busca em largura é mais vantajosa quando a árvore de estados é muito alta [KRISHNAMOORTHY, RAJEEV, 1996].

4.4.1 Busca em Largura

Na busca em largura, todos os nós do nível atual são analisados antes de descer para o próximo nível. Tomando como exemplo a busca da Figura 4.2, a análise da árvore de estados pelo método de busca em largura chegaria à solução analisando 11 estados de um total de 14 estados como mostra a Figura 4.3.

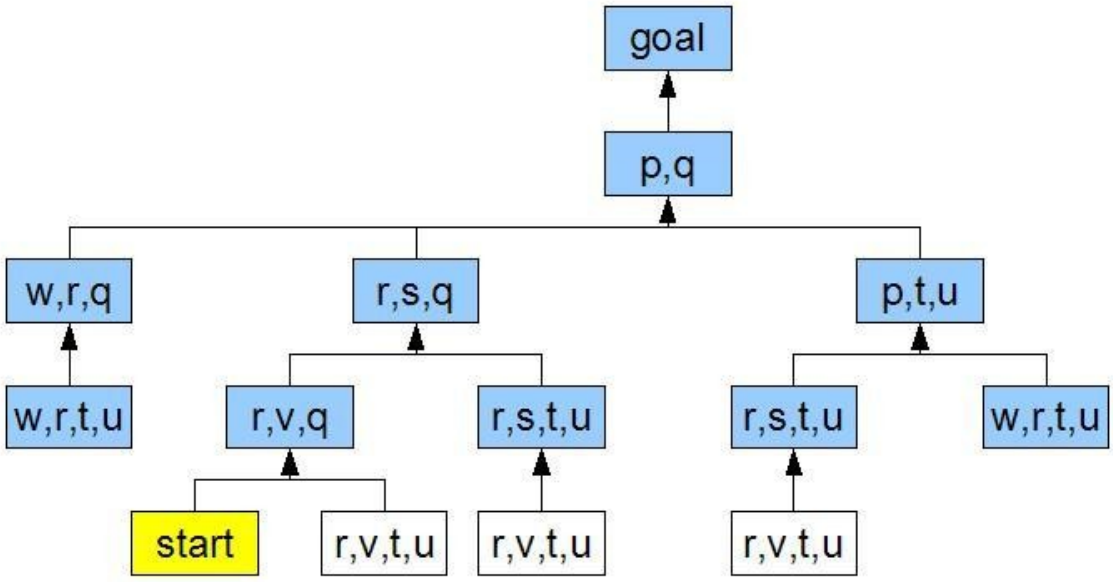

Figura 4.3: Árvore de estados com nós em azul representando a seqüência da busca em largura e em amarelo a produção inicial.

No primeiro nível da árvore de busca o objetivo (*goal*) é analisado; no segundo nível analisa-se (*p*, *q*); no terceiro nível são (*w*, *r*, *q*), (*r*, *s*, *q*) e (p, t, u) nesta ordem; no quarto nível são analisados (*w*, *r*, *t*, *u*), (*r*, *v*, *q*), (*r*, *s*, *t*, *u*), (*r*, *s*, *t*, *u*) e (*w*, *r*, *t*, *u*); no quinto nível analisa-se o ponto de partida (*start*) e finaliza a busca.

4.4.2 Busca em Profundidade

Na busca em profundidade, o nó mais à esquerda no nível atual é escolhido e analisado, e assim sucessivamente com seus nós filhos até que um novo estado não possa ser derivado. Tomando a mesma árvore de estados da Figura 4.3, a Figura 4.4 mostra a busca em profundidade analisando apenas 7 estados, aumentando a eficiência da busca neste exemplo.

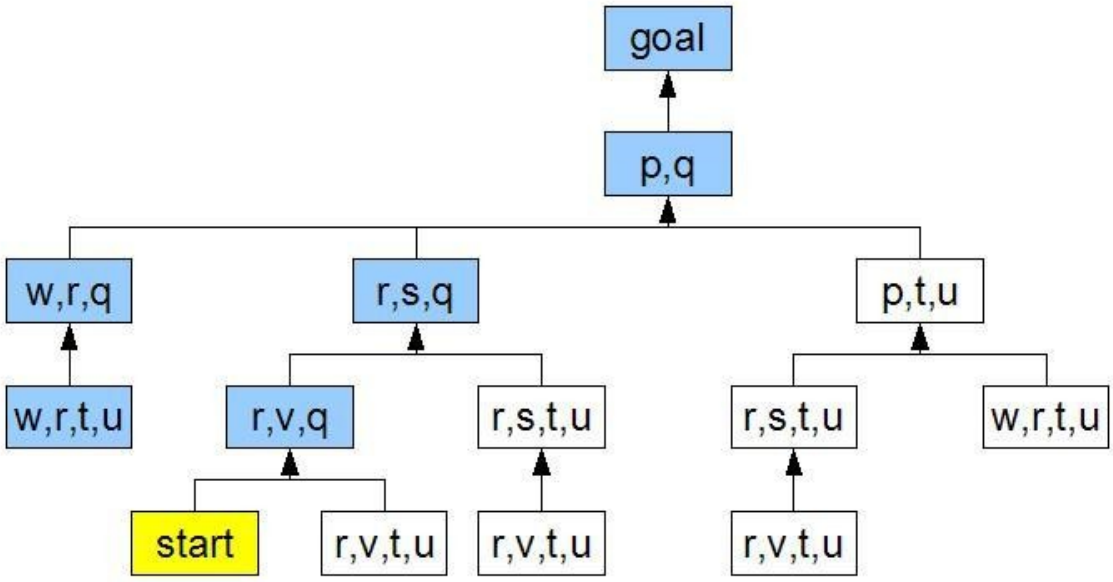

Figura 4.4: Árvore de estados com nós em azul representando a seqüência da busca em profundidade.

Partindo do objetivo (*goal*) analisa-se o estado mais à esquerda (*p*, *q*); em seguida o estado mais à esquerda de (*p*, *q*) é (*w*, *r*, *q*); o estado mais à esquerda de (*w*, *r*, *q*) é (*w*, *r*, *t*, *u*), como esse estado não possui derivações retorna-se ao estado pai (*w*, *r*, *q*) que também não possui mais derivações retornando a busca até o estado (*p*, *q*); o próximo estado a ser analisado é (*r*, *s*, *q*) e seu estado mais à direita é (*r*, *v*, *q*); por fim o estado mais à direita de (*r*, *v*, *q*) é o ponto de partida (*start*) e a busca é finalizada;

4.4.3 Heurísticas

A utilização de heurísticas no processo de busca é outro recurso que pode aumentar a eficiência, porém sacrificando a completude, ou seja, o processo é guiado para a região onde as soluções aceitáveis se encontram podendo descartar a melhor solução [RUSSEL, NORVIG, 1995]. Em alguns problemas dentro da Inteligência Artificial, esta abordagem é aceitável, assim, com base nesta afirmação, a utilização das heurísticas depende essencialmente do tipo do problema [KRISHNAMOORTHY, RAJEEV, 1996]. Em um sistema de diagnóstico, por exemplo, uma solução aceitável não é a mais recomendada, pois a melhor solução é o diagnóstico correto. Uma solução aceitável neste caso pode ser o grupo da doença, por exemplo uma virose, mas não um "palpite" sobre qual doença pode ser. Em resumo, esta técnica possibilita a poda de determinados ramos que podem não levar a solução alguma ou de ramos extenso onde a análise acarretaria na ineficiência da

busca.

4.4.4 *Backtracking*

Outra técnica utilizada na otimização no processo de busca é o *backtracking* (Retrocesso). Esta técnica possibilita a retomada da busca de um determinado nó quando a análise de um de seus sub ramos não provê uma solução para o problema. O processo se dá da seguinte maneira: quando um nó é atingido este é marcado e, conseqüentemente, um de seus sub ramos é analisado, caso este sub ramo não leve a uma solução a busca retorna para o nó marcado (raiz do sub ramo) e um outro sub ramo é analisado.

Baseando-se na mesma árvore das duas últimas figuras, a Figura 4.5 mostra a busca por *backtracking*, que nesse caso alcançou a mesma eficiência da busca em profundidade com 7 estados analisados.

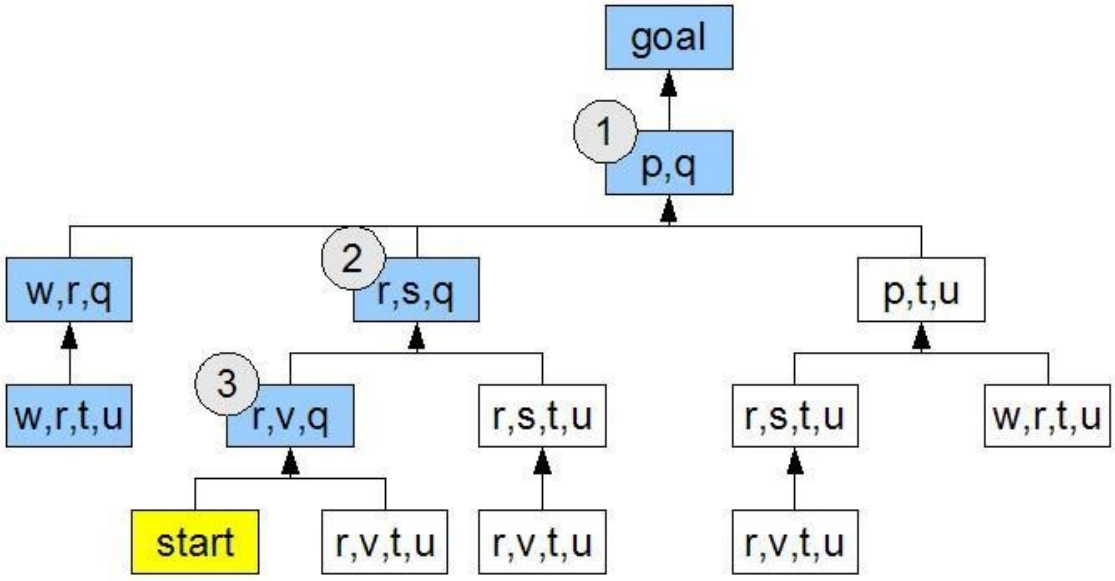

Figura 4.5: Processo de Backtracking. Nós contendo ramificações são marcados para possíveis retrocessos na árvore.

Partindo do objetivo (*goal*) analisa-se o estado mais à esquerda (*p*, *q*) e como esse possui mais de 1 derivação é marcado para retrocesso (1); em seguida o estado mais à esquerda de (*p*, *q*) é (*w*, *r*, *q*); o estado mais à esquerda de (*w*, *r*, *q*) é (*w*, *r*, *t*, *u*), como esse estado não possui derivações retorna-se ao estado marcado anteriormente (1) (*p*, *q*); o próximo estado a ser analisado é (*r*, *s*, *q*), marcado para retrocesso (2) e seu estado mais à direita é (*r*, *v*, *q*); o estado (*r*, *v*, *q*) também é marcado para retrocesso e seu estado mais à esquerda, o ponto de partida (*start*), é analisado e a busca é finalizada;

4.5 Máquina de Inferência - Método de Mamdani

Uma regra Se (antecedente) então (conseqüente) é definida pelo produto cartesiano *fuzzy* dos conjuntos *fuzzy* que compõem o antecedente e o conseqüente da regra. O método de Mamdani agrega as regras através do operador lógico *OU*, que é modelado pelo operador máximo e, em cada regra, o operador lógico *E* é modelado pelo operador mínimo. Veja as regras a seguir [LOPES et al, 2005]:

- Regra 1: Se (x é A_1 e y é B_1) então (z é C_1).
- Regra 2: Se (x é A_2 e y é B_2) então (z é C_2).

A Figura 4.6 ilustra como uma saída real *z* de um sistema de inferência do tipo Mamdani é gerada a partir das entradas *x* e *y* reais e a regra de composição *max-min*. A saída $z \in \mathbb{R}$ é obtida pela *desfuzzificação* do conjunto de saída $C = C'_1 \cup C'_2$.

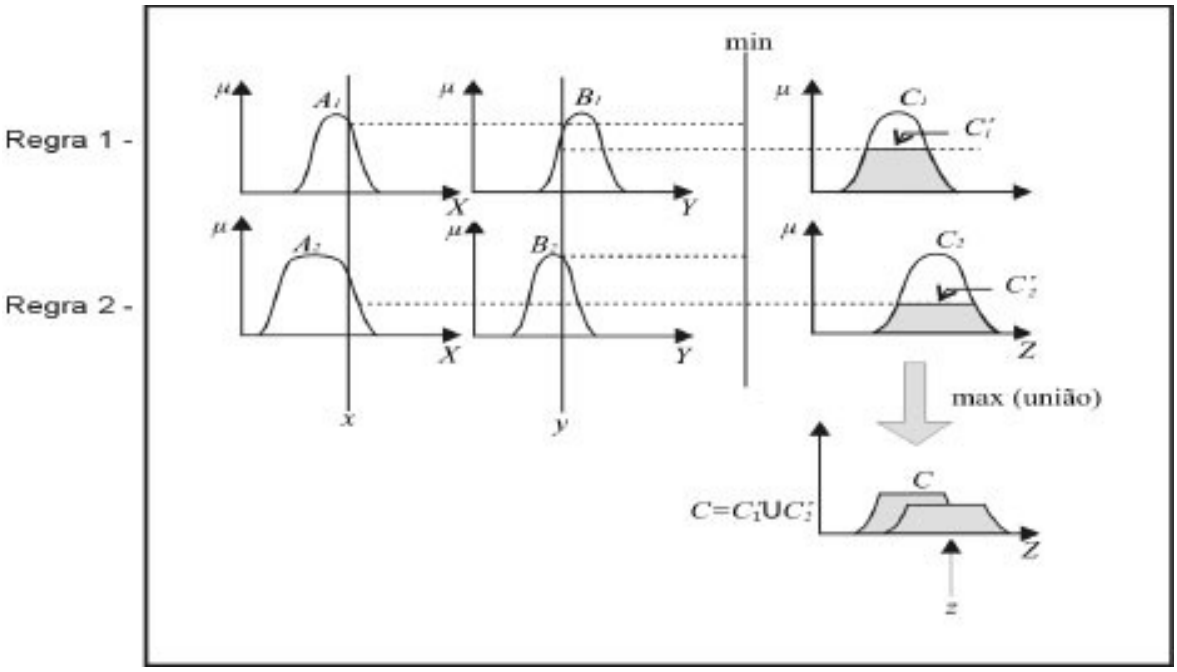

Figura 4.6: Método de Mamdani com composição max-min (Fonte: [LOPES et al, 2005])

Em seu trabalho, [LOPES et al, 2005] apresentou uma importante definição em diagnóstico médico, a composição de relações *fuzzy* binárias, que é apresentada na Figura 4.7. Esta relação *R* define uma função de $F(U_1)$ em $F(U_2)$ que a cada elemento $A_1 \in F(U_1)$, faz corresponder o elemento $A_2 \in F(U_2)$.

$$U_{A_2}(X_2) = U_{R(A_1)}(X_2) = \max_{x_1 \in U_1} [\min(U_{A_1}(X_1), U_R(X_1, X_2))]$$

Figura 4.7: Função de pertinência (Fonte: [LOPES et al, 2005]).

4.6 Classificação dos Sistemas Especialistas

A diversidade de domínios, como medicina, engenharia, educação e apoio empresarial, utilizando soluções baseadas em sistemas especialistas possibilitou a exploração de diversas categorias de problemas. Dentre estas categorias estão [LUGER, 1998]:

- Interpretação: Formula conclusões extraídas de coleções de dados. Determina relações e seus significados descrevendo situações, descartando dados inconsistentes ou fora de um determinado padrão;
- Previsão: Projeção futura baseada em situações atuais e dados do passado;
- Diagnóstico: Determinação de causas de mal funcionamento ou doenças em situações complexas baseadas em sintomas observados;
- Projeto: Projetos de sistemas de componentes atendendo aos objetivos estabelecidos, satisfazendo determinadas restrições. Estabelece etapas junto com seus sub objetivos visando atender um objetivo final;
- Planejamento: Determinação de um conjunto de ações para atingir um objetivo partindo de condições iniciais e condições não esperadas;
- Monitoramento: Comparação de um comportamento atual com um comportamento esperado;
- Depuração e Reparo: Prescrição e implementação de soluções para mau funcionamento;
- Instrutor: Detecção e correção de deficiências de estudantes;
- Controle: Controle de ambientes complexos como controle de processos em fábricas e linhas de montagem. Possui características dos outros sistemas, pois monitora, faz previsões e planeja ações de correção ou alterações no processo.

4.7 Alguns Sistemas

4.7.1 MYCIN

O sistema MYCIN foi um dos primeiros Sistemas Especialistas. Seu objetivo é prover conselho a respeito de diagnóstico e terapia de doenças infecciosas. Esse tipo de aconselhamento pode ser muito útil, pois nem sempre o médico responsável é um

especialista em infecções, principalmente em ambiente hospitalar [BITTENCOURT, 2001].

Uma seção do sistema inicia-se com um questionário, a ser respondido pelo usuário, a respeito do paciente. Informações como nome, idade, sexo, tempo de manifestação dos sintomas e resultados de exames são solicitados. A partir dessas informações e utilizando sua base de regras, o sistema é capaz de estabelecer um diagnóstico e propor uma terapia adequada [BITTENCOURT, 2001].

A base de regras do sistema contém 450 regras, que lhe permitem diagnosticar e prescrever tratamentos para bacteremia (infecção no sangue), meningite, cistite infecciosa. As regras são representadas internamente na forma de uma lista (na linguagem de programação *Lisp*) do tipo [BITTENCOURT, 2001]:

```
((and (same cntct infect primary-bacteremia)
      (membf cntxt site sterilesites)
      (same cntct portal gi))
(conclude cntxt ident bacteroides tally 0.7))
```

Esta representação é automaticamente traduzida pelo sistema para uma forma mais legível. O resultado desta tradução, que pode ser utilizado para explicar o raciocínio do sistema, é o seguinte [BITTENCOURT, 2001]:

If *the infection is primary-bacteremia, and*
the site of the culture is one of the sterile sites, and
the suspected portal of entry of the organism is a gastrointestinal tract,
Then *there is suggestive evidence (0,7) that the identily of the organism is bacteroides.*

As premissas das regras são constituídas por combinações *booleanas* de cláusulas. Cada cláusula é composta por um predicado e uma tripla de parâmetros, com a seguinte interpretação [BITTENCOURT, 2001]:

(<predicado> <objeto> <atributo> <valor>)

Os predicados são independentes do domínio. Existem 21 predicados pré-definidos, por exemplo *SAME*, *KNOWN* e *DEFINE*. Os objetivos, os atributos e os valores dependem do domínio de aplicação. Alguns exemplos de triplas (<objeto> <atributo> <valor>) utilizadas no sistema são [BITTENCOURT, 2001]:

(ORGANISM IDENTITY E-COLI), (CULTURE SITE BLOOD).

A parte de ação da regra contém um ou mais fatos a serem concluídos.

Os elementos da memória de trabalho, que contêm fatos médicos referentes ao

paciente e à sua doença, são representados na forma de listas de 4 elementos [BITTENCOURT, 2001]:

(<atributo> <objeto> <valor> <coeficiente de certeza>)

Por exemplo:

(IDENT ORGANISM-2 KLEBSIELLA 0.25)

Cada regra e cada elemento da memória de trabalho são associados a um coeficiente de certeza. Os coeficientes de certeza no *MYCIN* variam entre -1 (totalmente falso) e 1 (totalmente verdadeiro) e são utilizados para propagar a incerteza inicial de uma informação através da cadeia de inferências. O método para realizar esta propagação foi desenvolvido de maneira *ad-hoc*, mas provou-se eficiente para a aplicação desenvolvida. Caso o coeficiente de certeza fique entre -0.2 e 0.2, o fato ou a conclusão da regra é considerado desconhecido [BITTENCOURT, 2001].

O sistema *MYCIN* utilizou o encadeamento regressivo associado a uma busca em profundidade. A busca realizada é completa, no sentido em que, dado um objetivo, as evidências a favor e contra são pesquisadas. A aquisição de conhecimento para o sistema *MYCIN* é facilmente explorada no sistema *TEIRESIAS*, que permite modificar interativamente a base de regras do *MYCIN* [BITTENCOURT, 2001].

4.7.2 NIACIN

O sistema NIACIN é composto por um programa principal, onde estão as rotinas de manipulação de arquivos e de consulta, e uma série de arquivos independentes, onde estão as bases de conhecimento. O programa principal, por sua vez, é dividido em três módulos: Arquivos, Edição e Opções [SILVA, PARIZE, 1995].

No módulo Arquivos, estão as funções do programa que permitem ao usuário manipular as diferentes bases de conhecimento armazenadas em disco, assim como a interface com o sistema operacional (Carregar, Salvar, DOS, etc.) [SILVA, PARIZE, 1995].

No módulo Edição, é possível optar por diferentes meios de montar e alterar uma base de conhecimento. Estão previstas as funções necessárias para incluir novos diagnósticos, sintomas e tratamentos, corrigir ou modificar os que estão incluídos na base e indicar associações entre os termos da base. Essas associações podem ser efetuadas entre diagnósticos e sintomas, entre sintomas e outros sintomas e entre diagnósticos e

tratamentos [SILVA, PARIZE, 1995].

No módulo Opções, estão as funções usadas para a listagem e consulta à base de conhecimento. A listagem pode ser feita de diferentes formas, tais como: quais são os sintomas de uma determinada doença, quais as doenças que apresentam um determinado sintoma, quais os tratamentos para uma determinada doença, etc. A consulta ao sistema especialista, para realizar um determinado diagnóstico e propor tratamentos, se faz mediante uma lista de sintomas, que aparece em ordem alfabética na tela do computador. Pode-se trabalhar com todos os sintomas cadastrados ou apenas com aqueles referentes a um determinado sistema orgânico (cardiovascular, respiratório, locomotor, etc.). Após o usuário ter informado os dados iniciais, o NIACIN passa a gerar perguntas adicionais, visando a diferenciação das hipóteses diagnósticas. À medida que isto é feito, o programa informa na tela as estratégias de raciocínio que estão sendo seguidas, assim como uma pontuação quantitativa para as três hipóteses mais prováveis até aquele momento [SILVA, PARIZE, 1995].

O sistema NIACIN foi testado com uma base de conhecimento contendo 16 causas de vaginites. Os sintomas desta base se constituem, na maior parte, de características de leucorréias, tais como aspecto, odor, prurido e cor. Em adição, são consideradas como sintomas as características gerais das lâminas de esfregaço de secreção vaginal, em suas diferentes preparações (a fresco, Papanicolau e coloração pelo método Gram). O sistema foi capaz de, mediante informação dos dados iniciais simples, proceder a consultas, com resultados acurados e satisfatórios [SILVA, PARIZE, 1995].

O sistema mostrou-se ser capaz de trabalhar com diferentes bases de conhecimento (bases sobre vaginites, raiva, diarréia infantil, diarréia infecciosa em neonatos e doenças exantemáticas). A montagem dessas bases usando o NIACIN é relativamente simples e rápida. A coleta de dados fiéis e definitivos constituí-se no passo de maior dificuldade no desenvolvimento de qualquer sistema especialista. O uso de variáveis semânticas contribui para facilitar a assimilação de dados encontrados na forma textual, os quais são muito mais fáceis de obter, através da literatura médica e de entrevista aos especialistas humanos, do que, por exemplo, tabelas de ocorrência e sintomas e prevalência de doenças, como é necessário nos sistemas baseados no teorema de Bayes. As bases de conhecimento que abordam campos específicos de conhecimento tendem a ser melhor avaliadas pelo sistema [SILVA, PARIZE, 1995].

Dessa forma conclui-se que o sistema NIACIN pode ser facilmente utilizado por profissionais de saúde para o desenvolvimento de sistemas especialistas em qualquer área da Medicina. O seu uso mais extenso nesse sentido servirá para realizar uma melhor avaliação de suas potencialidades e desvantagens [SILVA, PARIZE, 1995].

4.7.3 *Sistemas Especialistas em Cardiologia* - SEC

A cardiopatia isquêmica é uma doença cardíaca decorrente da diminuição, significativa, do fluxo coronariano devido à formação de placas de gordura (ateroma) em determinados segmentos das artérias coronarianas (segmento proximal, médio ou distal) e/ou redução do calibre por espasmos. A cardiopatia isquêmica pode se apresentar das seguintes formas [RABELO Jr. et al, 1993]:

- *Angina Estável*, que pode ser caracterizada pela dor no peito resultante de esforço físico realizado pelo paciente. A angina estável melhora com repouso e/ou nitrato sub-lingual, pode ser controlada com medicação e é causada por obstruções fixas, de graus variáveis, em uma ou mais artérias coronarianas;
- *Angina Instável*, que pode ser caracterizada como uma síndrome clínica aguda entre a angina estável e o infarto agudo do miocárdio (angina pré-infarto). Ocorre em pacientes que percebem sintomas nunca sentidos ou em pacientes que possuam angina estável e percebem sintomas mais fortes causada por espasmo coronariano ou por instabilidade da placa de ateroma;
- *Infarto Agudo do Miocárdio*, uma síndrome clínica aguda caracterizada por dor no precórdio, geralmente retro-esternal, em aperto ou peso, que não melhora com repouso e/ou nitrato sub-lingual. É causada, quase sempre, pela formação do trombo nas artérias coronarianas, decorrente da instabilidade da placa de ateroma, cujo trombo não sofreu trombólise espontânea e obstruiu totalmente a luz da artéria. O espasmo coronário é responsável por pequena parcela dos casos.

Algumas características particulares das ocorrências de atendimentos de pacientes com suspeita de cardiopatia isquêmica são [RABELO Jr. et al, 1993]:

1. Urgência na formulação do diagnóstico e nas medidas terapêuticas a serem tomadas. O tempo decorrido entre o diagnóstico preciso da dor e a tomada de uma decisão acertada é crucial, devido ao fato de que a maioria das

complicações, inclusive o óbito, nos eventos coronarianos agudos ocorre nas primeiras horas. Existem ainda medidas terapêuticas que estão associadas ao tempo, como é o caso da trombólise. A administração de trombolíticos para desobstruir a artéria responsável pelo infarto, se feita nas primeiras horas, tem, aproximadamente, 80% de chance de sucesso. Este índice cai drasticamente até a 12ª hora e, daí em diante, há uma diminuição significativa do benefício trazido pelo seu uso. A própria indecisão do paciente quanto à sua ida a um posto de saúde gera uma perda de tempo inicial. Ao definir que o paciente deve ser removido para um centro especializado, mais um gasto de tempo ocorre neste deslocamento. Ainda são gastos alguns minutos desde a sua chegada no centro especializado e o início da trombólise. Por isso, deve-se procurar encaminhar o paciente o mais rápido possível para locais com condições de fazer o tratamento adequado [RABELO Jr. et al, 1993].

2. O diagnóstico de eventos agudos da cardiopatia isquêmica está centrado na história clínica do paciente, no exame físico e no eletrocardiograma realizados. Estes são os elementos básicos e essenciais para a elaboração do diagnóstico com elevada chance de sucesso. Assim, qualquer posto de saúde, mesmo que pequeno, poderia formular um diagnóstico de evento coronariano agudo. O que ocorre é o diagnóstico não ser feito pela falta de conhecimento especializado [RABELO Jr. et al, 1993].

Neste contexto, um sistema especialista pode ser de extrema validade, pois, a partir de um grupo de informações básicas iniciais, o sistema pode conduzir um diálogo com o usuário-médico, a fim de levá-lo ao diagnóstico mais provável, além de fornecer uma explicação em linguagem natural do caminho seguido para a formulação do diagnóstico [RABELO Jr. et al, 1993].

Considerando-se esta realidade, o SEC tem como primeiro objetivo o diagnóstico de eventos agudos da cardiopatia isquêmica. Seus usuários serão médicos não especialistas em cardiologia ou em formação na especialidade (internos e residentes). O sistema dará apoio na formulação diagnóstica do problema do paciente e na determinação do tipo e forma de atendimento que lhe deverá ser dado, em função da gravidade de seu quadro clínico. Assim sendo, o objetivo do sistema é ser utilizado em unidades de atendimento periféricas urbanas, de natureza primária da rede pública de saúde, visando auxiliar o

médico não especialista, no encaminhamento de pacientes que requeiram atendimento em uma unidade de referência especializada [RABELO Jr. et al, 1993].

4.8 Considerações Finais

Se por um lado sistemas especialistas se destacaram na pesquisa em Inteligência Artificial por achar aplicação prática no mundo real; por outro lado, essa aplicação tem sido restrita. Sistemas especialistas são notoriamente limitados no domínio do conhecimento e, portanto, propenso a cometer erros que um especialista humano facilmente detectaria. Além disso, uma vez que a mística em torno do tema diminuiu, vários programadores perceberam que sistemas especialistas simples eram essencialmente versões ligeiramente mais elaboradas de programas procedurais que eles vinham utilizando há bastante tempo. Sendo assim, algumas das técnicas de sistemas especialistas podem ser encontradas em vários programas complexos sem qualquer alarde em relação a isso.

5 SISTEMA MOBILEX-DAVIS

5.1 Considerações Iniciais

A modelagem proposta por [LOPES et al, 2005], serve como base para o desenvolvimento do sistema MobileX-DAVIS, pois trata-se de um sistema baseado em regras *fuzzy* e contém quatro componentes: o processador de entrada, a máquina de inferência, a base de regras e o processador de saída.

Segundo [BARROS e BASSANEZI, 2001] APUD [LOPES et al, 2005], o Diagnóstico Médico *Fuzzy* é uma aplicação da teoria dos conjuntos *fuzzy* que é feita com a ajuda de um especialista médico. O objetivo desta aplicação é propor um sistema especialista para ajudar o médico a tomar decisões e optar por exames laboratoriais mais detalhados.

Devido à maior quantidade de dispositivos disponíveis no mercado implementar a configuração CLDC, o escopo deste trabalho é restringido a este tipo de configuração e, conseqüentemente, ao modelo da arquitetura baseado nesta implementação. Isso também restringe a implementação do *profile* ao MIDP, o único atualmente suportado pela configuração CLDC.

Apesar de dispositivos como o *Palm* serem capazes de processarem números de ponte flutuante, todo o processamento desse tipo será efetuado pelo servidor. O servidor também será responsável pelo armazenamento dos dados dos pacientes e pelo *WebService*.

5.2 Arquitetura

O MobileX-DAVIS trata-se de um sistema baseado em regras *fuzzy* e assim como tal, possui um processador de entrada que realiza a *fuzzificação* dos dados de entrada, uma coleção de regras nebulosas chamadas base de regras, uma máquina de inferência *fuzzy*, e um processador de saída que fornece um número real. Estes componentes estão conectados conforme a Figura 5.1.

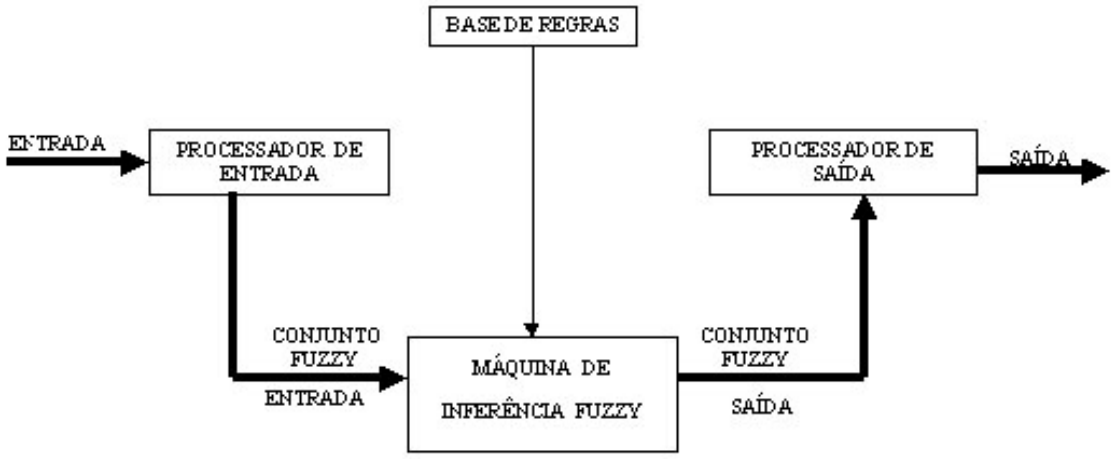

Figura 5.1: Arquitetura de sistemas baseados em regras fuzzy (Fonte: [LOPES et al, 2005]).

Componentes de um sistema baseado em regras *fuzzy* segundo [LOPES et al, 2005]:

- Processador de Entrada - Neste componente, as entradas do sistema são traduzidas em conjuntos *fuzzy* em seus respectivos domínios. No MobileX-DAVIS, é representado pelo módulo para *Palm*;
- Base de Regras - Este componente, juntamente com a máquina de inferência, pode ser considerado o núcleo dos sistemas baseados em regras *fuzzy*. No MobileX-DAVIS, a base de regras encontra-se acoplada à máquina de inferência;
- Máquina de Inferência *Fuzzy* – Neste componente, utiliza um método particular de inferência *fuzzy*, o Método de Mamdani. Pelo método de Mamdani trabalhar com números de ponto flutuante e dispositivos como o *Palm* terem menor capacidade de processamento, a máquina de inferência foi implementada para atuar em um servidor. E então para a comunicação com o módulo para *Palm* houve a necessidade de utilizar-se dos *WebServices*.

A partir dos componentes apresentados anteriormente pode-se compor o sistema MobileX-DAVIS conforme a Figura 5.2:

1. O paciente é atendido pelo médico ou responsável; este por sua vez, fazendo uso de um *Palm*, fornece o grau dos sintomas relacionados às doenças das vias aéreas superiores e inferiores;
2. Os sintomas são enviados para o *WebService* como um conjunto *fuzzy;*
3. O *WebService* é encarregado de repassar o conjunto *fuzzy* para a máquina de

inferência;

4. Após a máquina de inferência encontrar uma solução, ela retorna para o *WebService* um conjunto *fuzzy* com a possibilidade do paciente estar infectado com cada umas das doenças;

5. O *WebService* por sua vez encarrega-se de repassar esse conjunto *fuzzy* para o *Palm*. Após a recepção dos dados pelo *Palm*, são exibidas na tela as possibilidades de cada doença.

Figura 5.2: Arquitetura do sistema MobileX-DAVIS.

5.3 A Base de Regras do MobileX-DAVIS

Em seu trabalho, [LOPES et al, 2005] relacionou os sinais e os sintomas dos pacientes com as possíveis doenças das vias aéreas superiores e inferiores de acordo com os conhecimentos médicos da especialista. Foram considerados os seguintes conjuntos universais:

U = conjunto dos pacientes;

V = conjunto de sinais e sintomas;

W = conjunto de doenças.

Neste caso, trata-se de doenças das vias aéreas superiores e inferiores. Foram analisados 7 pacientes (P_1, P_2, P_3, P_4, P_5, P_6 e P_7) com sinais e sintomas definidos por s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}, s_{13} e s_{14}. Os diagnósticos foram relacionados por d_1, d_2, d_3, d_4, d_5, d_6 e d_7, onde:

- s_1 = febre;
- s_2 = tosse produtiva;
- s_3 = tosse seca;
- s_4 = cefaléia;
- s_5 = dor torácica;
- s_6 = dores musculares;
- s_7 = mal-estar geral;
- s_8 = irritação de garganta;
- s_9 = rouquidão;
- s_{10} = coriza;
- s_{11} = espirros;
- s_{12} = dispnéia;
- s_{13} = sudorese;
- s_{14} = calafrios.

- d_1 = pneumonia;
- d_2 = bronquite;
- d_3 = rinite;
- d_4 = sinusite;
- d_5 = gripe;
- d_6 = laringite;
- d_7 = amigdalite.

Esses dados irão compor a base de conhecimentos e serão expressos por meio de relações *fuzzy*. Em [LOPES et al, 2005], solicitou-se a uma especialista que estabelecesse o grau da relação *fuzzy* R mostrada na Tabela 5.1, onde as colunas são as doenças consideradas, as linhas os sinais e os sintomas, e os valores da tabela são o grau com que os sinais e sintomas se relacionam com as doenças. Essa tabela foi implementada em banco de dados, para fácil utilização e modificação.

s \ d	d_1	d_2	d_3	d_4	d_5	d_6	d_7
s_1	1.0	0.1	0.0	0.6	0.5	0.2	0.9
s_2	0.8	0.3	0.2	0.7	0.5	0.4	0.1
s_3	0.8	0.9	0.8	0.5	0.5	0.4	0.2
s_4	0.3	0.2	0.2	0.9	0.8	0.1	0.3
s_5	0.8	0.4	0.1	0.1	0.2	0.1	0.0
s_6	0.4	0.0	0.4	0.2	0.9	0.3	0.6
s_7	0.9	0.3	0.2	0.7	0.8	0.3	0.9
s_8	0.1	0.1	0.3	0.4	0.8	0.5	1.0
s_9	0.0	0.3	0.2	0.1	0.3	1.0	0.4
s_{10}	0.2	0.2	0.9	0.8	0.5	0.2	0.1
s_{11}	0.2	0.2	1.0	0.2	0.6	0.1	0.0
s_{12}	0.8	1.0	0.3	0.2	0.5	0.3	0.2
s_{13}	0.7	0.6	0.0	0.1	0.4	0.0	0.1
s_{14}	0.8	0.0	0.1	0.4	0.6	0.2	0.5

Tabela 5.1: Relação fuzzy sintomas x doenças (Fonte: [LOPES et al, 2005]).

A Tabela 5.2 apresenta o grau em que os sinais e os sintomas apareceram nos pacientes, estes dados também foram solicitados à especialista em [LOPES et al, 2005]. Estes dados não foram implementados, mas foram utilizados para os testes do MobileX-DAVIS.

P \ s	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
P1	0.0	0.0	0.9	0.3	0.0	0.0	0.1	0.6	1.0	0.2	0.0	0.1	0.0	0.0
P2	0.8	0.0	0.1	0.2	0.0	0.0	0.6	1.0	0.0	0.0	0.0	0.0	0.5	0.2
P3	0.0	0.0	1.0	0.3	0.4	0.0	0.0	0.0	0.0	0.0	0.3	1.0	0.8	0.0
P4	1.0	0.7	0.4	0.5	0.6	0.6	0.8	0.1	0.0	0.0	0.0	0.5	0.3	0.3
P5	0.0	0.0	0.4	0.3	0.0	0.0	0.2	0.2	0.0	0.8	0.9	0.0	0.1	0.1
P6	0.3	0.5	0.0	1.0	0.0	0.0	0.3	0.1	0.0	0.4	0.3	0.2	0.0	0.0
P7	0.8	1.0	0.0	0.5	0.2	0.9	0.6	0.2	0.0	0.3	0.5	0.1	0.8	0.5

Tabela 5.2: Relação fuzzy pacientes x sintomas (Fonte: [LOPES et al, 2005]).

5.4 Implementação da máquina de inferência

A partir da Figura 4.7 (página 33), foi modelado e implementado o algoritmo da Figura 5.3.

O método diagnóstico tem como parâmetro de entrada um vetor de números reais representando os sinais e os sintomas do paciente. Como parâmetro de saída um outro vetor de números reais representando a possibilidade do paciente estar infectado com cada uma das doenças.

```
public static double probabilidade(int doenca, double[] paciente) {
    double min = 0;
    for (int i = 0; i < SD.length; i++) {
        double a = SD[i][doenca];
        double b = paciente[i];
        min = Math.max(min, Math.min(a, b));
    }
    return min;
}

public static double[] diagnostico(double[] paciente) {
    double[] retValue = new double[doencas.length];
    for (int i = 0; i < doencas.length; i++) {
        retValue[i] = probabilidade(i, paciente);
    }
    return retValue;
}
```

Figura 5.3: Métodos diagnóstico e possibilidade que atuam como máquina de inferência.

O método possibilidade tem como parâmetros de entrada um inteiro representando uma doença *d* e um vetor de números reais representando os sinais e os sintomas do paciente; apresentada como parâmetro de saída um número real que representa a possibilidade do paciente estar infectado com a doença *d*. A variável *SD* é uma matriz que representa a Tabela 5.1 e *SD.length* é a quantidade de sintomas. A variável *paciente* pode ser vista como cada uma das linha da Tabela 5.2.

A máquina de inferência localiza-se em um servidor, então faz-se necessária a utilização de um *WebService* para acessá-la de um *Palm*.

5.5 Modelagem UML

A Figura 5.4 apresenta o diagrama de casos de uso do sistema. Nela, o ator médico ou responsável, pode: efetuar diagnóstico do paciente, inserir paciente, consultar histórico do paciente e calibrar doenças. Ao calibrar as doenças, o médico ou responsável estará adequando a base de dados da máquina de inferência para suas próprias regras de inferência.

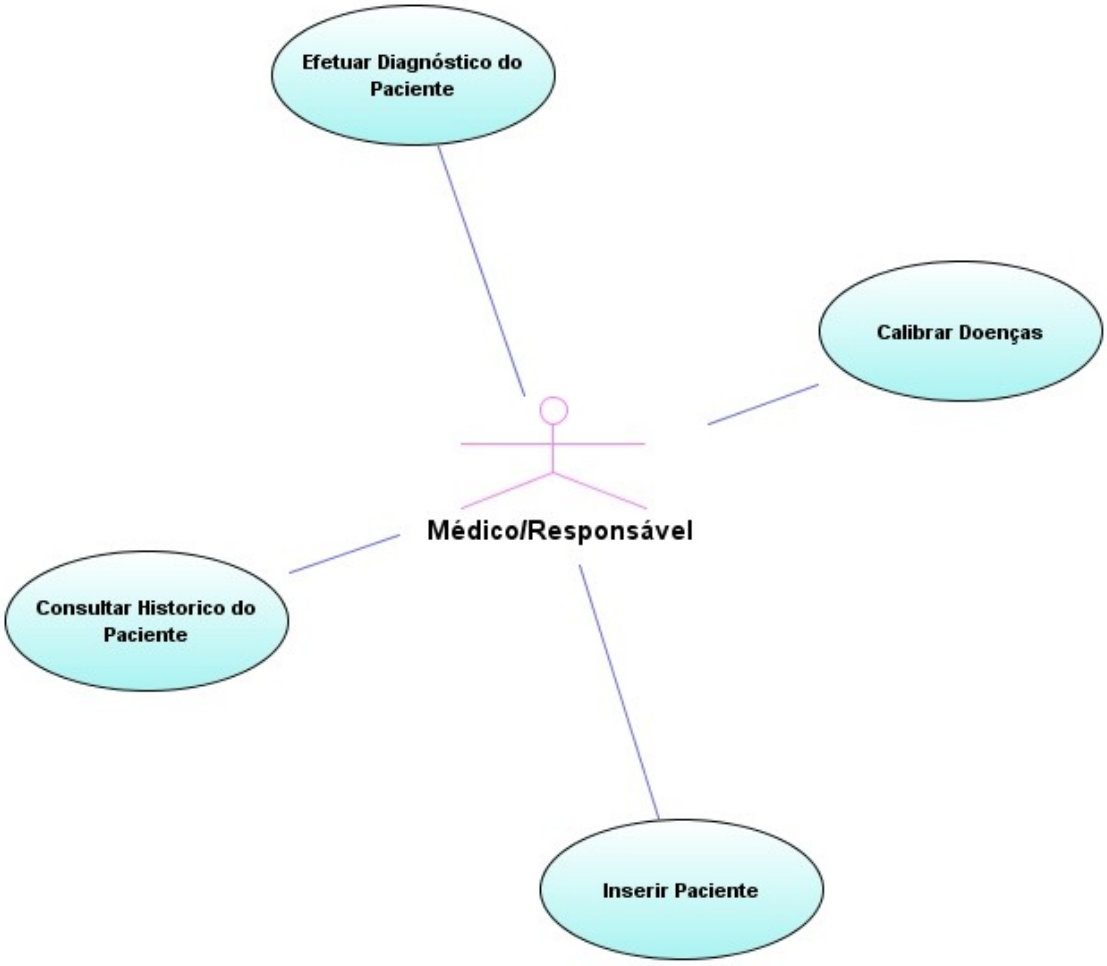

Figura 5.4: Diagrama de casos de uso do sistema.

5.5.1 Módulo *WebService*

Como dito anteriormente, existem várias ferramentas que tornam mais rápida e fácil a implementação de *WebServices*. Tais ferramentas, a partir da simples descrição das operações do *WebService*, são capazes de gerar todo o código necessário. Ficando a cargo desse trabalho somente a implementação da máquina de inferência e da base de dados.

A Figura 5.5 mostra o diagrama de classes para o módulo *WebService*. As Classes *DavisWSImpl* e *DavisWSSEI* são o *WebService* propriamente dito. A classe *mobileServlet*, gerada por ferramenta, é o ponto de comunicação entre o *WebService* e o cliente *Palm*. A Classe *DavisWS* é a máquina de inferência, que acessa diretamente a base de dados, representada pela classe *Conexao*.

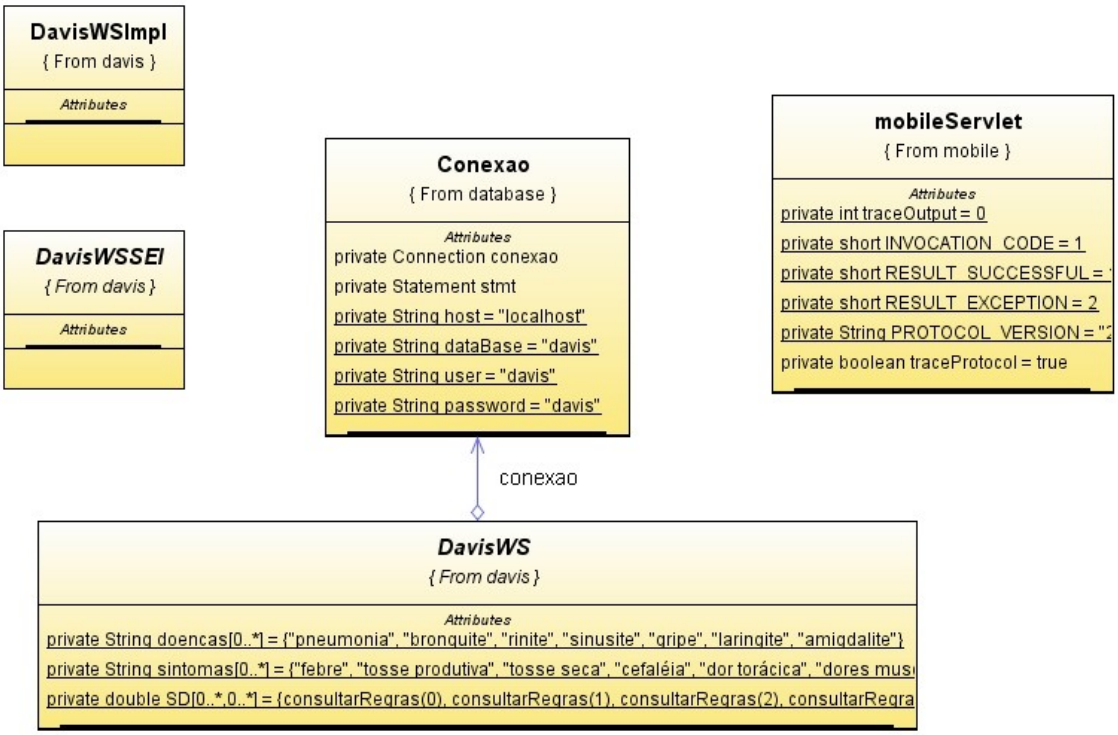

Figura 5.5: Diagrama de classes para o módulo WebService.

A Figura 5.6 apresenta o diagrama de seqüência para o módulo *WebService*. Nela é possível visualizar a seqüência de ações quando o cliente faz alguma requisição ao *WebService*. A classe *mobileServlet* recebe a requisição do *Palm* e envia para a classe *DavisWSImpl*. A classe *DavisWSImpl* se encarrega de enviar a requisição para a máquina de inferência *DavisWS* que acessa a sua base de dados para dar uma resposta á requisição.

Figura 5.6: Diagrama de seqüência para o módulo WebService.

5.5.2 Módulo Cliente

Também existem ferramentas que facilitam o desenvolvimento de clientes para *WebServices*, então no desenvolvimento da aplicação cliente levou-se em conta somente a interface com o usuário.

A Figura 5.7 mostra o diagrama de classes do módulo cliente. Esse diagrama é composto por apenas duas classes:

- *MobileXDAVISMIDlet* – é a interface com o usuário do *Palm*;
- *MobileXDAVIS* – é gerada por ferramenta e se comunica diretamente com a classe *mobileServlet* do módulo *WebService*.

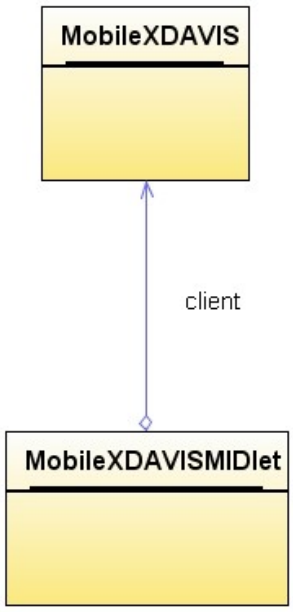

Figura 5.7: Diagrama de classes para o módulo cliente.

A Figura 5.8 apresenta o diagrama de seqüência para o módulo cliente. Nele, a classe *MobileXDAVISMIDlet* envia o comando do usuário para a classe *MobileXDAVIS*, que solicita uma resposta ao módulo *WebService*.

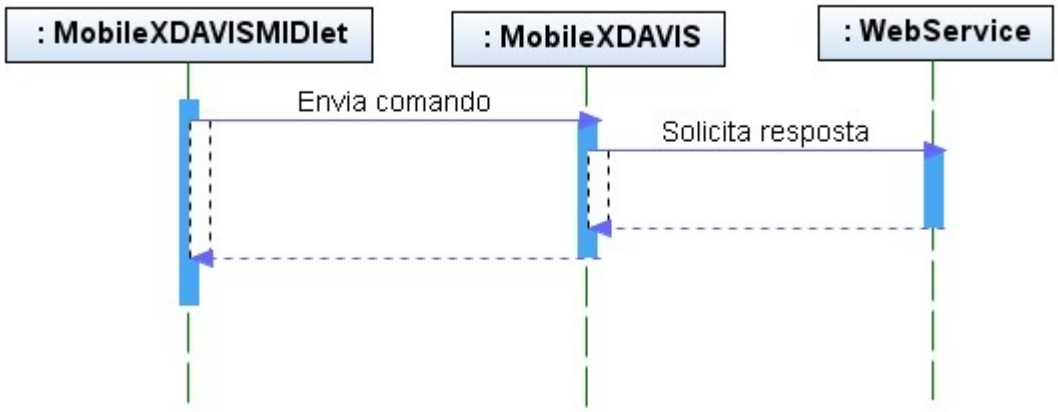

Figura 5.8: Diagrama de seqüência para o módulo cliente.

5.6 Apresentação do MobileX-DAVIS

Como demonstração, será utilizado o paciente P_1 conforme a Tabela 5.2 (página 45).

Para execução dos testes, foram utilizados o *PALM OS Simulator* e um microcomputador com o *Sun Java System Application Server PE 8.2* para a execução do *WebService*.

Para suportar o MobileX-DAVIS, o *Palm* deve possuir a máquina virtual IBM.

A Figura 5.9 apresenta o simulador de *Palm* com a máquina virtual e o MobileX-DAVIS já instalados. A seta vermelha indica o MobileX-DAVIS.

Na Figura 5.10 é apresentada a tela inicial do MobileX-DAVIS, com as opções: Inserir paciente, Consultar histórico do paciente, efetuar diagnóstico do paciente, calibrar amigdalite, gripe, laringite, sinusite, pneumonia bronquite e rinite. Além dessas opções existem os botões para sair e Alterar a URL do *WebService*.

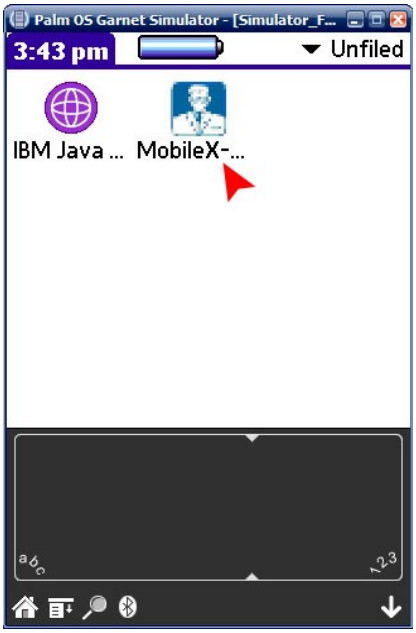

Figura 5.9: MobileX-DAVIS já instalado no Palm.

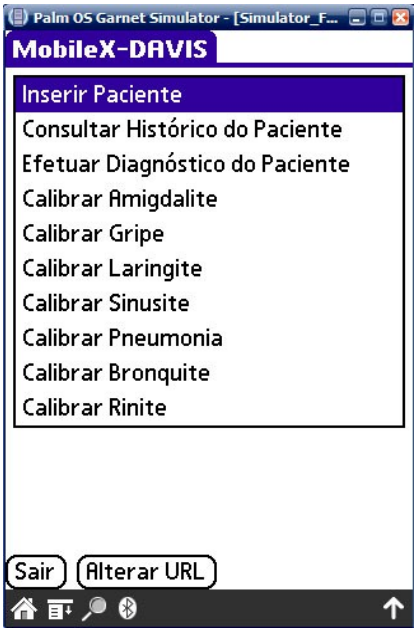

Figura 5.10: Tela inicial do MobileX-DAVIS.

5.6.1 Configurando a URL do *WebService*

É possível alterar a URL do *WebService* caso ele esteja hospedado em um servidor diferente. Como mostrado na Figura 5.11.

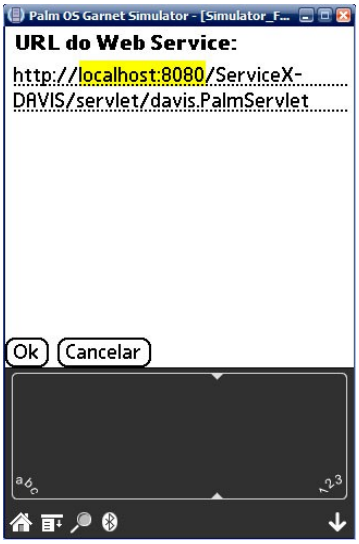

Figura 5.11: Alterando a URL do WebService.

5.6.2 Inserir Paciente

A Figura 5.12 mostra a tela para cadastro dos dados de pacientes. A Figura 5.13 mostra tela após o paciente ser inserido.

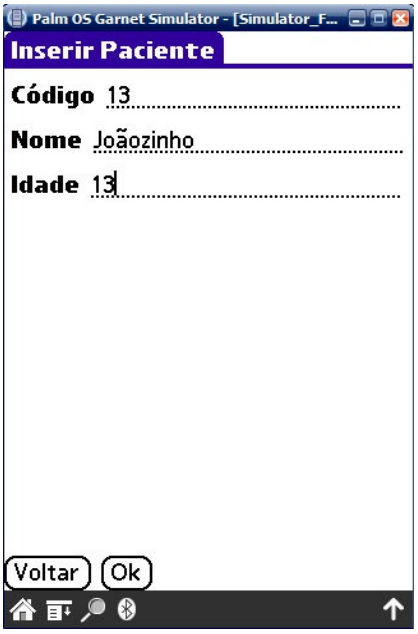

Figura 5.12: Inserindo paciente.

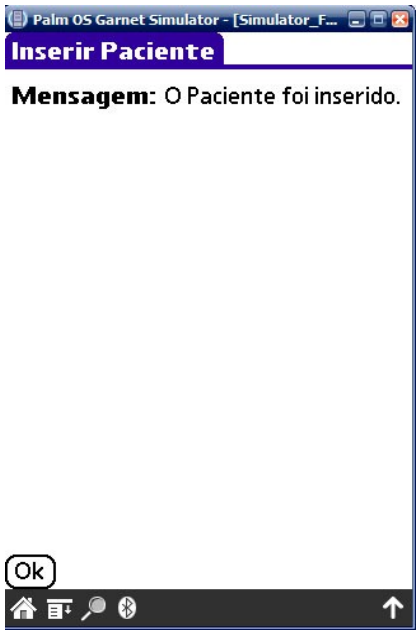

Figura 5.13: Paciente inserido

5.6.3 Efetuando Diagnóstico do Paciente

Antes de examinar o paciente, o médico ou responsável deve informar o código do paciente conforme a Figura 5.14. Assim, se o paciente existir, poderá ser feito o diagnóstico e a tela da Figura 5.15 será exibida.

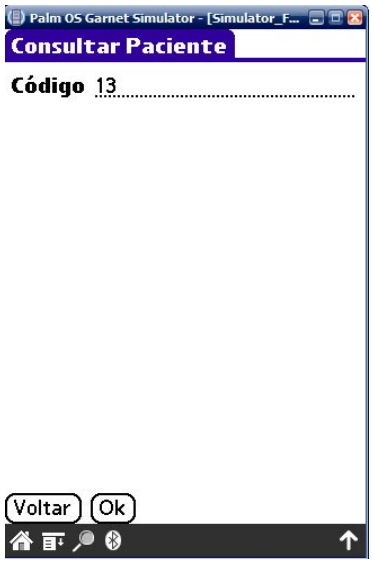

Figura 5.14: Informando código do paciente.

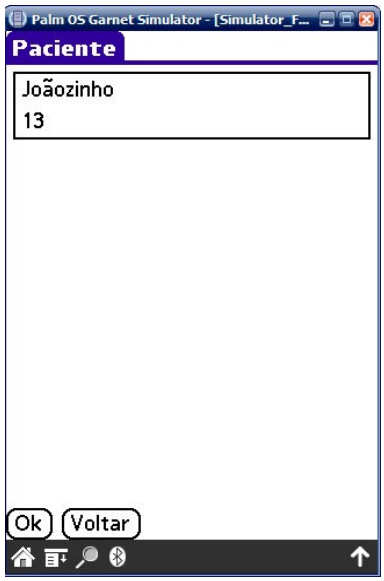

Figura 5.15: O paciente é exibido na tela.

Na Figura 5.16, o médico ou responsável deve informar o grau dos sinais e dos sintomas através de números reais entre 0 e 1. Após o *WebService* devolver os resultador, os dados serão exibidos na tela do *Palm* como indicado na Figura 5.17. Pela sugestão de diagnóstico do MobileX-DAVIS, o paciente P_1 apresenta maior possibilidade de estar com laringite e, segundo, o paciente P_1 foi diagnosticado com laringite. Ao confirmar, o diagnóstico é adicionado ao histórico do paciente.

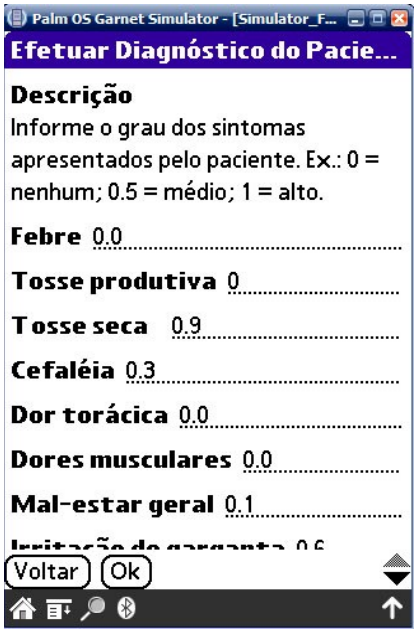

Figura 5.16: Informando grau dos sinais e dos sintomas.

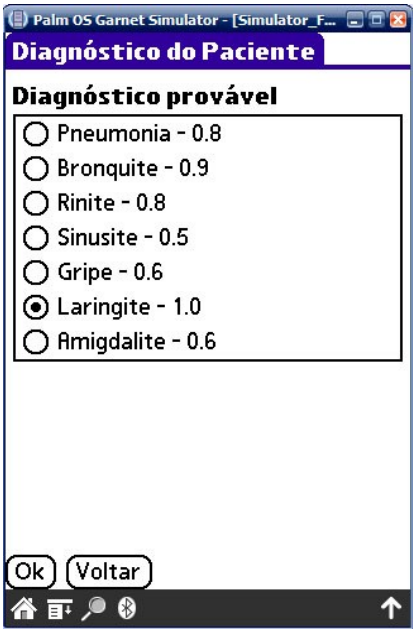

Figura 5.17: Sugestão de diagnóstico do sistema.

5.6.4 *Consultar Histórico do Paciente.*

Assim que o diagnóstico é confirmado pelo médico, este é armazenado no histórico do paciente. A Figura 5.18 mostra o histórico do paciente Joãozinho.

Figura 5.18: Histórico do paciente Joãozinho.

5.7 Considerações Finais

A resposta da composição é também um conjunto *fuzzy*, ou seja, a composição nem sempre responde qual doença o paciente possui. A composição *fuzzy* fornece a distribuição de possibilidades do paciente no conjunto de doenças.

Uma propriedade importante da relação *fuzzy* é, após ter diagnósticos de novos pacientes, estes podem ser incluídos na base de conhecimentos e, assim, aumentar a capacidade de obter mais diagnósticos por meio da relação *fuzzy* R, tal como faz o médico.

6 CONSIDERAÇÕES FINAIS

6.1 Conclusões

A tecnologia móvel não é apenas uma invenção, ela pode ser considerada uma revolução, pois foi capaz de atingir o cotidiano das pessoas e fazer parte da vida delas, modificando suas rotinas e formas de tomar decisões. Muitas pessoas não vivem sem celular, outras estão 24 horas disponíveis e as encontramos em qualquer lugar, algumas não abrem mão de estarem com seu *Palm* conectado na Internet e ao mesmo tempo se deslocando pela casa. Assim, as tecnologias móveis são, atualmente, muito úteis, pois dão maior mobilidade para a execução de tarefas sem que seja necessário utilizar uma estação que faça todo o processamento.

Além disso, os *WebServices* permitem a integração entre sistemas e compatibilidade de aplicações. Com esta tecnologia é possível que novas aplicações possam interagir eficientemente com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. E ainda tornam possíveis aplicações no modelo cliente-servidor com cliente móvel. Isso torna-se muito interessante, pois pode deixar todo o processamento pesado para um equipamento mais potente, uma vez que operações com ponto flutuante são mais caras do ponto de vista computacional.

Por fim, um sistema especialista que aprende à medida que vai sendo utilizado é de difícil concepção, pois envolve todo um aparato matemático para alterar corretamente a sua base de dados.

6.2 Contribuições

Existem muitas razões pelas quais os sistemas especialistas não estejam sendo usados rotineiramente. Alguns exigem a existência prévia de um sistema de registro médico computadorizado para fornecer os dados, e muitas instituições ainda não têm seus dados disponíveis eletronicamente. Outros são prejudicados por um *design* pobre da interface com o usuário, e deste modo nunca são usados, embora tenham benefícios. Porém, o módulo *WebService* do MobileX-DAVIS não precisa necessariamente estar ligado à instituição que o utilizará, ele poderá, por exemplo, ser disponibilizado por uma agência reguladora. E o módulo cliente para *Palm* possui um interface intuitiva.

Há muita relutância em usar sistemas especialistas, simplesmente pelo fato de que os sistemas especialistas não se adaptam facilmente ao processo de assistência médica, o que exige um esforço adicional por parte dos profissionais médicos, já bastante ocupados. Entretanto, um dos objetivos do MobileX-DAVIS é propiciar uma maior mobilidade aos médicos.

6.3 Trabalhos Futuros

Como trabalho futuro, pode-se desenvolver mais o MobileX-DAVIS, a ponto que ele, além do diagnóstico das doenças das vias aéreas superiores e inferiores, também faça o diagnóstico de outras classes de doenças, pois o método de Mamdani, usado na implementação da máquina de inferência, pode ser usado para muitos tipos de problemas.

Outra funcionalidade seria armazenar quantas vezes o diagnóstico do sistema foi substituído pelo diagnóstico do médico, possibilitando que o MobileX-DAVIS pudesse alterar as regras de maneira automática para melhorar o diagnóstico, aprendendo à cada diagnóstico, sem que seja necessário intervenção do especialista nas regras do sistema.

Também é possível, devido ao uso de *WebServices*, o desenvolvimento de clientes que não sejam móveis. Clientes *desktop,* para uso em computadores pessoais, e clientes para rede ou *Web*, para uso em hospitais e clínicas.

Além disso, pode-se utilizar o método de Mamdani para desenvolver outros sistemas especialistas, não somente para outras classes de doenças, mas para outros áreas que não a medicina.

REFERÊNCIAS BIBLIOGRÁFICAS

[BARRETO, 2002] BARRETO M.J.. Introdução as Redes Neurais e Artificiais. http://www.lia.ufc.br/~eti/2003/menu/modulos/RNA/RNA-RedesNeuraisArtificiaisI.pdf. Agosto de 2005.

[BARROS e BASSANEZI, 2001] BARROS L. ,BASSANEZI R.; "Introdução à teoria fuzzy aplicações em biomatemática". 2001.

[BITTENCOURT, 2001] BITTENCOURT Guilherme; "Inteligência Artificial - Ferramentas e Teorias". Editora da UFSC, 2001.

[CAWSEY, 2005] CAWSEY A.. Expert Systems. http://w3.ualg.pt/~hshah/expert-system/node1.html. Maio de 2005.

[CERAMI, 2002] Cerami E.; "Web Services Essentials – Distribuited Applications with XML-RPC, SOAP, UDDI & WSDL". Ed. O'Reilly, 2002.

[CERTISIGN, 2006] CertiSign Certificadora Digital S.A.. . http://www.certisign.com.br/suporte/glossario/a.jsp. Agosto de 2006.

[CHAPPEL, JEWELL, 2002] CHAPPEL, D., JEWELL, T.; "Java Web Services". Ed. O' Reilly, 2002.

[CRIPPA, 2005] CRIPPA M.. Sistemas Especialistas: A Engenharia do Conhecimento Aplicada as Organizações. http://prof-esag.udesc.br/demo/trabalhos/alunos/mc/aplica.html. Junho de 2005.

[DEITEL et al, 2003] DEITEL H.M., DEITEL P.J., GADZIK J.P., LORNLI K., SANTRY S.E., ZHANG S.; "Java Web Services For Experienced Programmers". Ed. Deitel, 2003.

[ESPERANÇA, 2006] ESPERANÇA, Claudio. Python: Tupllas e Strings. http://www.lcg.ufrj.br/Cursos/algprog/Programando%20em%20Python%20-%20Tuplas%20e%20Strings.pdf. Agosto de 2006.

[FEIGENBAUM, ENGELMORE, 1993] FEIGENBAUM E., ENGELMORE Robert S.. "Expert Systems And Artificial Intelligence. http://www.wtec.org/loyola/kb/c1_s1.htm. Maio de 2005.

[JCP, 2006] Java Community Process. Web Services Specification. http://jcp.org/aboutJava/communityprocess/review/jsr172/index.html. Julho de 2006.

[KEOGH, 2003] KEOGH J.; "J2ME - The Complete Reference". Ed. Osborne, 2003.

[KRISHNAMOORTHY, RAJEEV, 1996] Krishnamoorthy C.S., Rajeev S.; "Artificial Intelligence and Expert Systems for Engineers". Ed. CRC Press, 1996.

[LAURENT et al, 2001] Laurent S., Johnston J., Dumbill E.; "Programming Web Services With XML-RPC". Ed. O'Reilly, 2001.

[LOPES et al, 2005] LOPES W. A., JAFELICE R. S. M., BARROS L. C.; "Modelagem Fuzzy de Diagnóstico Médico e Monitoramento do Tratamento da Pneumonia". 2005.

[LUGER, 1998] Luger, Stubblefield; "Artificial Intelligence; Structures And Strategies For Complex Problem Solving". Ed. Addison Wesley, 1998.

[MERCK, 2006] Merck Sharp & Dohme Farmacêutica Ltda.. Distúrbios dos Pulmões e das Vias Aéreas. http://www.msd-brazil.com/msdbrazil/patients/manual_Merck/mm_sec4_31.html. Agosto de 2006.

[RABELO Jr. et al, 1993] RABELO Jr. A., ROCHA A.R., SOUZA A.D., XIMENES A.A., LOBO N., CARVALHO D., WERTHER J.C.S. Filho, OLIVEIRA K.M., SOUZA L.A., WERNECK V.M.. Um Sistema Especialista para Diagnóstico de Cardiopatias

Isquêmicas. http://www.informaticamedica.org.br/informed/isquem.htm. Julho de 2006.
[REHM, 2005] MARCUS VINICIUS GALVÃO REHM; "Sistema de Apoio ao Diagnóstico Médico utilizando Tecnologias Móveis". 2005.
[RUSSEL, NORVIG, 1995] Russel S. J., Norvig P.; "Artificial Intelligence – A Modern Approach". Ed. Prentice Hall, 1995.
[SAVARIS, 2002] Savaris S.V.A.M.Sistema Especialista para primeiros socorros para cães. 2002.
[SILVA, PARIZE, 1995] SILVA Roberto, PARIZE Marisa M.G.. NIACIN: Um Programa para o Desenvolvimento de Sistemas Especialistas em Medicina . http://www.informaticamedica.org.br/informed/niacin.htm. Julho de 2006.
[SUN, 2005] Sun Microsystems, Inc.. Java 2 Plataform, Micro Edition, Data Sheet. http://java.sun.com/j2me/j2me-ds.pdf. Março de 2005.
[SUN, 2005 a] Sun Microsystems, Inc.. CDC: JAVA™ PLATFORM TECHNOLOGY FOR CONNECTED DEVICES. http://java.sun.com/products/cdc/wp/cdc-whitepaper.pdf. Julho de 2006.
[SUN, 2006 b] Sun Microsystems, Inc.. Connected Limited Device Configuration 1.1. jcp.org/aboutJava/communityprocess/final/jsr139. Junho de 2006.
[SUN, 2006 c] Sun Microsystems, Inc.. Java Technology Overview. http://java.sun.com/overview.html. Junho de 2006.
[SUN, 2006 d] Sun Microsystems, Inc.. Java ME - Micro App Development Made Easy. java.sun.com/javame/. Junho de 2006.
[SUN, 2006 g] Sun Microsystems, Inc.. Mobile Information Device Profile. java.sun.com/products/midp/midp-ds.pdf. Junho de 2006.
[SUN, 2006 h] Sun Microsystems, Inc.. SOFTWARE JAVA para seu computador. http://java.com/pt_BR/download/index.jsp. Agosto de 2006.
[TIDWELL et al, 2001] TIDWELL D., SNELL J., KULCHENKO P.; "Programming Web Services with SOAP". Ed. O'Reilly, 2001.
[TOPLEY, 2002] TOPLEY K.; "J2ME in a Nutshell. A desktop quick reference". Ed. O'Reilly, 2002.
[W3C, 2005] BOOTH D., HAAS H., McCABE F., NEWCOMER E., CHAMPION M., FERRIS C.,ORCHAND D. Web Services Architecture. www.w3.org/TR/ws-arch. Julho de 2005.
[WIKIPÉDIA, 2006] Wikipédia, a enciclopédia livre.. Serviço Web. http://pt.wikipedia.org/wiki/Web_service. Junho de 2006.
[WIKIPÉDIA, 2006 c] Wikipédia, a enciclopédia livre. Jar. http://pt.wikipedia.org/wiki/Jar. Julho de 2006.
[WIKIPÉDIA, 2006 d] Wikipédia, a enciclopédia livre. Interface Gráfica. http://pt.wikipedia.org/wiki/Gui. Julho de 2006.
[WIKIPÉDIA, 2006 e] Wikipedia, the free encyclopedia. Abstract Window Toolkit. http://en.wikipedia.org/wiki/Awt. Julho de 2006.
[WIKIPÉDIA, 2006 f] Wikipedia, the free encyclopedia. Xlet. http://en.wikipedia.org/wiki/Xlets. Julho de 2006.
[WIKIPÉDIA, 2006 g] Wikipédia, a enciclopédia livre. Applet. http://pt.wikipedia.org/wiki/Applet. Julho de 2006.
[WIKIPÉDIA, 2006 h] Wikipédia, a enciclopédia livre. JavaBeans. http://pt.wikipedia.org/wiki/JavaBeans. Julho de 2006.