UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

SISTEMA ESPECIALISTA PARA TELEDIAGNÓSTICO DE MALÁRIA NO AMBIENTE DE WEB SERVICES

WST Web Services Telediagnosis

ANTONIO LUIS DO RÊGO LUNA FILHO

SÃO LUIS – MARANHÃO – BRASIL
2005

SISTEMA ESPECIALISTA PARA TELEDIAGNÓSTICO DE MALÁRIA NO AMBIENTE DE WEB SERVICES

WST Web Services Telediagnosis

Por

ANTONIO LUIS DO RÊGO LUNA FILHO

Dissertação de Mestrado submetida à Coordenação do curso de Pós-Graduação em Engenharia de Eletricidade da UFMA Universidade Federal do Maranhão, como parte dos requisitos para obtenção do título de Mestre em Engenharia de Eletricidade, com área de concentração em Ciências da Computação.

SÃO LUIS – MARANHÃO – BRASIL

2005

SISTEMA ESPECIALISTA PARA TELEDIAGNÓSTICO DE MALÁRIA NO AMBIENTE DE WEB SERVICES

WST Web Services Telediagnosis

MESTRADO

Área de Concentracão: CIÊNCIAS DA COMPUTACÃO

ANTONIO LUIS DO RÊGO LUNA FILHO

Orientador: Prof. Dr. Sofiane Labidi

Co-Orientador: Prof. Dr. Zahir Abdelouhab

Curso de Pós-Graduação em Engenharia de Eletricidade

Universidade Federal do Maranhão

SISTEMA ESPECIALISTA PARA TELEDIAGNÓSTICO NO AMBIENTE DE WEB SERVICES

WST Web Services Telediagnosis

ANTONIO LUIS DO RÊGO LUNA FILHO

DISSERTAÇÃO APROVADA EM ___/___/2005

Prof. Dr. Sofiane Labidi
(Orientador)

Prof. Dr. Zahir Abdelouhab
(Co-Orientador)

Profa. Dra. Rossana Maria de Castro Andrade

Prof. Dr. Aristófanes Correia Silva

Geralmente as pessoas epigrafam frases de personalidades célebres.

Modestamente eu vou ousar neste trabalho epigrafar uma frase cotidiana das madrugadas de quem passa pela angústia da maturação do conhecimento...como nós.

“Puxa vida meu Deus eu vou conseguir...”

Autor: todos nós.

AGRADECIMENTOS

Seria injusto não agradecer a todas as pessoas que me cercam no dia-a-dia. Aquelas que direta ou indiretamente fazem parte do meu contexto. Sintam-se calorosamente agradecidas.

No entanto existem algumas pessoas que não necessariamente na ordem que citarei devem se sentir eternamente e especialmente agradecidas por mim. Agradecidas não só por este trabalho, mas por onde eu modestamente estou e para onde certamente eu vou.

Um grande abraço a Maria Luiza (minha mãe sempre viva na minha memória), a Ana Teresa e Mariana (puxa bens como nós temos lutado), a papai Urbano e Leyr (sou um afortunado, tenho dois pais e duas mães), a Popó (pelo incentivo para eu ser alguém), a Sofiane e Zahir (mais que orientadores grandes amigos), a Maria da Guia (pelo crédito), a Casanova e Fernando (bons companheiros nesta jornada), não posso me esquecer de Dona Elza (esta sim segurou cada onda), Bito e Silvana (o meu afeto), ao Núcleo de Parasitologia da UFMA na pessoa dos Profs. Rafael e Heloisa (pela orientação da parte clínica deste trabalho), a Anurag Bela (por me fazer entender o que é clareza e compreensão, viver e conviver, etc.), enfim estes estiveram muito perto de mim, como um verdadeiro sustentáculo nestes últimos anos da minha vida.

DEDICATÓRIA

Ai que saudades que eu tenho da aurora da minha vida da minha infância querida que os dias não trazem mais...

Casimiro de Abreu

Esse título é para minha mãe.

A pessoa que apesar do seu estado senil, não perdeu a característica mais fascinante da mulher – a maternidade. Sempre quis me ver grande.

Um grande beijo bichim.

RESUMO

Este trabalho aborda a telemedicina através da área de estudo sobre o telediagnóstico.

O maior objetivo é desenvolver um mecanismo de inferência utilizando uma ferramenta chamada JESS Java Expert System Shell, para diagnosticar malária, disponibiliza-lo como serviço no ambiente da internet através de um *framework* para construção de *Web Services*, para que se possa oferecer uma solução às regiões menos favorecidas.

Algumas tecnologias irão dar suporte ao trabalho, como o software Protegé, que será usado na modelagem do conhecimento acerca da malária, através de ontologias e a UML Unified Modeling Language para modelagem do protótipo do mecanismo de inferência.

ABSTRACT

This work approaches the telemedicine through an area of study about telediagnosis. The solution proposed here will be the development of an expert system which has the objective of sharing rule base in an internet environment, using the web services technology. The knowledge basis of expert system will be built on malaria diseases. Some technologies such as: the JESS - Java Expert System Shell, the Java language and also Protegé tool to ontologies generations. The major interest of this project is to attend communities located far from cities giving them accurate medical support.

LISTA DE FIGURAS

Página

SIGLAS

AIM	*Advanced Informatics in Medicine*
ATM	*Asynchrony Transference Mode*
API	*Application Programming Interface*
CASE	*Computer Aided Software Engeneering*
CB	*Conhecimento Bruto*
CC	*Coleta de Conhecimento*
CEE	*Comunidade Econômica Européia*
CLIPS	*C Language Integrated Production System*
CORBA	*Common Object Request Broker Architecture*
DDT	*Dimetil Difenil Tricloretano*
DOM	*Document Object Model*
ES	*Expert System*
EC	*Engenheiro de Conhecimento*
ECG	*Eletrocardiograma*
ERP	*Enterprise Resource Programming*
FC	*Fonte de Conhecimento*
FUNAI	*Fundação Nacional do Índio*
FUNASA	*Fundação Nacional de Saúde*
GPS	*General Problem Solver*
GUI	*Guide User Interface*
HTTP	*Hyper Text Transfer Protocol*
IA	*Inteligência Artificial*
IMEX	*Integrated Malária Expert System*
JDK	*Java Development Kit*
J2EE	*Java 2 Enterprise Edition*
JESS	*Java Expert System Shell*
JSP	*Java Server Pages*
JVM	*Java Virtual Machine*
KA	*Knowledge Acquisition Phase*
KBS	*Knowledge Base System;*
KR	*Knowledge Representation Phase*
LPO	*Lógica de Primeira Ordem*

MC	*Modelagem do Conhecimento*
MCo	*Modelo Conceitual*
MDI	*Multiply Document Interface*
OAV	*Object-attribute-value triplets*
OMS	*Organização Mundial de Saúde*
SE	*Sistema Especialista*
SEAMED	*Sistema Especialista para Área Médica*
SEDIL	*Sistema de Diagnóstico Médico da doença Leishimaniose*
SQL	*Strucutured Query Language*
SOAP	*Simple Object Access Protocol*
RPC	*Remote Procedure Call*
TELECOS	*Tele-consulta*
UDDI	*Universal Description, Discover and Integration*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
UTI	*Unidade de Terapia Intensiva*
VB	*Visual Basic*
XML	*eXtended Markup Language*
W3C	*World Wide Web Consortium*
WST	*Web Services Telediagnosis*
WSDL	*Web Services Description Language*
WSIF	*Web Service Invocation Framework*

TABELAS

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS

LISTA DE TABELAS

Capítulo 1 Introdução

Neste capítulo será visto:

- A introdução desta dissertação;
- O cerne do projeto de telediagnóstico;
- Quais os objetivos deste projeto;
- As fases de desenvolvimento do projeto;
- A relevância da abordagem usada na solução do problema;
- O conteúdo de cada capítulo.

1 INTRODUÇÃO

Um número cada vez maior de empresas e profissionais demanda informações gerenciais em forma de conhecimento. Através dessa informação contextualizada, o conhecimento constrói um valor incomensurável em tais organizações: a memória funcional e o capital de conhecimento.

De meados da década de 40, logo após a guerra alguns fenômenos comportamentais vêm provocando uma verdadeira revolução no mundo, que fizeram inclusive com que aumentasse em quase 1/3 a população mundial em menos de 10 anos [32].

Com esse crescimento populacional a demanda por interesses até então inusitados, provocou a evolução rápida de indústrias como a de **informação.** Esta com o advento dos computadores, das telecomunicações e das redes de longa distância.

Uma das marcas deste processo evolutivo que vem ocorrendo desde 1945, é o interesse pelo estudo da inteligência. Nunca este tema foi investigado com tamanha profundidade, certamente pelo fato de que o homem tenha se encontrado em um estado limítrofe de compreensão dos seus processos de inferência.

Surge então na década de 60 um ramo novo na ciência - a Inteligência Artificial ou IA - e em torno das pesquisas neste ramo, alguns seguimentos de estudo geraram artefatos como os sistemas de monitoramento por computador que vieram dar suporte a uma infinidade de práticas profissionais, uma delas a telemedicina [22], que surgiu com o projeto Mercury um dos primeiros da agência espacial americana entre 1960 e 1964 e logo foi integrada aos sistemas especialistas (i.e., MYCIN – sistemas especialista para diagnóstico médico).

Na década de 70 com o significativo surgimento das tecnologias na área de telecomunicações e com a expansão das redes de computadores através da internet a telemedicina iniciou um processo de evolução e consolidação rápida.

Nos dias atuais a telemedicina ramificou-se em várias áreas de estudo como: o telemonitoramento, a teleterapia, o **telediagnóstico**, a teledidática, a teleanálise entre outras.

Uma significativa quantidade de projetos atualmente dão origem a pesquisas principalmente no campo do telediagnóstico, visto que, este é a ferramenta principal na prática médica em busca da cura de uma patologia.

O telediagnóstico, como catalisador dos procedimentos clínicos acerca da terapêutica humana, comprova sua eficiência em artefatos como: o LEPIDUS-ON LINE [24], HealthNet [3] e o próprio *WST – Web Services Telediagnosis* objeto deste projeto, usando a tecnologia de *Web Services*.

Telediagnóstico é um segmento de estudo da telemedicina que aborda não só a identificação direta de uma patologia através da aplicação de uma ferramenta informática e/ou de telecomunicações, mas em uma ótica mais conseqüente, provê um suporte detalhado à decisão médica.

Os sistemas especialistas quando aplicados ao telediagnóstico, via de regra, tem o papel de coleta, tratamento e transformação de informação em conhecimento, a partir de um domínio.

1.1 O cerne deste projeto

A fusão telediagnóstico/sistema especialista, veiculado na *web* através de um ambiente de publicação de serviços, inteiramente portável, origina o *WST – Web Services Telediagnosis* um sistema especialista para telediagnóstico usando a tecnologia de *Web Services* – escopo desta dissertação.

1.2 O objetivo do projeto

Primordialmente a intenção deste telediagnóstico como um todo será propor uma contribuição de acessibilidade do ponto de vista do usuário final, visto que, todo ele será desenvolvido no ambiente da rede mundial de computadores (Internet) e interoperabilizando aplicações legadas de processos e controle da malária, independente de plataforma através de *Web Services* e atender conseqüentemente - dada toda esta viabilidade - as regiões mais inóspitas onde hoje se encontram focos endêmicos da malária.

No entanto, o *WST – Web Services Telediagnosis,* apesar de ter o foco voltado para o diagnóstico da malária, é um sistema especialista que poderá ser aplicado a qualquer doença que se manifeste em um cenário de poucos recursos.

Este sistema especialista **em sua primeira fase de desenvolvimento** possui especificamente dois objetivos primordiais: (a) a segunda opinião médica no diagnóstico da malária e; (b) o compartilhamento de conhecimento acerca desta doença através do provimento de serviços *web*.

1.3 As fases do projeto

O projeto *WST – Web Services Telediagnosis* esta dividido em três fases bem definidas:

1. A primeira de modelagem do domínio de conhecimento acerca da doença tropical malária e implementação deste domínio no ambiente de serviço *web* através de um sistema especialista;

2. A segunda de construção de mecanismos de atualização das bases de regras implementadas na fase 1, objetivando um compartilhamento de conhecimento acurado à segunda opinião médica;
3. A terceira de construção de interfaces essencialmente amigáveis e interativas visando o uso direto da solução pelos mais variados perfis de usuários.

O *WST – Web Services Telediagnosis*, é um protótipo que em sua primeira fase, a partir do ciclo de vida de um *KBS - Knowledge Based System*[1], fará todo o levantamento de conhecimento acerca do domínio da malária chegando ao vetor de contágio da doença.

A partir deste ponto, através de ontologias usando o software *Protegé* [21] serão elaborados os diagramas com as entidades identificadas para comporem as associações semânticas (mosquito, *plasmodium*, homem).

As associações ontológicas envolvendo as entidades do vetor de contágio (mosquito, *plasmodium*, homem) darão suporte, por sua vez, a elaboração das regras que irão compor os mecanismos de inferência do sistema, e conseqüentemente serão aplicadas a fatos extraídos de bases de conhecimentos disponibilizadas como serviços *web*.

Comumente nas organizações do âmbito governamental que tratam processos de endemias como a malária (i.e FUNASA – Fundação Nacional de Saúde, FUNAI – Fundação Nacional do Índio), existem sistemas nomeados neste trabalho como legados que processam de alguma forma informações acerca da doença. Legados, visto que, o enfoque é interoperabilizar o que já existe em funcionamento com o sistema especialista proposto – *WST – Web Services Telediagnosis* – criando desta maneira um ambiente único de interação no processo de diagnóstico.

Um sistema chamado *Medic*, desenvolvido em paralelo ao *WST* e seguindo os padrões da programação estruturada – outra peculiaridade da maioria das organizações – utilizando a linguagem de programação Visual Basic e um banco de dados SQL Server será usado na prototipação desta integração com o objetivo de torná-la mais próxima da realidade.

Este sistema – *Medic* – possui um fluxo de atendimento clínico (cadastro do paciente, marcação da consulta, anamnese, etc.) com base no que foi observado na maioria dos artefatos para este fim. Será criado dentro deste sistema um mecanismo (i.e. botão de acesso) que capture ou permita que o usuário submeta as observações de uma anamnese da malária ao mecanismo de inferência disponível como serviço *web*.

[1] KBS Knowledge Based System são sistemas baseados no conhecimento a partir de um domínio.

Um outro sistema, desta vez o *WST – Web Services Telediagnosis* desenvolvido a partir da metodologia de orientação a objeto, utilizando a linguagem Java e o pacote para desenvolvimento de sistemas especialistas *JESS Java Expert System Shell* será o responsável pela submissão dos fatos às regras do mecanismo de inferência retornando uma resposta acerca do diagnóstico do paciente.

Este último sistema será disponibilizado como serviço *web* alcançando desta forma o objetivo da primeira fase do projeto que é criar um ambiente interativo / cooperativo em que o homem e a máquina interagem no ambiente de *Web Services*.

1.4 A relevância da abordagem do projeto

É importante que se leve em consideração para a contribuição deste projeto, os aspectos técnicos e sociais que o envolvem.

A telemedicina como já comentado anteriormente é um instrumento tecnológico de indiscutível aplicabilidade nos processos de automação da prática médica. No entanto é necessário que se confronte alguns sistemas especialistas que objetivem a segunda opinião, para que se possa ter uma postura crítica.

Pelo que se tem investigado a última contribuição tecnológica acerca de sistemas especialistas, especificamente para a cura de malária como é o proposto nesta dissertação, foi um artefato desenvolvido nos laboratórios do MIT Massachussets Institute of Tecnology chamado IMEX em 1990. O IMEX System [] como é conhecido tem o objetivo de agir como uma ferramenta no diagnóstico e tratamento da malária, assim como, melhorar o gerenciamento da doença e dos seus pacientes.

Em face dos diversos sistemas especialistas que tem como escopo o apoio à decisão médica no processo de diagnóstico, define-se abaixo um quadro comparativo (Tabela 1.1), que leva em consideração o critério do uso da tecnologia de serviços *web*: O sistema IMEX, apesar da citação e de ser um sistema especialista específico para malária, não poderá ser confrontado, visto que, a contribuição deste trabalho esta calcada em sistemas especialistas que pelo tenham sido idealizados no interstício de tempo das tecnologias pospostas para o *WST*.

O HEALTNet [3], o LEPIDUS [24]e o SEDIL [16], são os três sistemas especialistas que tem servido de parâmetro inicial para uma comparação com o que se propõe o WST – Web Services Telediagnosis – o sistema em questão.

SE[2] características	HEALTH Net	*LEPIDUS*	SEDIL
1.O SE foi desenvolvido para o ambiente da internet ?	Sim	Sim	Não
2. O SE permite que suas bases de conhecimento sejam disponibilizadas no ambiente de serviços *web* ?	Não	Não	Não
3. O SE contemplará aprendizagem automática pelas bases de regras ?	Não	Não	Não
4. O SE tem suporte a linguagem Java ? (Portabilidade)	Sim	Não	Não

Tabela 1.1 Aspectos comparativos entre sistemas especialistas

Os motivos que levaram a escolher os referidos sistemas confrontados na Tabela 1.1 são:

1. aplicabilidade ao telediagnóstico;
2. possibilidade de comparar as vantagens e as desvantagens baseadas nas características peculiares de cada um;
3. facilidade em usar bases de regras no ambiente de *Web Services*.

As primeiras observações são as seguintes:

1. Sistema Especialista para telediagnóstico que contempla consulta a serviços publicados no ambiente da internet (*Web Services*) viabiliza uma resposta de cura muito maior ao paciente, visto que, o potencial de conhecimento veiculado na rede por parte do especialista é maior.
2. A linguagem Java tem sido uma tendência neste ambiente, ainda que não em detrimento a outras tecnologias.
3. O mecanismo de aprendizagem, a ser implementado na fase 2 do projeto, pretende garantir o suporte à atualização constante dos conhecimentos a serem absorvidos pela fase 1 do projeto.
4. A utilização de um motor de inferência sobre as bases de regras compartilhadas na *web* proporciona ao especialista um espectro de conhecimento que os sistemas locais não oferecem.

Mais que todas as justificativas que fazem referência aos aspectos técnicos, vale ressaltar a importância do foco do mesmo voltado ao auxílio à cura da malária. E como principal argumento tem o fato de que a malária é uma das doenças mais antigas que a humanidade conhece e que infelizmente depende de ações efetivas de organismos do poder

[2] SE Sistema Especialista

público para sua erradicação. O que de certa forma leva os quadro de endemias a números muitas vezes alarmantes.

A malária como já foi comentado acima é uma doença histórica. Provavelmente tenha surgido na África, apesar de existirem registros desta em múmias egípcias com mais de 1000 anos.

Imagina-se que esta doença se alastrou para o resto do mundo por intermédio das viagens tans-continentais, especificamente no tempo das grandes explorações e colonizações do século XVI.

Embora na antiguidade, por volta do século V na Grécia, Hipócrates foi o primeiro a desmistificar esta doença, inclusive relatando todo o quadro clínico da mesma, até então envolta por superstições. A malária tem pelo menos 1500 anos de estagnação sobre seus aspectos evolutivos e seu processo de tratamento e cura [28]. Somente no século XIX através de muitas pesquisas clínicas, a malária voltou a ter algumas descobertas significativas a seu respeito.

A iniciativa para erradicação da doença era a eliminação do mosquito transmissor através de ações de aplicação de inseticida – isso foi possível graças ao DDT Dimetil Difenil Tricloretano descoberto em 1942 por um pesquisador chamado Paul Muller – que viabilizou um programa aprovado mundialmente pela OMS Organização Mundial da Saúde.

Embora este tenha sido um passo positivo a falta de continuidade e controle de ações desta natureza fez com este quadro se deteriorasse na década de 1980 fazendo com que houvesse um crescimento significativo de casos da doença em todo o mundo.

A malária está presente em toda a região tropical e sub-tropical do planeta como se pode ver na Figura 1.1. O maior foco da doença é a África Sub-Sahariana, responsável por 90% dos casos.

A malária é endêmica em 53 países na África (incluindo 8 países ao sul), em 21 países nas Américas, 4 países na Europa e 14 na região leste do Mediterrâneo, e no sudeste Asiático.

A cada ano, ocorrem 300 a 500 milhões de casos, com cerca de 1 milhão de óbitos. Estes óbitos ocorrem na África, em áreas remotas com difícil acesso aos serviços de saúde. Dos 25 a 30 milhões de pessoas que viajam para áreas endêmicas, entre 10 a 30 mil contraem malária.

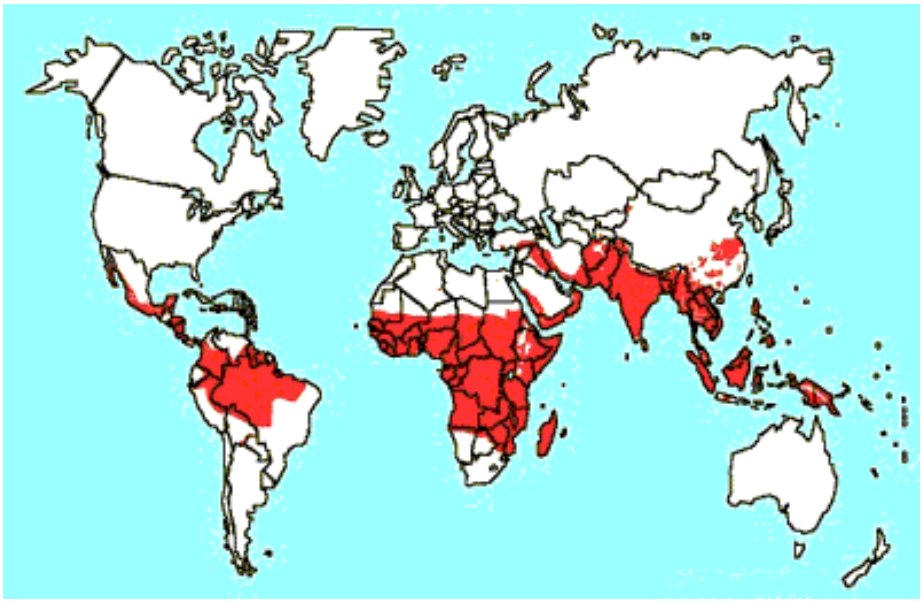


Figura 1.1 Regiões tropicais de incidência da malária no mundo.

No Brasil, a área endêmica é conhecida como Amazônia Legal. Esta área é composta pelos estados do Acre, Amapá, Amazonas, Maranhão, Mato Grosso, Mato Grosso do Sul, Pará, Rondônia, Roraima e Tocantins como se pode ver na Figura 1.2 inclusive com uma legenda que expressa o grau de risco de cada uma destas regiões.

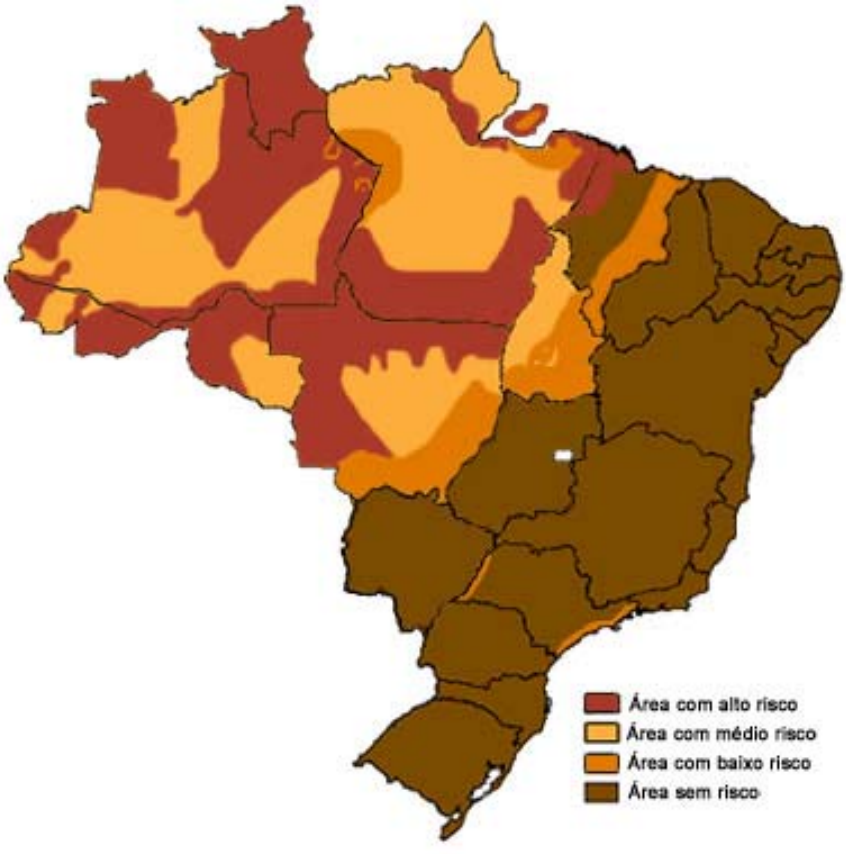


Figura 1.2 Regiões de incidência da malária no Brasil.

1.5 Conteúdo deste trabalho

Os capítulos deste primeiro momento do projeto *WST – Web Services Telediagnosis*, abordado neste trabalho se dividem a seguir:

- **Capítulo 1 – Introdução**

Apresentação da proposição do projeto como um todo – em suas três fases, abordando não só o telediagnóstico, mas situando a relevância deste dentro de um contexto de aplicação do conhecimento humano através dos sistemas especialistas.

- **Capítulo 2 – Estado da Arte**

O que existe sobre as pesquisas no campo do telediagnóstico. Quais as contribuições deste em um ambiente de compartilhamento e de serviços na rede mundial de computadores. As tecnologias utilizadas para construção desta primeira fase (Java, JESS, *Web Services*, etc.). A inspiração dos mecanismos de raciocínio.

- **Capítulo 3 – Modelagem do Conhecimento**

Os objetivos da modelagem. Os processos de teorização. As técnicas de modelagem do conhecimento. A imersão no domínio da malária. O modelo de conhecimento para o WST – Web Services Telediagnosis.

- **Capítulo 4 – O Sistema Especialista**

A definição de técnicas de documentação e ferramentas livres para construção dos protótipos. A modelagem e implementação do protótipo da aplicação de anamnese a partir de um sistema legado usando técnica estruturada. A modelagem e implementação do protótipo da aplicação do sistema especialista para telediagnóstico na web usando orientação a objeto. A integração dos dois sistemas a partir do sistema legado (não ainda no ambiente de serviços web).

- **Capítulo 5 – O SE no *Web Services***

A definição do framework para implementação do protótipo no ambiente de serviços da web. O processo de configuração do ambiente para disponibilização deste protótipo. A definição da padronização das interfaces e tipos de dados dos fornecedores das anamneses, assim como a construção classes das aplicações de serviço para troca de mensagens XML com o(s) provedor(es) do serviço de telediagnóstico. A organização dos processos acima no protocolo SOAP e a geração do script para comunicação via WSDL. Finalmente a disponibilização do serviço através da página até então integrada ao sistema legado descrito no capítulo 4.

- **Capítulo 6 – Conclusão**

Os resultados esperados e obtidos. A perspectivas de extensão para a fase 1. As sugestões para implementação para as fases 2 e 3 do projeto. As considerações finais.

Capítulo 2 Estado da Arte

Neste capítulo será visto:

- A telemedicina e o Telediagnóstico;
- A inteligência artificial;
- Os sistemas especialistas;
- O Java e o JESS;
- Web Services;
- O ciclo de vida de um KBS Knowledge Based System

2 ESTADO DA ARTE

O projeto *WST – Web Services Telediagnosis* objetiva uma proposta de telediagnóstico que se utiliza além de tecnologias de ponta como sistemas especialistas, Java, *Web Services* e *JESS Java Expert System Shell*, de filosofias que o tornam um desafio exeqüível. Por isso é importante que se comente toda esta fundamentação que irá situar o leitor nos próximos capítulos deste trabalho.

2.1 Telemedicina / Telediagnóstico

É importante que se tenha primeiro o entendimento da área de estudo do telediagnóstico – a telemedicina. Uma definição clara expressa o assunto da seguinte maneira:

> "Define-se como telemedicina a utilização de recursos de Informática e Telemática (redes de computadores conectados por meios de telecomunicação) para a transmissão remota de dados biomédicos e para o controle de equipamentos biomédicos à distância" [22].

A telemedicina como já foi comentado na introdução deste projeto, foi iniciada na década de 1960, através do projeto *Mercury* [22] que consistia na telemetria fisiológica de dados dos astronautas em órbita. Com os avanços e a conseqüente viabilização de algumas tecnologias digitais como as telecomunicações, as redes de longa distância e os baixos custos dos computadores tornaram este campo de pesquisa e aplicações uma realidade.

As aplicações da telemedicina são classificadas até hoje em cinco áreas fundamentais de pesquisa e desenvolvimento de soluções:

- telediagnóstico: envio remoto de dados de sinais e imagens médicas, dados laboratoriais para finalidades diagnósticas;
- telemonitoração - acompanhamento de pacientes à distância, monitorando de parâmetros vitais de cardíacos, gravidez de risco, epilépticos proporcionando serviços automáticos e semi-automáticos de vigilância e alarme;
- teleterapia - controle de equipamentos à distância, tais como hemodialisadores;

- teledidática - aplicação das redes telemáticas na implementação de cursos médicos à distância;
- telefonia social: aplicações dos modernos recursos de telefonia convencional à assistência dinâmica, telecomunicação para pessoas deficiente, como surdos, cegos e mudos, apoio à medicina preventiva, e suporte a pessoas idosas (tele-socorro).

As duas áreas (especialidades médicas) que mais tem recebido benefícios do telediagnóstico a partir da transmissão de dados são a neurologia e a cardiologia. Esta última, transmitindo dados provenientes de eletrocardiogramas, ritmo respiratório e pressão e fluxo sangüíneo e na neurologia dados como eletroencefalografia, potenciais evocados cerebrais.

Estas iniciativas tem amenizado o risco de morte em pacientes graves através do monitoramento à distância dando-lhes um atendimento domiciliar em regiões distantes e reduzindo o número de internações com vistas a um acompanhamento até então inusitado.

Um exemplo de sistemas telemédicos com resultados satisfatórios para a cardiologia pode-se mencionar o tele-EEG e o cardiotelefone.

O cardiotelefone já aprovado em países como a Itália, consiste de um transmissor digital de 12 canais por via telefônica normal (terrestre ou celular), que colhe e envia o ECG, em tempo real, para um centro especializado de análise. Este, por sua vez, dispõe também de um equipamento especializado, com computador, modem, vídeo e registrador de ECG. A transmissão telefônica dispõe de um canal de voz bidirecional, que permite ao centro orientar o diagnóstico e a conduta ao solicitante. O transmissor é portátil (3 kg), com bateria, acondicionado em uma maleta. Desta maneira, pode ser transportado e utilizado por médicos ocupacionais em empresas, em atendimento domiciliar ou de emergência, em ambulâncias (UTI móvel).

Muitas regiões do mundo despontam em projetos voltados ao atendimento à distância. A Europa em função da União Européia é a que mais se destaca em iniciativas neste campo.

A CEE[3] na década de 1990 além de colocar a telemedicina em seus programas estratégicos através do projeto AIM[4] [5] alocou em torno de US$ 15 milhões como verba inicial para estas iniciativas.

[3] CEE Comunidade Econômica Européia.
[4] AIM Advanced Informatics in Medicine

Monitoração domiciliar de gestantes e de cardíacos na França e Inglaterra, telesocorro para idosos na Itália e Suiça, teleconsulta entre redes de hospitais na Holanda, Alemanha, Suécia, Itália, telediálise na Itália, envio de sinais eletrocardiográficos e imagens radiológicas em vários países, são alguns exemplos.

Significativa é a experiência italiana em Telemedicina, iniciado nos anos 70, com experimentos de tele-ECG na Universidade de Roma, e de tele-EEG em Udine, o que posteriormente levou, em 1976, à concretização da primeira rede nacional experimental de transmissão de ECG envolvendo 52 hospitais; e, em 1983, ao projeto TELECOS, de teleconsulta entre hospitais de quatro regiões italianas.

Esses esforços determinaram a criação, em 1989, de um consórcio de cooperação envolvendo universidades, centros de pesquisa, usuários e cerca de 23 empresas nas áreas de Informática, telecomunicações e engenharia biomédica.

Entre os projetos iniciados pelo consórcio estão o INSIEME, que implementa uma cidade inteiramente interligada por sistemas de telesocorro difuso, e o TELEMISM [27], de estabelecimento de uma rede telemédica de emergência para pequenas ilhas italianas, que posteriormente levou a um interessante projeto realizado em conjunto com a Grécia, na implantação de um sistema de telemedicina integrado para as ilhas gregas do mar Egeu, que estão em comunicação com a Universidade de Atenas.

A partir de 1990, a telemedicia foi apropriada como programa prioritário do Ministério de Pesquisa Científica e Tecnológica da Itália, tendo recebido dotação de US$ 80 milhões.

A seguir na Figura 2.1, o diagrama de uma solução de telemedicina na Ásia mais precisamente no Japão onde em torno de 450 mil habitantes da cidade de Wakayama, em uma região montanhosa são atendidas através desta tecnologia. Tecnologia vista também como estratégica pelo governo japonês.

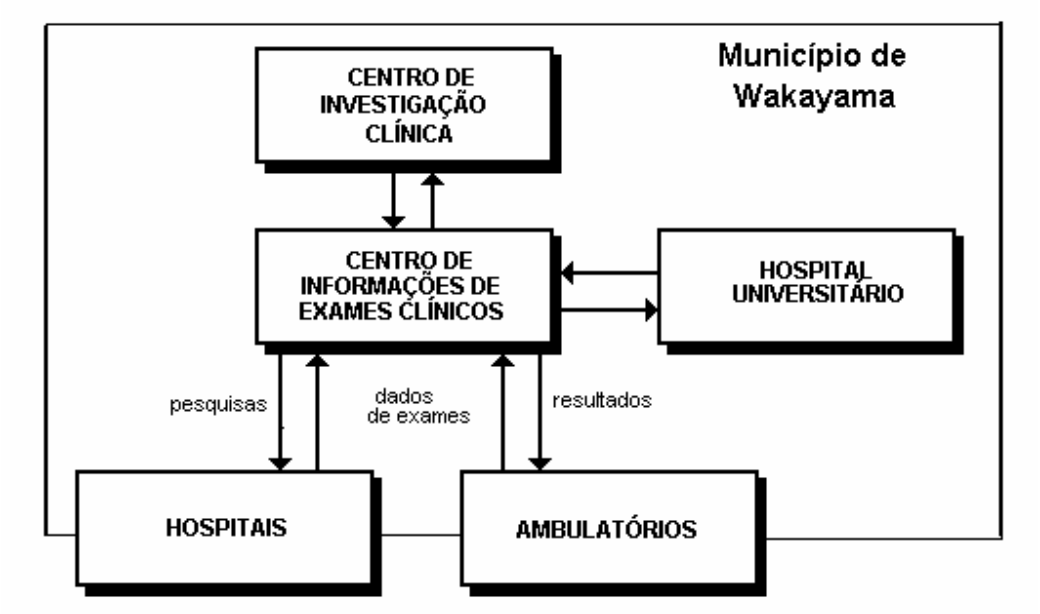


Figura 2.1 Solução de telemedicina na Ásia.

Pelo que se pode perceber, a hegemonia das redes de computadores, interligados por sistemas públicos e comerciais de telecomunicação, como Genie, MCI, CompuServe, America OnLine, Bitnet, Internet, está viabilizando um fenômeno influente de interconectividade das instituições e especialistas médicos em nível mundial.

Existem dados que, em 1993, cerca de 12 milhões de computadores e dezenas de milhões de usuários estariam ligados a sistemas desta natureza, que cada vez mais se interpenetram.

Diversos experimentos realizados nos EUA e Canadá, como o projeto SYNAPSE [22], que interliga hospitais rurais remotos a centros médicos avançados, através da rede Internet, têm mostrado a viabilidade de utilização dessa infra-estrutura na medicina.

De especial significância é o fato de que a rede Internet, a maior de todas, que inicialmente permitia apenas a conexão de usuários acadêmicos, está se transformando, nos EUA e em todo o mundo, em uma rede aberta a qualquer pessoa.

Já é comum nos EUA, por exemplo, o acesso dos próprios pacientes à rede, intercomunicando-se com seus médicos.

Investigações acerca da eficácia da telemedicina em diversos países demonstraram que a mesma é um recurso que contribui significantemente para a melhoria da qualidade da

assistência médica, para a redução do tempo gasto entre o diagnóstico e a terapia, e para a extensão dos serviços médicos especializados e de qualidade aos locais que não os apresentam.

Algumas aplicações da telemedicina na Itália, por exemplo, comprovaram uma redução de até 60 % nos custos da assistência, principalmente em decorrência da decentralização dos serviços, redução da necessidade de hospitalização, e diminuição com os gastos de deslocamento do paciente e de pessoal especializado. Além disso, a telemedicina permite implementar a assistência médica temporariamente em pontos remotos, em casos de catástrofes, eventos de grande afluência de público (eventos esportivos, artísticos e políticos, turismo, etc.).

A impressionante utilidade da telemedicina para o aumento da qualidade da assistência de saúde foi documentada durante o projeto TELECOS [22] de teleconsulta hospitalar, realizado entre 1987 e 1989 em quatro regiões da Itália.

Um levantamento em uma amostra de 1072 teleconsultas realizadas durante o período mostrou que, em 31 % dos casos, o centro solicitante declarou haver modificado a sua orientação diagnóstica ou terapêutica, em função da mesma, e que concordou com uma transferência do paciente para o centro consultado, em 21 % dos casos. Em outras palavras, sem o auxílio da teleconsulta, provavelmente esses pacientes teriam recebido assistência errada ou de menor qualidade.

No Brasil pode-se mencionar a Empresa Telemedicina da Bahia [31] que atende a pelo menos 50 municipios no Estado da Bahia.

Um outro exemplo de plena resposta em telediagnóstico e segunda opinião médica é o projeto HealthNet em Recife-PE com mostra a Figura 2.2.

O HealthNet é uma aplicação implementada na cidade de Recife a partir de uma rede metropolitana de computadores [3]. Esta aplicação terá o suporte de duas instituições: o Hospital Português (Beneficência Portuguesa de Pernambuco) e o Hospital das Clínicas.

Estes dois irão prover outros hospitais menores no Estado com serviços de telediagnóstico através do que é conhecido como “Rede Integrada de Cooperação em Saúde”.

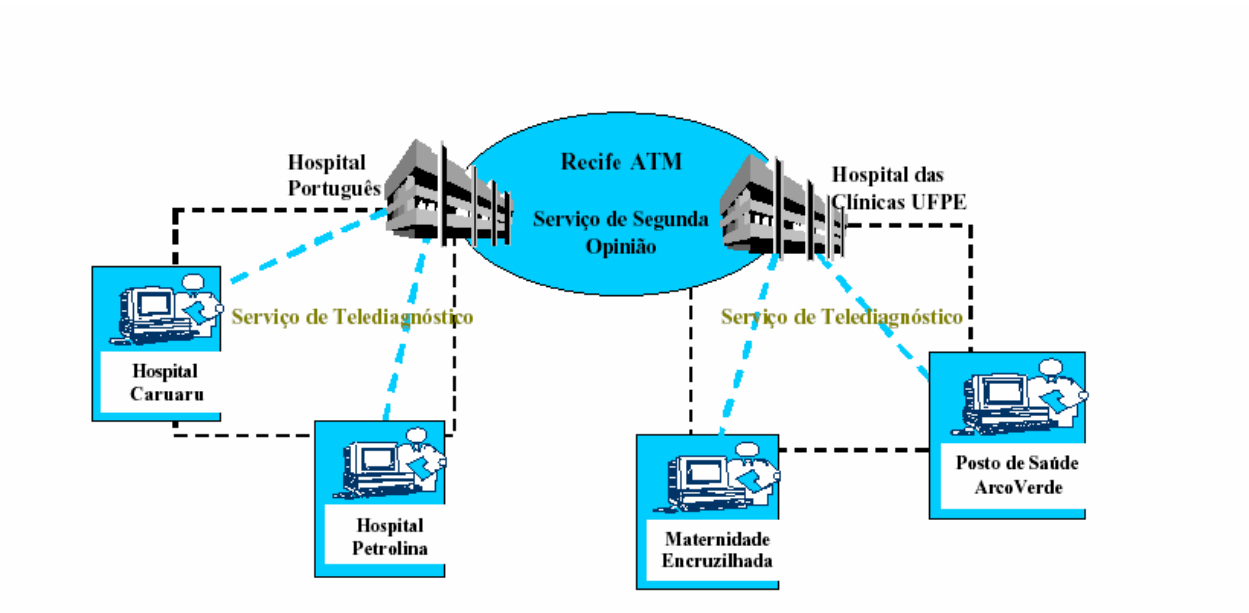


Figura 2.2 Ambiente onde irá funcionar o projeto Health Net [3]

O futuro da telemedicina parece ser promissor, visto que, os fatores são favoráveis para seu desenvolvimento. O desenvolvimento de novas soluções tecnológicas avançadas, como o da realidade virtual e da telepresença permitirá ações médicas complexas à distância.

Recentemente, por exemplo, um cirurgião italiano, localizado em Milão, operou um porco situado na Califórnia utilizando duas luvas de posicionamento digital ("datagloves'), que sentem e transmitem para o computador a posição espacial exata das mãos do cirurgião e a movimentação dos dedos; e um capacete de visão binocular, dotado de duas telas de computador que transmitiam uma visão tridimensional do campo operatório; o médico operou à distância mãos robóticas altamente precisas, para segurar instrumentos cirúrgicos.

Sistemas desse tipo foram desenvolvidos pela NASA, a agência espacial norte-americana, com a finalidade de prestar assistência médico-cirúrgica especializada aos astronautas a caminho da Lua e de Marte.

A telemedicina é uma tecnologia altamente inovadora, na qual quem se desloca é a informação e não o paciente. Sua gradativa implantação, que já está ocorrendo em diversos países, como no Canadá, Itália, EUA, Reino Unido, paises escandinavos, Japão, tem o potencial de promover uma grande revolução na maneira como a Medicina é praticada, bem como na própria estrutura do sistema de saúde desses países.

Seu desenvolvimento nos últimos anos tem sido explosivo, principalmente nos EUA, onde a recente preocupação com a redução dos custos do setor saúde e com a universalidade dos serviços, colocou as instituições médicas sob pressão, parte da qual pode ser resolvida a contento com a implantação da telemedicina.

Entre as tendências futuras, que certamente favorecerão um maior desenvolvimento da telemedicina, estão a desospitalização, o atendimento descentralizado, e o aumento da idade média da população, com seus conseqüentes impactos sobre os custos médicos e maior incidência de pacientes com doenças crônicas.

A experiência européia mostrou a necessidade de estabelecimento de programas integrados de apoio ao desenvolvimento da telemedicina, envolvendo governo e iniciativa privada. Tomando como exemplo o programa italiano de telemedicina que contempla o seguinte leque de áreas de desenvolvimento:

- Telemedicina hospitalar: sistemas inteligentes, integrados e distribuídos para processamento de prontuários;
- Imagens biomédicas: software para elaboração, integração e transmissão de imagens, para apoio à decisão médica;
- Assistência extra-hospitalar: sistemas de telemonitoração, telediálise, medicina perinatal, comunicação com deficientes;
- Desenvolvimento de padrões de hardware e software;
- Formação de recursos humanos, treinamento de pessoal técnico e aculturação da área médica.

Por ser, justamente, muito nova, a telemedicina exige um grande esforço e investimento nessa última área. Gerando uma experiência a ser repassada aos países que ainda não as tem, principalmente nas regiões em desenvolvimento.

Em um país como o Brasil, caracterizado por dimensões verdadeiramente continentais, e distribuição pouco uniforme de recursos de assistência médica, a telemedicina poderia ser de grande utilidade para proporcionar serviços remotos e móveis de medicina especializada às zonas menos dotadas.

Desta forma, com investimentos proporcionalmente baixos, a medicina de qualidade poderia ser estendida às regiões mais remotas do país, com pequeno retardo entre diagnóstico e conduta.

Um dos fatores que facilitaria a difusão da telemedicina no Brasil seria o bom estado de desenvolvimento tecnológico da informática brasileira, bem como um sistema extenso e funcional de telecomunicações, que dispusesse de modernos recursos de telefonia pública e celular, sistemas de transmissão de dados, ligação por satélite em todo o território nacional, redes Internet e Bitnet.

Se os bancos brasileiros já utilizam com grandes vantagens essa portentosa e eficiente infra-estrutura, porque não o pode fazer o setor de saúde ?

2.2 Inteligência Artificial

O grande anseio humano foi sempre compreender como se dá o ato de pensar, como se processa a capacidade de compreensão das coisas dentro de um contexto. “O campo da **inteligência artificial** tenta não apenas compreender, mas construir entidades inteligentes”[23]. A inteligência artificial é um dos campos de pesquisa mais recentes, apesar do desejo humano supra citado. As primeiras iniciativas sobre o assunto se deram com o pós-guerra, mais precisamente em 1956.

Inúmeras definições – obviamente todas congruentes – explicam os objetivos da IA, assim como, as suas aplicabilidades.

> “A arte de criar máquinas que executam funções e exigem inteligência quando executadas por pessoas”.
>
> (Kurzweil, 1990) [23]

> “O estudo das computações que tornam possível perceber, raciocinar e agir.”
>
> (Winston, 1992) [36]

O que se percebe em comum nas definições como estas acima e em muitas outras é que todo estudo sempre envolve processos de pensamento e raciocínio. No entanto apesar de congruentes, como em toda ciência, correntes de pensamento dividem o assunto.

Alguns abordam os aspectos humano-comportamentais que são permeados pelo cognitivismo em si, enquanto que outros exploram o racionalismo onde encontram explicação matemática para o pensamento - conexionismo.

Alan Turin (1950), bem mais objetivo, propôs um teste simples que consiste na confrontação homem e máquina sobre um determinado assunto. Caso não se consiga identificar de onde provém as repostas dentro de um censo de coerência esta provado a possibilidade existencial da IA.

Infelizmente não basta a Lei de Turing para provar que máquinas inteligentes são possíveis. Para que programas de computador de fatos “pensem” é necessário que se lance mão de maneira profunda de componentes reais da mente humana, tais como, a captura destes pensamentos à medida que eles se desenvolvem e através de experimentos sobre estes pensamentos.

Embora que ainda existam muitos mistério acerca da mente humana a IA tem evoluído rapidamente suportada por uma pluridisciplinaridade considerável. A principal prova desta assertiva é a própria história da inteligência artificial.

Nos dias de hoje a IA evolui progressivamente, atuando em campos como processos de aprendizagem e percepção, jogos eletrônicos, diagnóstico e monitoramento médico entre outros.

Como se percebe o maior objetivo da IA é automatizar as atividades intelectuais humanas, logo isso a torna a área de estudo mais próxima destes seres no que diz respeito à acuidade das soluções propostas por ela.

Para que a IA possa provar sua eficácia é necessário penetrar nos elementos reais da mente humana. Isso pode ser conseguido de duas formas: através da introspecção onde capturamos nossos próprios pensamentos a partir do seu processo de desenvolvimento ou por meio de uma corrente de estudo psicológico que dê explicação aos simbolismos do comportamento e suas associações.

Só então se pode afirmar, a existência de uma teoria suficientemente precisa para que seja possível expressar a inteligência em um programa de computador, quando os elementos de entrada / saída e sincronização deste referido programa coincidirem ao comportamento humano. Como diz (Newell e Simon) – criadores do GPS[5] - "não basta criar um mecanismo de resolução correta de problemas, é importante se acompanhar os passos de raciocínio para resolução do mesmo".

A ciência cognitiva reúne modelos computacionais da IA e técnicas psicológicas para calcar teorias precisas acerca dos processos da mente humana, mas é necessário que se compreenda que as duas correntes têm seus próprios elementos. O que é de fundamental importância para que se possa associá-las.

O pensamento humano foi investigado na antiguidade por um filosófo chamado Aristóteles. Ele na verdade tinha a pretensão de formular um modo de pensar considerado "correto" (i.e. "Sócrates é um homem; todos os homens são mortais; logo, Sócrates é mortal") Processo de inferência como estes deveriam na opinião do pensador dirigir as leis do pensamento humano – inclusive dando origem aos princípios da lógica.

Com base na linha de raciocínio acima, já no século XIX, lógicos descreveram notações irrefutáveis para declarações sobre todos os tipos de coisas do mundo e suas relações. O que deu origem logo em seguida às tradições logicistas[6] permitindo desta forma a

[5] GPS General Problem Solver
[6] Logicista = tradições baseadas em modelos de raciocínio lógico.

criação de sistemas computacionais inteligentes, apesar dos obstáculos que separam a teoria da prática.

2.3 Sistemas Especialistas

Há uma década atrás o campo de estudos acerca dos sistemas especialistas teve o seu mais significativo crescimento. Inúmeras implementações e publicações evidenciam este fato. Interessante é, observarmos algumas aplicações – ainda que em linhas gerais - para comprovarmos a essencialidade esta assertiva.

Um primeiro projeto é o SEAMED [33]. Uma das proposições é a modelagem através de redes bayesianas[7] de um problema de uma consulta simples. Construiu-se uma ferramenta que auxilia os médicos na indicação de diagnósticos em domínios específicos, através do conhecimento adquirido junto aos especialistas, e representado na rede probabilística.

Ainda na área de telemedicina, desta vez no segmento de pesquisa da teleaprendizagem encontramos um sistema inteligente para ensino da eletrocardiologia [15]. STI[8] como são conhecidos os artefatos de software que associam a Inteligência Artificial a fins didáticos.

Este baseado na filosofia de interação efetiva do aluno com o conteúdo a ser exposto, objetiva primordialmente modelar o conhecimento produzido por esta relação visando tornar mais eficaz o processo de transmissão de conhecimento do professor para o aluno.

> "As dificuldades encontradas pelos professores referem-se a exposição de alguns conceitos médicos como, por exemplo, demonstrar e analisar o funcionamento de um determinado órgão durante a execução de procedimentos para diagnósticos médicos". [15]

2.4 Java / JESS

O JESS – Java Expert System Shell, é um shell para construção de sistemas especialistas construído inteiramente na linguagem Java da Sun Microsystems [8]. O JESS tem a finalidade de desenvolver sistemas baseados em regras os quais podem ser fortemente acoplados a portabilidade da linguagem Java.

[7] Redes Bayesianas – ramo da Inteligência Artificial que trabalha com incertezas associadas.
[8] STI Sistemas Tutores Inteligentes.

O JESS 6.1 é compatível com todas as versões do Java a partir da 1.2. Embora os esforços do fabricante, ainda não é um software "perfeito", talvez nele ainda possam ser encontrados alguns *bugs* .

O JESS é uma biblioteca de programação escrita em Java para que possa ser referenciada a partir de uma classe. Esta biblioteca possui uma linguagem com recursos para construção de mecanismo de inferência. A linguagem JESS é semelhante à linguagem definida pelo CLIPS que é uma outra ferramenta com o mesmo escopo.

O autor Ernest J. Friedman sugere que os programadores que desejam usar o JESS tenham conhecimento de programação em Java e Lisp, visto que esta última é uma versão especializada do CLIPS.

O JESS, antes de ser usado deve ter seus arquivos de texto Java convertidos em *bytecode* – caso a sua distribuição seja a que contém os arquivos fonte da ferramenta.

O JESS tem uma linha de comando interativa que pode ser acessada escrevendo-se:

c:\>java jess.Main

no *promtp* de comando. Isso faz com que seja executado o modo interativo dos comandos da linguagem JESS. Uma outra forma de utilização do JESS é através da execução de um arquivo .clp (um arquivo com o texto da linguagem JESS). E finalmente a referência ao pacote JESS a partir de uma classe Java.

O JESS possui uma classe chamada jess.ConsoleApplet que funciona como um *display*. É possível ser usado em situações de perguntas e respostas combinados com classes applets no ambiente da *web*.

Existe uma discussão acirrada em torno da JVM[9], em face da versão usada pela maioria dos browsers. A API do Java 2 não trabalha na JVM nativa. A recomendação é que se baixe o *plug-in* para esta versão do site da Sun www.sun.com ou se use as versões 4 ou 5 do JESS.

É preciso levar em consideração que mesmo nas versões 4 e 5 a classe ConsoleApplet e ConsoleDisplay usam o modelo de evento do Java 1.1 apesar de não suportado por algumas das bases instaladas dos *web* browsers – nestes casos o *plug-in* talvez será necessário.

O fabricante do JESS recomenda que não use o ConsoleApplet caso o desenvolvedor queira *applets* altamente portáveis, a idéia alternativa é que execute o JESS no

[9] JVM Java Virtual Machine – mecanismo que garante a portabilidade da linguagem Java.

lado servidor como *servlet* e execute uma GUI no cliente (como será implementado no projeto *WST Web Services Telediagnosis*).

Um programa de base de regras pode processar milhares de regras, e o JESS continuamente aplicará os dados às bases de conhecimento. Freqüentemente as regras são representadas a partir do conhecimento heurístico, ou seja, a partir de um conjunto de normas de *expert* humano sobre algum domínio e a base de conhecimento representa um estado envolvendo uma situação, então isto é considerado um **sistema especialista**.

Sistemas especialistas são largamente usados nos mais variados domínios. Dentre as aplicações recentes, os sistemas especialistas estão como raciocinadores nos agentes inteligentes nos sistemas ERPs e nas validações nos sistemas de comercio eletrônico.

A concepção do JESS é na realidade ser linguagem de programação de mecanismos de inferência para os mais variados propósitos, além do mais ele pode ser referenciado por todas as classes e bibliotecas do Java. Por esta razão o JESS é freqüentemente usado como script dinâmico ou em ambiente de desenvolvimento de aplicações rápidas.

JESS é diferente então de alguns sistemas baseados em rede, nisso inclui ao mesmo tempo uma espécie de *backwards chaining* e uma construção chamada *defquery* nas quais se faz consultas diretas às bases de conhecimento. Ambos ajudam o JESS a uma melhor adequação para algumas aplicações.

O algoritmo de rede é todo sobre computação de objetos, assim este nunca precisa ser recomputado e então reutilizado.

O JESS oferece um rico conjunto de possibilidades. Uma em que a mortalidade é codificada diretamente nos fatos, assim nunca é necessário ser toda computada poderá ser vista no apêndice A.

Os fatos usados no apêndice A são expressos para aqueles humanos que são mortais, e um para cada humano conhecido. Neste exemplo fatos extras não são gerados. Todavia a mortalidade de Sócrates é relembrada e talvez usada para otimizar a computação mais tarde.

Pelo fato do JESS ser uma aplicação de consumo memória intensiva, esta performance é sensível para o comportamento do coletor do Java *Garbage*. As características das JVM mais recentes incluem uma chamada *HotSpot* nas quais incluem a flexibilidade para a configuração da *garbage collection subsystems*.

2.4.1 A Linguagem JESS

A notação usada é extremamente informal. Strings entre <> são tipos de dados que devem ser fornecidos, objetos entre [] são opcionais, objetos terminados com + podem aparecer uma ou mais vezes e objetos terminados com * podem aparecer nenhuma ou mais vezes.

O átomo ou o símbolo é o centro da concepção da linguagem JESS. Os átomos são muito parecidos com os identificadores na outras linguagens. Um átomo JESS pode conter letras, números e as seguintes pontuações: $*=+/<>_?#. Átomos JESS são case sensitive.

O JESS usa as funções Java *java.lang.Integer.parseInt* e *java.lang.Double.parseDouble* para converter inteiros em floats respectivamente.

Caracter string no JESS são denotadas usando aspas dupla (“). Contrabarras podem ser usadas para substituir as aspas. É importante que se saiba que as strings do JESS são diferentes das strings do Java. Primeiro na seqüência de escape elas não são reconhecidas. Você não pode encaixar novas linhas usando \n por exemplo. Por outro lado novas linhas são permitidas dentro de aspas duplas - estas se tornam parte da string.

Uma outra unidade fundamental para a sintaxe do JESS são as listas. Uma lista consiste de átomos, números, strings ou outras listas dentro de parênteses. O primeiro elemento de uma lista (o *car* na lista do LISP) é freqüentemente chamado de *head* no JESS.

Usa-se [;] para comentários no JESS e estes podem aparecer em qualquer lugar do programa.

Todo código no JESS (estruturas de controle, chamadas de procedimentos) toma a forma de uma chamada de função. Chamadas de função no JESS são simplesmente listas. Estas usam uma notação prefixada. O átomo da cabeça da lista é o nome da função de chamada. Por exemplo, uma função que usa + para adicionar o número 2 e 3 deve ser escrita (+ 2 3). Quando avaliada, o valor desta expressão é 5 (não é uma lista contendo simplesmente 5 elementos). Em geral, expressões são reconhecidas como tal, e avaliadas num contexto apropriado.

Uma outra característica é que se pode aninhar chamadas de funções, ou seja, as funções mais externas são responsáveis pela avaliação das chamadas de funções mais internas. JESS vem com um largo número de funções pré-construídas.

Uma das funções mais comumente usadas é o *printout*. Este é usado para enviar texto do JESS para uma saída padrão ou para um arquivo.

Outra função muito usada é o *batch*. Este avalia o código de um arquivo JESS.

Variáveis de programação no JESS são átomos com o caracter de interrogação. O caracter de interrogação é parte do nome da variável. Uma variável normal pode se referir a um átomo simples, um número ou uma string. Uma variável em que o primeiro caracter é um $ (i.e. $?X) é multivalorada a qual pode fazer referência a um tipo especial de lista chama *multifield* . Você declara variáveis usando a função *bind*.

Multifields são criados geralmente usando-se funções *multfields* especiais como *create$* e pode então ser restringidas a multivariáveis.

Variáveis não precisam ser declaradas antes do seu uso, exceto um tipo especial chamado *defglobals*. Para se verificar uma variável no *prompt* do JESS é necessário simplesmente escrever o nome da variável.

Variáveis criadas a partir do *prompt* do JESS ou no nível mais alto no programa utilizando a linguagem JESS, são a qualquer momento “limpas” com o comando *reset*.

Ao criar variáveis globais estas não são destruídas pelo *reset*. Você pode usar o construtor *defglobal*. Nomes de variáveis globais devem ser iniciadas e terminadas por *.

O JESS usa para declaração de funções o construtor *deffunction*. O controle de fluxo dentro de uma função é executado através do *foreach*, *if* e *while* .

As funções podem ser invocadas no JESS a partir de vários pontos da aplicação.

O JESS converte os valores do Java para os tipos do JESS de acordo com a Tabela 2.1:

Java type	Jess type
A null reference	The atom 'nil'
A void return value	The atom 'nil'
String	RU.STRING
An array	A Jess multifield
Boolean or java.lang.Boolean	The atoms 'TRUE' and 'FALSE'
byte, short, int, or their wrappers	RU.INTEGER
long or Long	RU.LONG
double, float or their wrappers	RU.FLOAT
char or java.lang.Character	RU.ATOM
anything else	RU.EXTERNAL_ADDRESS

Tabela 2.1 Conversão de tipos de dados Java / JESS

JESS converte valores do JESS para os tipos do Java com muita flexibilidade de acordo com a Tabela 2.2. Geralmente quando convertendo nesta direção faz uma associação mais ou menos aproximada quanto aos tipos envolvidos.

Jess type	Possible Java types
RU.EXTERNAL_ADDRESS	The wrapped object
The atom 'nil'	A null reference
The atoms 'TRUE' or 'FALSE'	java.lang.Boolean or boolean
RU.ATOM, RU.STRING	String, char, java.lang.Character
RU.FLOAT	float, double, and their wrappers
RU.INTEGER	long, short, int, byte, char, and their wrappers
RU.LONG	long, short, int, byte, char, and their wrappers
RU.LIST	A Java array

Tabela 2.2 Conversão de tipos de dados JESS / Java

Um Sistema baseado em regras é uma coleção de sentença chamadas fatos. Esta coleção é conhecida como base de conhecimento. Isto é de certa forma uma base de dados relacional. Especialmente porque os fatos devem ter uma estrutura especifica. No JESS estas são as três espécies de fatos: *ordered facts, unordered facts e definstance facts.*

Ordered facts são listas simples, quando o primeiro campo (a cabeça da lista) age como classificador da categoria por fato.

Ordered facts são mais usados porque são desestruturados. Em alguns momentos você vai precisar de uma organização maior. Nas linguagens orientadas a objeto, objetos tem nomes e campos que contém dados. Unordered facts oferecem esta capacidade (embora os campos sejam chamados tradicionalmente de slots).

Uma recomendação é antes de criar os unordered facts, se deve definir os slots que serão usados, através do *deftemplate construct*,.

2.5 Web Services

O ambiente da internet tem servido de canal de aproximação entre os mais variados seguimentos usuários de computadores no mundo. Agora recentemente, uma por assim dizer, nova descoberta acerca desta tecnologia - os serviços *web*, tem tornado mais próximo estes usuários.

Serviços *web* são funcionalidades de aplicações, até então não públicas agora disponibilizadas na *web* [9]. Tais funcionalidades vão desde a verificação de cartões de

crédito passando por marcação de consultas em um consultório médico até transações de e-business envolvendo diversas instituições remotas.

2.5.1 Arquitetura dos Web Services

A arquitetura na verdade é o conjunto de componentes que auxiliam na construção de um serviço *web*. Já foi dito que o objetivo maior de um serviço *web* é tornar funcionalidades públicas, ou seja, viabilizar a interoperacionalização entre plataformas diversas.

Com este objetivo é importante que se entenda algumas premissas do funcionamento de um serviço *web* na prática. O diagrama da Figura 2.3 que demonstra em linhas gerais como um serviço *web* é publicado e disponibilizado:

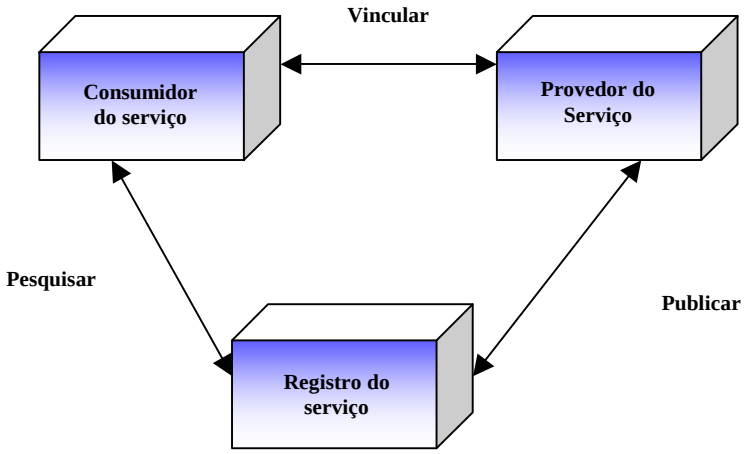


Figura 2.3 Modelo de serviço *web* [9]

Dois elementos conhecidos como operações e papeis fundamentam o funcionamento de um serviço *web*. Papeis são os diversos tipos de entidades envolvidas no processo de provimento do serviço e operações são as funções executadas pelas entidades.

Partindo do pressuposto que entidades e operações formam uma associação indispensável observe a análise do modelo básico de um serviço *web*.

O provedor de serviço como o nome sugere, é o criador do serviço, ou seja, uma empresa que possui uma funcionalidade que precisa ser difundida através do ambiente da internet via um serviço *web*.

O consumidor é o elemento cliente. O elemento que faz uso da funcionalidade disponibilizada pelo provedor. De que maneira? Pesquisando o registro do serviço publicado pelo provedor.

Finalmente o registro do serviço. Esse é uma espécie de central de registros, onde o cliente ou consumidor faz a busca pela funcionalidade que deseja utilizar.

O que associa de maneira ordenada estes três elementos básicos, são também três operações básicas: a localização o vinculo e a publicação do serviço.

2.5.2 Pilha de serviço web

Para que estas operações possam alcançar o objetivo da interoperabilidade é necessário, sobretudo que haja uma transparência no que tange a plataforma operacional, linguagem, etc. Para que isso ocorra é necessário um padrão para cada uma das operações citadas acima, o que levou ao desenho de uma pilha básica de serviços – Figura 2.4 - com as seguintes tecnologias:

- HTTP Hiper Text Transfer Protocol é o protocolo de transporte mais usado na *Web*;
- SOAP Simple Object Access Protocol é um protocolo de passagem de mensagem puramente neutro através de XML;
- WSDL *Web Services* Description Language é a linguagem que descreve as funcionalidades a serem interoperacionalizadas;

UDDI Universal Description, Discover and Integration é o “elo” de ligação entre publicação de serviços e busca dos serviços pelo cliente.

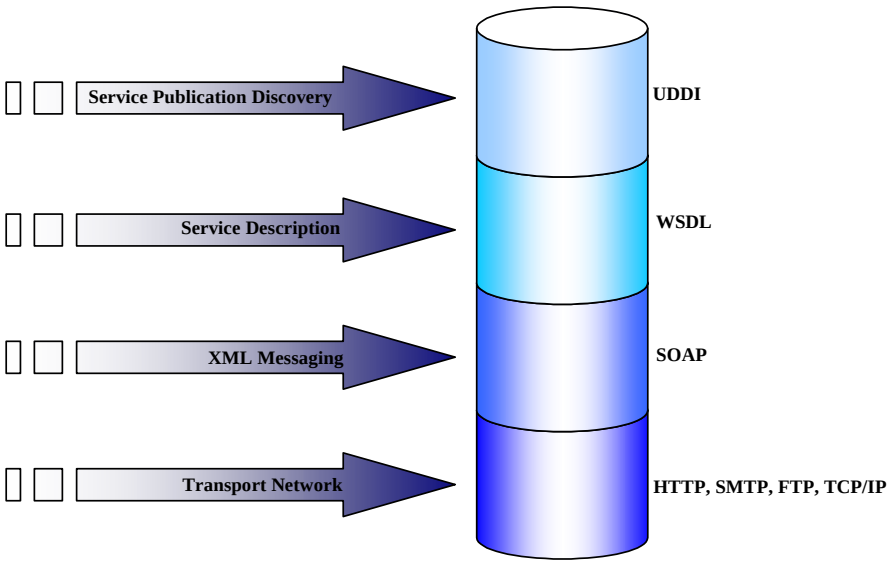


Figura 2.4 Pilha de services *web*

2.5.3 Arquitetura dos serviços no J2EE

Observa-se do ponto de vista técnico alguns detalhes sobre elementos que compõem um serviço *web* em uma corporação.

As funcionalidades comerciais devem primordialmente dispor de chamadas que possam ser acessadas. Esse é o grande desafio. Muitas aplicações, além de não possibilitarem este acesso seguem todo um fluxo dentro da corporação até o resultado da consulta do usuário, demandando desta forma tempo.

O sistema de serviço *web* tem um conceito semelhante ao *container* do J2EE[10], ou seja, mapeia as solicitações SOAP[11] para os componentes adequados das funcionalidades comerciais.

O servidor e o cliente são o centro de todo o processo. Ele (o servidor) é o responsável pelo tratamento das solicitações SOAP do cliente, além do que os arquivos WSDL[12] estão localizados nele. Cabendo ao cliente acessar via a referencia UDDI[13] a URL para acesso ao WSDL.

Com o objetivo de classificar, os grupos de serviços *web* foram divididos em dois: serviços *web* simples e serviços *web* de negócio. Os primeiros mais comuns se baseiam no padrão solicitação-resposta, já os serviços *web* de negócio dependem de uma estrutura bem mais arrojada, que envolve a colaboração de vários negociantes.

Antes que se fale em serviço *web* em uma empresa se deve avaliar a possibilidade de definir que uma funcionalidade possa ser ou não exposta como serviço público.

A seguir o detalhamento de um serviço web simples, através do diagrama de aplicação demonstrado na Figura 2.5.

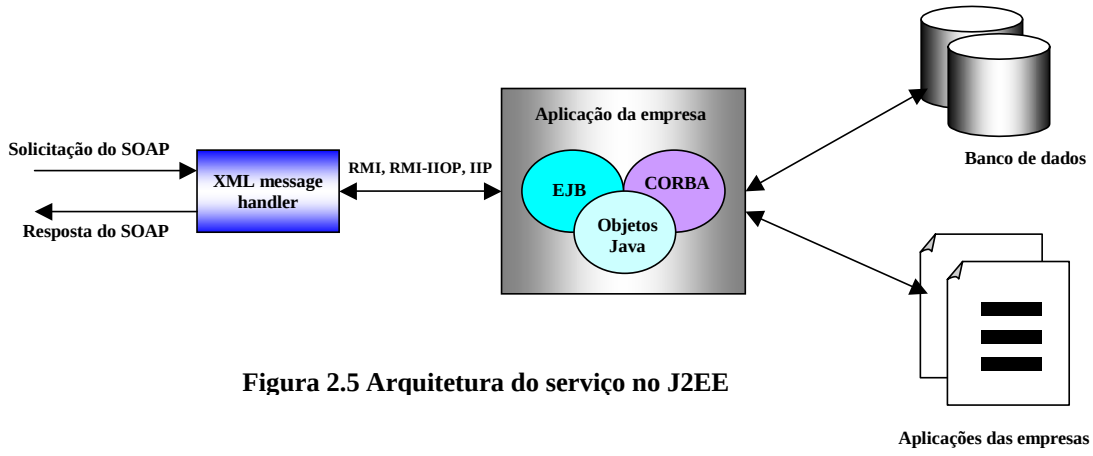


Figura 2.5 Arquitetura do serviço no J2EE

[10] J2EE Java 2 Enterprise Edition
[11] SOAP Simple Object Access Protocol
[12] WSDL Web Services Description Language
[13] UDDI Universal Description, Discovery and Integration

Baseado no diagrama acima que fornece uma visão do nível da aplicação, se pode definir alguns componentes centrais indispensáveis em um serviço *web*, são eles:

- Mecanismo de análise do XML que valida a especificação XML em relação ao SOAP;
- Mecanismo tradutor do XML que traduz o dispositivo de solicitação do cliente de acordo com a resposta que este pretende obter;
- Segurança validação sobretudo das credenciais do usuário;
- Mecanismo de logging e registro de ocorrências durante as transações de chamada-resposta.

Dessa forma agora fica o diagrama do nível de aplicação para um serviço web simples – Figura 2.6:

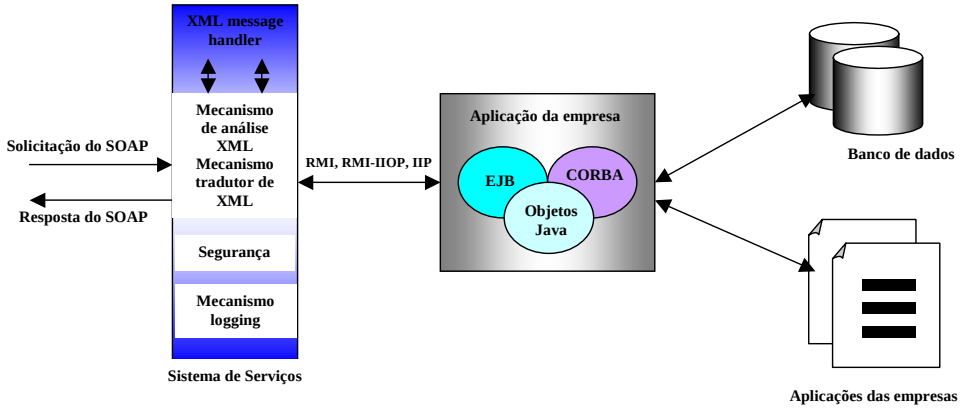


Figura 2.6 Diagrama de nível de aplicação para serviço *web*

2.5.4 O SOAP

O SOAP – Simple Object Access Protocol, é um protocolo de comunicação com peculiaridades interessantes como a superficialidade a sua capacidade de trabalhar em ambientes de sistemas distribuídos [12], além do que o SOAP utiliza a linguagem XML[14] para troca de mensagens fornecendo desta maneira somente o *framework* para envio e resposta, ao contrário dos outros protocolos do mesmo escopo onde o formato é binário.

Este protocolo também pode usar a camada de transporte de protocolos como HTTP[15] viabilizando a questão de custo de infra-estrutura de servidores e ambiente *web*.

Essas vantagens acima citadas se traduzem em um protocolo independente de plataforma de hardware ou sistema operacional e tem padrões totalmente abertos.

[14] XML eXtensible Markup Language.
[15] http Hyper Text Tranfer Protocol.

Um comentário a fazer, é sobre a superficialidade no que diz respeito ao nível de complexidade do SOAP em relação a outros protocolos. O SOAP é bem mais simples, um exemplo disso é que o CORBA precisa de um ORB para se comunicar e o SOAP não, este também não descreve nem localiza objetos remotos nele mesmo – isso é trabalho para a WSDL e para o UDDI.

O SOAP basicamente se especifica em duas partes: o encapsulamento da mensagem e a chamada desta através de RPC[16].

É de acordo com o diagrama da Figura 2.7 que o SOAP se comunica com outros protocolos:

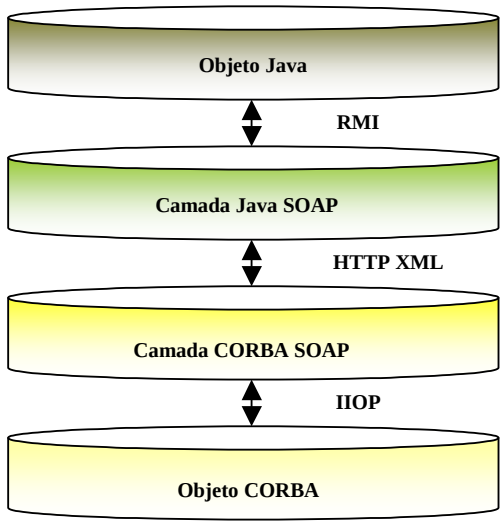


Figura 2.7 Comunicação do SOAP com outros protocolos

A idéia do SOAP em 1998 foi juntar esforços de empresas para definir a especificação de um protocolo que transmitisse documentos XML com comandos para chamadas em sistemas remotos RPC.

Para isso era necessária uma linguagem fortemente tipada. Criaram alguns tipos no XML como arrays e estruturas, além de alguns tipos primitivos.

Vantagens e desvantagens do SOAP.

Vantagens:

- Atravessa firewalls com facilidade
- Dados estruturados usando XML
- Multi-protocolo de transporte http, smtp, etc.

[16] RPC – Remote Procedure Call.

Desvantagens:

- Mecanismo de segurança imaturo
- Não existe garantia de entrega da mensagem
- Não existe publicação nem assinatura

2.5.5 Arquitetura SOAP no Java

Várias arquiteturas podem inclusive ser combinadas em uma aplicação. Na Figura 2.8 o diagrama padrão do SOAP e alguns diagramas que demonstram a sua flexibilidade no Apêndice B:

Diagrama Padrão.

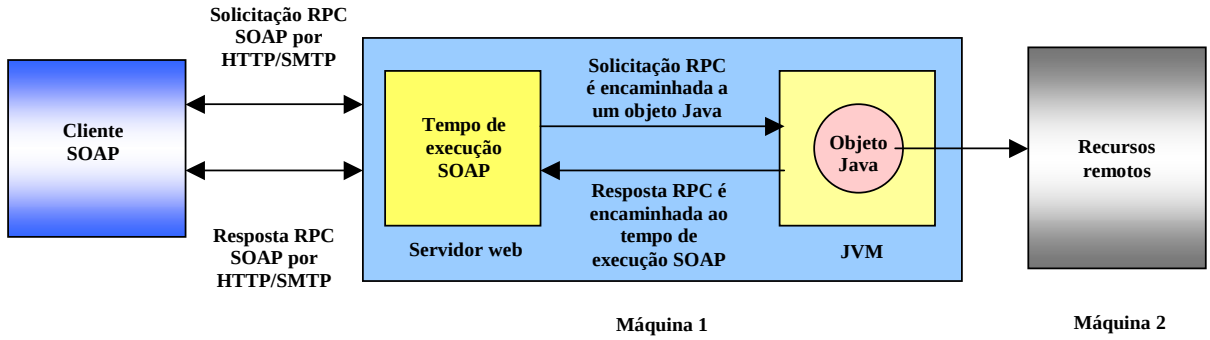


Figura 2.8 Arquitetura SOAP no Java

Nesta arquitetura tem um cliente SOAP e um recurso remoto a ser acessado. Uma mensagem remota é enviada através de um protocolo de transporte HTTP ou SMTP. Ao receber a solicitação o ambiente do SOAP se encarrega de rotear a solicitação ao objeto java que contém o método desejado.

2.5.6 XML

A linguagem XML – Extend Markup Language é usada pela sua simplicidade para codificar as mensagens no SOAP. A mensagem SOAP inclui todos os *namespaces* da XML para definição dos elementos e atributos caso contrário a mensagem talvez não seja processada.

A grande vantagem de usar os namespaces do SOAP é que os elementos definidos pelo usuário não se conflitarão com o padrão dele – SOAP. Na Figura 2.9 o formato básico de uma mensagem SOAP.

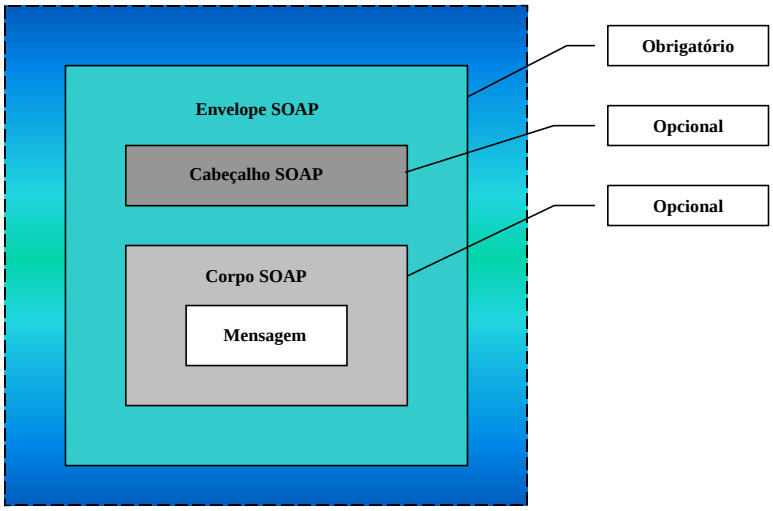


Figura 2.9 Formato de mensagem SOAP

No Apêndice C estão disponíveis os detalhes de cada elemento que compõe esta mensagem.

2.5.7 SOAP-http

Veremos agora o vínculo SOAP / HTTP. Apesar de saber que o SOAP na prática se vincula a qualquer protocolo de transporte, visto que o HTTP é o mais usado veremos como isso acontece.

Uma solictação HTTP do SOAP é uma solicitação HTTP básica só que contendo uma solicitação SOAP, veja a codificação XML no Apêndice D.

2.5.8 SOAP-rpc

A principal vantagem do SOAP é permitir que chamadas possam ser encapsuladas dentro da carga útil do SOAP, isso significa que chamadas e respostas RPCs são mapeadas naturalmente para o HTTP.

Para criar um método usando o HTTP, são necessárias as informações:

- A URI do objeto de destino;
- O nome do método;
- Os parâmentros do método.

2.5.9 Linguagem de descrição WSDL

Nesta seção será descrita a linguagem que provê a integração através de uma interface, permitindo que os serviços disponíveis na *web* possam ser interoperáveis.

A *WSDL Web Services Description Language* [6] é uma linguagem baseada no padrão de especificação XML. Através de um esquema de XML consegue-se descrever a estrutura de um documento WSDL. Para que se entenda a WSDL é necessário que se conheça cada elemento desse esquema e as suas funcionalidades.

2.5.10 Estrutura do documento WSDL

A WSDL é uma iniciativa da IBM, Microsoft e Ariba e foi submetido a W3C que é a sociedade responsável pela padronização de vários formatos entre eles o *XML Extensible Markup Language*.

Um documento WSDL, na realidade é o componente externo, que através de uma interface, representa o acesso a um serviço *web*. A WSDL esta para o ambiente de serviço *web* como o *IDL Interface Definition Language* esta para o padrão CORBA. A WSDL além de descrever a interface do documento ela indispensavelmente informa a localização deste serviço, apesar do mesmo estar disponível em um local reconhecido, permitindo um acesso confiável.

Para que se perceba o mecanismo da WSDL é necessário que se conheça e se entenda como é estruturado um serviço *web*. Nele estão disponíveis de maneira comum os seguintes componentes:

- Tipos de dados (strings, int, object, etc.);
- Parâmetros: na verdade basicamente o conteúdo da mensagem;
- Operações: assinatura dos métodos;
- Tipo de porta: na verdade significa um agrupamento lógico de métodos;
- Protocolo a ser usado para acessar os métodos;
- Endereço do serviço a ser acessado.

Observa-se como são mapeadas as informações dos serviços em um documento WSDL:

- O documento WSDL tem um elemento “maior” chamado <definitions>. Dentro do <definitions>;
  - <types> define os tipos de dados;
  - <message> define as mensagens usadas pelo sistema;

- <operations> define as operações (mensagens de solicitação resposta);
- <portType> encapsula a coleção de operações (o referido agrupamento de operações lógicas citados acima);
- <binding> descreve como a porta é mapeada para o protocolo da rede;
- <service> e o elemento <port> do <service> inclui a localização da implementação.

2.5.11 WSDL e Java

O desenvolvimento de uma interface usando a WSDL, depende do contexto do usuário. Este pode ser um provedor de serviços, um solicitante (cliente) ou ambos.

Existem na verdade duas abordagens para o problema: a) top-down (o usuário já tem a definição WSDL, o que consiste em somente fazer "adaptação" dos tipos de dados para a clesse Java existente); b) bottom-up (Esta é exatamente o contrário. A definição será construída com base no que esta definida como tipo de dados na classe existente).

2.5.12 WSDL para Java

Agora que já se sabe como funciona e a aplicabilidade da WSDL, observa-se como ela interage na linguagem Java, ou seja, como se dá o processo de criação e interpretação de um documento WSDL a partir de um programa Java.

Existe uma API (WSDL4J) que permite acesso total a documentos WSDL através do Java. Esta API esta contida em um pacote chamado *javax.wsdl*. A utilização desta API se dá pela referência ao pacote – previamente baixado e instalado adequadamente - a partir do código Java e também pelo uso do *framework* (o toolkit que lhe provê o ambiente de desenvolvimento do serviço *web*).

Somente no caso do usuário ser um provedor de serviço, ele necessitara gerar documentos desta natureza. Documentos WSDL podem ser criados manualmente usando-se um editor de texto ou um editor XML, mas existem ferramentas que geram a interface do serviço a partir da classe Java.

O papel dos clientes dos serviços *web*, inicialmente, é identificar que serviços lhe interessam nos documentos WSDL. Esta tarefa invariavelmente é provida também pelo *framework* onde o serviço foi desenvolvido. Este é responsável por gerar algum tipo de código que permita o usuário integrar sua aplicação ao serviço.

A classe *javax.wsdl.Definition* fornece os principais métodos da API WSDL4J.

Como foi comentado acima, esta API necessita para o seu funcionamento de alguns componentes do ambiente (framework) – outras APIs, no caso o JAXP. A JAXP provê um método *DocumentBuilderFactory,* que cria um objeto *DocumentBuilder*.

Este referido objeto cria uma instância que implementa a interface *org.w3c.dom.Document*. Esta interface possui métodos para criação de tipos de todos os outros elementos usados na XML DOM (i.e. *Node, Element, etc.*)

Enfim a classe *javax.wsdl.Definition* fornece métodos *createXXX( )* (i.e *createBinding( ), createMessage( ), etc.*). No Apêndice E será visto como se instancia a classe *javax.wsdl.Definition*.

2.5.13 Chamada dinâmica de serviço

Ao contrário do que foi abordado acima o acesso dinâmico não gera código direcionado a um serviço em especial – na verdade é uma espécie de acesso "genérico". Cria um cliente que pode acessar qualquer serviço a partir de um WSDL existente.

Novamente dependendo do *framework* que será desenvolvido o serviço *web*, o usuário terá disponível o pacote adequado ao acesso dinâmico a este serviço. No nosso caso, usaremos o WSIF[17] da IBM. Este é baseado na API WSDL4J e em face disto fornece as classes necessárias à análise dos documentos WSDL. No Apêndice F encontra-se descrito a forma como este *framework* funciona.

2.5.14 GLUE

Visto que, viu-se todo o processo de criação de interfaces, integração de aplicações e acesso dinâmico através da WSDL, resta agora implementar todas as etapas acima. Para isso se faz necessário um ambiente. Existem inúmeros. Um deles é o GLUE.

Este ambiente suporta todas as tecnologias de serviço *web*: WSDL, UDDI, SOAP, etc. Vem com um servidor *web*, mecanismo SOAP, servidor e cliente UDDI e console para administração de serviços. Totalmente implementada em Java.

2.6 Ciclo de vida de um KBS – Knowledge Base System (Sistemas Baseados em Conhecimento)

Serão abordadas agora as etapas para a elaboração de um KBS. Inicialmente a ferramenta que irá nortear todo o processo de desenvolvimento é o diagrama geral de ciclo de vida de um sistema baseado em conhecimento.

[17] WSIF Web Service Invocation Framework.

Este diagrama ilustra cada elemento envolvido na modelagem e cada elemento deste é refinado através de ferramentas adequadas (i.e. Ontologias [21] para modelagem do conhecimento, brainstorm para a coleta, etc.). Na Figura 2.10 será possível se acompanhar cada uma das etapas deste ciclo de vida e em seguida o seu detalhamento.

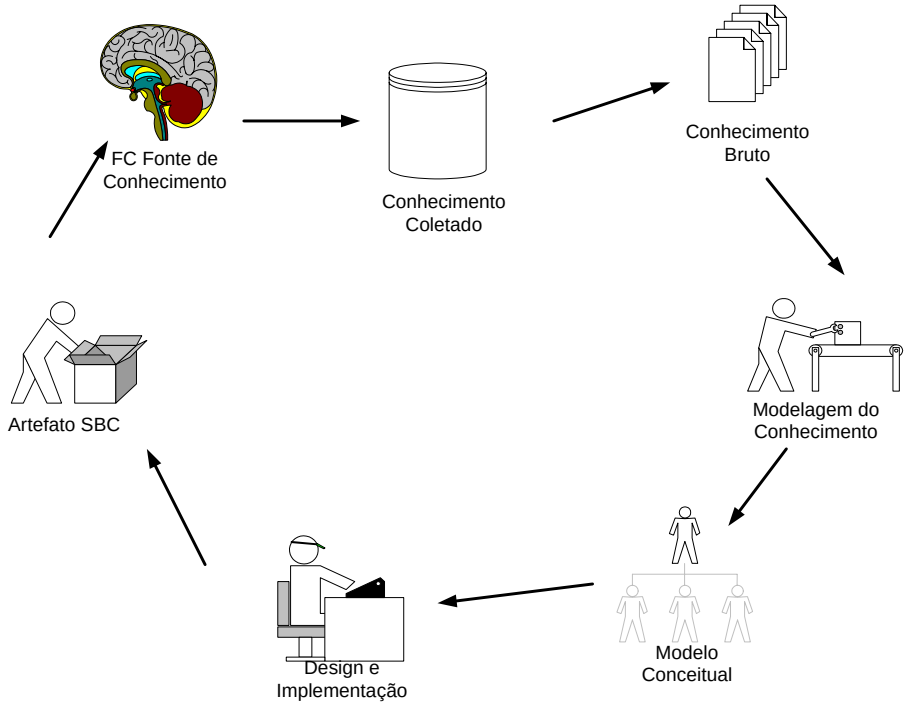


Figura 2.10 Ciclo de vida de um KBS

2.6.1 Coleta de conhecimento (CC)

A fonte de conhecimento é via de regra a figura do *expertise* ou especialista. A partir desta fonte um profissional denominado EC[18] (Engenheiro de Conhecimento) procede todo o apanhado de informações que permita formular uma semântica acerca do problema que naturalmente dará origem ao conhecimento.

A propósito vale a pena comentar que o referido EC deve ter algumas características que certamente facilitarão a elaboração da empatia com o especialista. Algumas delas são: sensibilidade, conhecimento – ainda que empírico - sobre o que esta coletando, capacidade de concentração, interpretação e assimilação de informação. Existem alguns desafios a serem vencidos no processo de coleta de conhecimento, dois deles são os que irão determinar o sucesso ou fracasso do modelo: 1) o especialista pode ser comparado a um “acervo humano” do domínio a ser extraído o conhecimento, em face disso

[18] EC Engenheiro de Conhecimento (profissional responsável pela coleta e modelagem do conhecimento)

ele precisa também conhecer minuciosamente os objetivos do compartilhamento das informações que ele detém e para que fins elas irão servir.

> "One of the best ways in which to gain the respect of the domain expert is to make na all out attempt – well before the initial meeting – to understand the problem, the environment in which it exist, and the specific terminology and jargon employed by the expert" [11];

2) o EC precisa ser, sobretudo **ético**.

Rigorosamente respeitada as premissas citadas acima, o primeiro passo na fase de aquisição de conhecimento é exatamente a identificação do(s) especialista(s) e em seguida do(s) seu(s) domínio(s) de conhecimento[11].

As ferramentas usadas no processo de coleta das informações têm características particulares no que diz respeito à resposta a ser interpretada pelo EC. Isso significa que cada técnica usada retorna uma resposta que poderá ser confrontada e usada para dirimir dúvidas. Entrevistas, questionários, *brainstorms*. darão suporte a metodologias de especificação como ontologias.

Em linhas gerais serão vistos alguns passos a serem seguidos no processo de aquisição de conhecimento e suas recomendações.

Como já foi comentada acima a primeira iniciativa é identificar o domínio, ou seja, a área de problema que esteja associada à abordagem da solução.

Tomadas de decisão dentro do domínio tem importância fundamental que refletirão no momento do suporte e da implementação deste domínio.

O gerenciamento de custos e riscos de um SE[19] é fato e deve ser levado em consideração, ou seja, uma engenharia de conhecimento utópica poderá ser desastrosa.

Um domínio deve ter uma certa estabilidade. As mudanças significativas devem ser previstas.

Quanto à seleção do EC, dois engenheiros no mínimo devem ser usados [11] e preferencialmente estes devem ser profissionais experimentados no desenvolvimento e implementação de sistemas especialistas; O EC deve ter uma visão holística e heurística acerca do domínio e primordialmente ter a habilidade de interpretar informações que formularão conhecimento e darão suporte à modelagem.

[19] SE Sistema Especialista

A identificação do especialista, via de regra fica a cargo da organização para a qual esta sendo desenvolvido o sistema.

Essas são as recomendações essenciais para a fase de coleta de conhecimento no processo de desenvolvimento da maioria dos SE. Elas conduzem o EC a um conjunto de elementos que facilitarão todo o processo de modelagem.

2.6.2 Conhecimento Bruto (CB)

Na realidade partindo-se do pressuposto que o conhecimento é proveniente da semântica construída pelos dados e as informações, e que o mote principal dos KBS é a geração de conhecimento nos mais diversos escopos, uma assertiva verdadeira é que todo o produto da coleta de conhecimento de fato representa no bojo informação a ser tratada ou interpretada. Somente o resultado deste tratamento dará origem ao processo de modelagem.

2.6.3 Representação do Conhecimento (RC)

Acredita-se que a representação do conhecimento possa ser argumentada de 05 maneiras [19] que conduzem o leitor a um conceito formal: (1) a representação do conhecimento pode ser vista como um substituto dos nossos próprios objetos ou entidades que determinam nossas ações; (2) a representação do conhecimento é um conjunto de relacionamentos ontológicos; (3) a representação do conhecimento é um fragmento da teoria do raciocínio inteligente, expressa através de uma árvore de componentes; (4) como uma computação eficiente e pragmática; (5) como as expressões intermediárias do ser humano.

Interpretando de maneira sensível as assertivas acerca da representação do conhecimento, se pode concluir que: (a) cada objeto ou entidade requer uma representação baseada nas suas características (propriedades); (b) os objetos ou entidades fornecem um *framework*[20] que caracteriza um conjunto de representações; (c) que, digamos os "desarranjos" anteriores, na realidade fornecem um *framework* com uma variedade de representações; (d) e finalmente o que se pode afirmar é que a visão da representação do conhecimento reflete na fonte de pesquisa e na implementação pratica do modelo.

Como visto o cerne da representação do conhecimento é primordialmente refletir através de algum formalismo de representação o conhecimento que foi coletado de um domínio. Existem alguns destes formalismos que comentaremos em linhas gerais, até para que se tenha uma visão das limitações e vantagens de cada um deles na elaboração do modelo.

[20] Framework é um conjunto de ferramentas ou elementos que viabilizam a solução de um problema.

2.6.3.1 Object-attribute-value triplets (OAV)

É um formalismo no qual se representam dados em uma base de conhecimento e também fornece suporte para possíveis representações heurísticas.

O OAV se interessa em particular por entidades ou objetos específicos na representação e cada um deste deve ter seus respectivos atributos. Para cada atributo no OAV existe um valor ou conjunto de valores associados. Observa-se na Figura 2.11 um exemplo gráfico da representação deste formalismo.

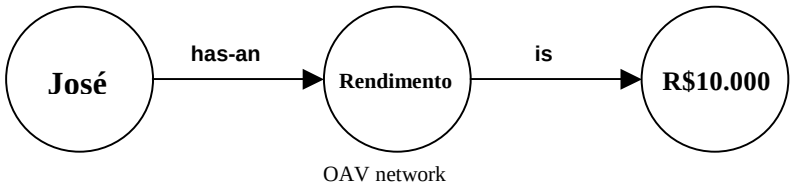


Figura 2.11 Representação do formalismo OAV

Observe no diagrama acima (Figura 2.11), que o objeto é José, o atributo é o rendimento dele e o valor do rendimento dele é R$10.000.

2.6.3.2 Rede semântica

Podemos inclusive afirmar que sofreu uma certa influência estrutural do formalismo OAV. “A semantic network may be thought of as a network that is composed of multiple OAV triplets in network form” [11]. As redes semânticas podem representar vários objetos, assim como, vários atributos por objetos, criando inclusive uma certa relação hierárquica entre estes elementos. Na Figura 2.12 um exemplo de rede semântica.

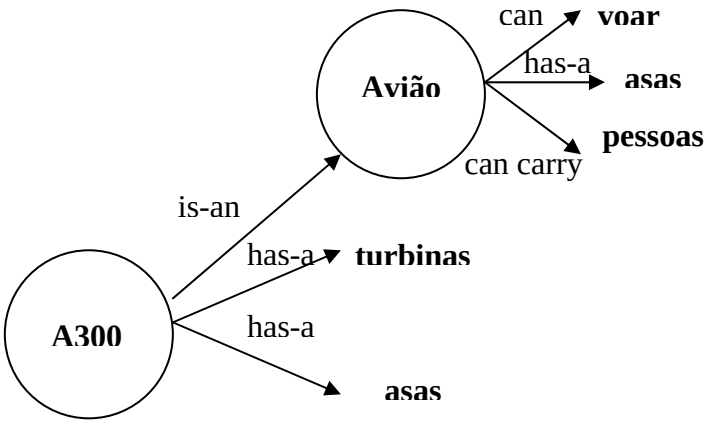


Figura 2.12 Rede semântica

É importante que se observe que o digrama das redes semânticas de fato fornece uma abordagem consistente para representação de associações entre entidades.

2.6.3.3 Frames

Observe que a impressão que nos passa é que todos esses formalismos são na realidade complementares um do outro – e na verdade são, dependendo obviamente da abordagem.

As redes semânticas fornecem a versatilidade de representarmos objetos e atributos de maneira ordenada, já os frames possibilitam a captura de detalhes ao nível de atributos e valores que as redes não permitem. “A frame contains an object plus (slots) for any and all information related to the object” [11]. Na Figura 2.13 um exemplo de frame:

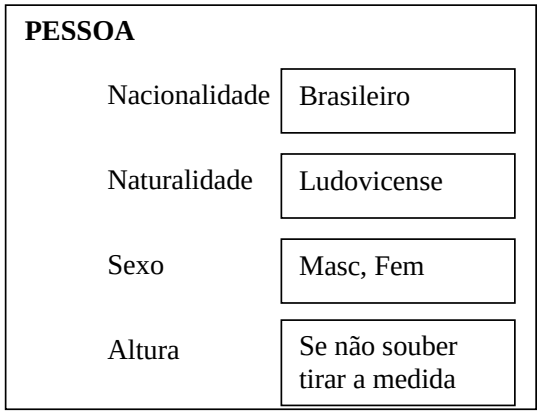


Figura 2.13 Representação do Frame

2.6.3.4 Sentença lógica

O outro formalismo, talvez considerado o mais comum é a representação através de **sentença lógica** usando o que se conhece como *lógica proposicional*. A lógica proposicional assume valores para as sentenças que podem ser falsos ou verdadeiros e lançam mão de conectores como AND, OR, NOT para compor as sentenças. Vejamos como se dá esta representação. Imaginemos três sentenças X, Y, Z e digamos que seu estado inicial é: X e Y verdadeiro e Z falso. Desta maneira podemos concluir o seguinte: X and Y são verdadeiros enquanto que X and Z é falso. Isso nos reporta a tabela verdade que diz que se duas sentenças verdadeiras são conectadas pelo AND certamente o resultado será verdadeiro.

2.6.3.5 Redes neurais

A grande vantagem de se usar este formalismo na representação do conhecimento é a sua proximidade com as linguagens naturais.

As **redes neurais** do ponto de vista do conexionismo tem se mostrado o formalismo mais próximo do conhecimento humano para sua representação. Isso ocorre em face da replicação deste conhecimento através de algoritmos neurais em um hardware (a mimetização do neurônio biológico a partir de um processo de aprendizagem). O pensamento humano se processa por meio de uma maciça quantidade de neurônios interconectados, e a grande vantagem é a plasticidade do cérebro [10]. Ela, baseada nos estudos científicos de pensadores como Jean Piaget e outros evolui dentro de um contexto social (teoria construtivista)[21].

Visto que as redes neurais são uma alternativa eficiente na representação do conhecimento humano principalmente se associadas a sistemas especialista na intenção de otimizar a performance na execução de tarefas, vejamos um pequeno modelo de uma rede neural na Figura 2.14.

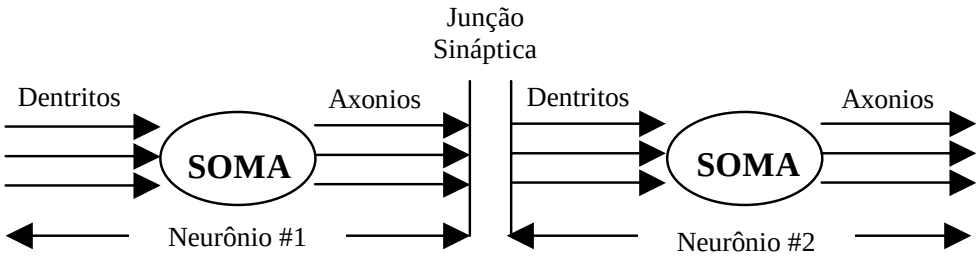


Figura 2.14 Rede neural

2.6.4 Modelo Conceitual (MC)

Este modelo também conhecido como modelo de *expertise*[22] é o conjunto documental de todo o processo de coleta e modelagem do conhecimento. Este conjunto, composto por vários documentos pode ser gerado a partir de alguns métodos/ferramentas como ontologias.

O modelo conceitual, digamos é o produto conseqüente do conhecimento que dará suporte ao processo de implementação do artefato. A intenção é falarmos agora sobre alguns

[21] A **teoria construtivista** desenvolvida por **Jean Piaget** se baseia no fato de que, o ser é um elemento envolvido em um contexto social e o seu processo evolutivo depende deste contexto.
[22] Expertise = especialista de domínio. Aquele que detém todo o conhecimento acerca de determinada tarefa dentro de um domínio ou do domínio como todo.

detalhes das ontologias. Essa abordagem nos dará compreensão acerca da estruturação do conhecimento.

Sera abordado agora o que vem a ser **ontologias** e suas aplicações e de que maneira as ontologias propiciam a modelagem conceitual do conhecimento.

Primeiro uma definição bem clara "*ontologia é uma descrição de conceitos e de relacionamentos que podem existir entre estes conceitos em um determinado domínio"* [11].

A ontologia quando usada como ferramenta de engenharia a serviço da inteligência artificial, visa essencialmente descrever um ambiente real através de um vocabulário específico fundamentado em um conjunto de preceitos com base no sentido das palavras deste referido vocabulário.

> "A hipótese de **Sapir-Whorf(1956)** afirma que a linguagem que falamos influencia profundamente o modo como pensamos e tomamos decisões, em particular definindo a estrutura de categorias pela qual dividimos o mundo em diferentes objetos" [17].

As ontologias via de regra são aplicadas a domínios complexos, onde conceitos como *ação, tempo, objetos físicos e crenças* interagem de maneira determinante [17]. A interação estruturada destes conceitos dá origem ao que se chama de engenharia ontológica.

A representação da estrutura geral destes conceitos é configurada através de um grafo onde os elementos da generalidade do domínio são expostos na parte superior, daí a nomenclatura "ontologia superior" o que se leva a concluir que os conceitos abaixo desta referida ontologia são especificações dela como mostra a Figura 2.15.

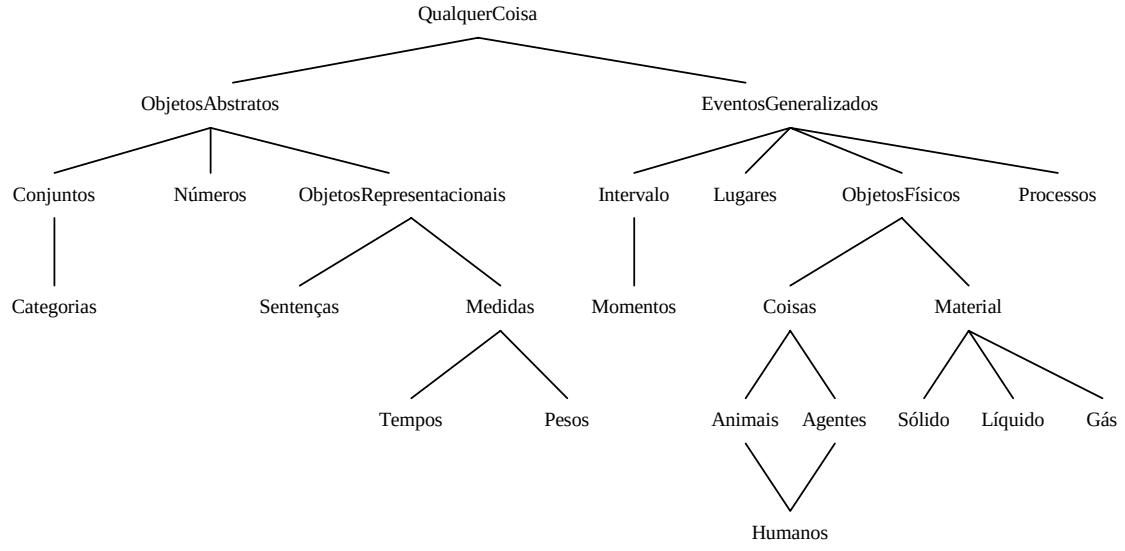


Figura 2.15 Ontologia superior do mundo

Existe um procedimento baseado na lógica para que se construa ontologias. A lógica proposicional apesar de útil se mostra limitada quando o desejo é expressar o mundo real.

Isso ocorre em face desta ter uma baixa capacidade em descrever de maneira sintética uma variedade de objetos muito grande. A opção neste caso é a LPO[23] (Lógica de Primeira Ordem), pelo processo de contextualização da realidade. *"Em uma linguagem composicional, o significado de uma sentença é uma função do significado de suas partes"*[16].

A citação acima se baseia essencialmente em dois aspectos: (1) a linguagem natural infelizmente sofre ambigüidades que perturbam o processo de comunicação; (2) a LPO consegue agregar associação sintática e semântica em uma mesma sentença, definido objeto relação e função de maneira clara, ou seja, os contextos do mundo real.

Agora fica evidente que além do compromisso da verdade ou falsidade da lógica proposicional a LPO se compromete com as ontologias do ponto de vista principalmente da relação entre os objetos do mundo.

Enfim, as ontologias podem utilizar vários tipos de lógica para representar o mundo (i.e. lógica temporal, lógica de alta ordem, teoria da probabilidade, etc.), o importante é notar que a fusão de várias lógicas ou a utilização de um único formalismo lógico depende exclusivamente da complexidade do universo a ser representado.

Levando-se em consideração a complexidade do universo (quanto mais complexo maior o refinamento em categorias) uma construção ontológica explora detalhadamente através da LPO as situações, o espaço, a temporalidade as crenças e as ações, visando representar rigorosamente o domínio dentro de uma base de conhecimento como será visto a seguir:

O cerne da representação do conhecimento, como já foi comentado acima é a representação do mundo em categorias. Isso se dá, visto que, a partir das características dos objetos do mundo real um sistema inteligente, percebe tais objetos partindo de suas propriedades para a construção de uma taxonomia.

Na LPO há duas maneiras para se categorizar os elementos do mundo real: através de predicados e de objetos (i.e. *CarroDeCorrida(c)* – predicados, *CarrosDeCorrida* - materialização dos objetos na categoria).

A seguir algumas demonstrações das relações através da LPO:

[23] LPO Lógica de Primeira Ordem

Elemento(c, CarrosDeCorrrida) significa que o elemento *c* pertence a categoria *CarrosDeCorrida*

A idéia principal da organização em categorias é fazer com que o conhecimento estabeleça uma relação de herança entre os objetos que o compõem, ou seja, *CarrosDeCorrida* $\subset$ *em Carros* a categoria *CarrosDeCorrida* esta contida dentro da categoria *Carros*.

Um outro objetivo deste processo taxonômico é explorar as propriedades dos objetos dentro das relações como é mostrado a seguir:

$x \in$ *CarrosDeCorrida* $\Rightarrow$ *Velozes(x)* os carros de corrida são necessariamente velozes.

Uma vez entendido o mecanismo de construção das categorias é importante que se leve em consideração algumas formas mais explicitas de demonstra-las em suas relações.

Os objetos das categorias precisam ser compostos, medidos, analisados a partir de sua ações, posições e eventos associados a eles:

Os motores potentes de um carro de corrida fazem parte dele, isto é demonstrado através de uma relação chamada *ParteDe* (i.e. *ParteDe(Ford, Benneton)*).

O que caracteriza esta relação é o processo de estruturação entre as partes dos objetos que compõem a categoria.

Todos os objetos que compõem o mundo real tem características que determinam as suas formas. Alguns valores precisam ser atribuídos a estes objetos, chamados medidas. O grande desafio não é a atribuição de medidas a objetos, mas mensurar objetos abstratos (i.e. *pode-se medir um cubo, porém, como medir a beleza de uma paisagem?*).

"Algumas categorias têm definições estritas: um objeto é um triângulo se e somente se ele é um polígono com três lados. Por outro lado, a maioria das categorias no mundo real não tem nenhuma definição clara"[16] isso é o que se denomina categoria de **espécies naturais.**

Esse problema leva a uma abordagem chamada de instanciação de objetos em categorias "típicas" deles. Isso é claramente demonstrado através da lógica filosófica redigida no *Tractatus Logico-Philosophicus (Tratado Lógico Filosófico)1929* do pensador vienense Ludwig Wittgenstein.

A partir de sua influência neopositivista ele defendia veementemente as imperfeições da linguagem natural. Daí a grande dificuldade da classificação pragmática dos objetos.

Um outro aspecto interessante do processo de taxonomização dos elementos do mundo é a individuação (i.e. imaginemos que se tenha uma pessoa e uma barra de sabão. Se eu partir uma pessoa ao meio certamente não terei duas pessoas porem, se partir uma barra de sabão, terei barras pequenas de sabão até chegar a ponto de indivisibilidade, ou terei outras propriedades que individualmente não caracterizem mais o sabão - outro objeto qualquer).

Esse fenômeno é classificado pela linguagem natural em dois grupos: substantivos contáveis (sabão, queijo, madeira, etc.) e substantivos de massa (cachorros, pessoas, peixes, etc.) e o que os caracteriza são as suas propriedades extrínsecas e intrínsecas respectivamente. Em LPO isso fica assim: $x \in Sabão \wedge ParteDe(y,x) \Rightarrow y \in Sabão$.
É desta forma que a ciência ontológica fornece subsidio para taxonomizar o domínio e entender claramente a semântica de seus elementos.

2.6.5 Design / Implementação do artefato

Esta fase detalha todo processo de elaboração do sistema especilista baseado nos modelos desenvolvidos acima. Nela será discutida a arquitetura e o projeto do mesmo, assim como toda a implementação da base de conhecimento. Um capítulo adiante trada especificamente desta abordagem. A construção do sistema especialista.

Capítulo 3 Modelagem do Conhecimento

Neste capítulo será visto:

- O ciclo de vida do WST Web Services Telediagnosis
- O levantamento do conhecimento acerca da malária;
- A elaboração do modelo de conhecimento;
- A construção da base regras do domínio de diagnóstico da malária.

3 MODELAGEM DO CONHECIMENTO

Os mecanismos de raciocínio encontraram inspiração para o seu desenvolvimento no campo da Inteligência Artificial basicamente por duas vias: pelos *sistemas adaptativos* (sistemas que interagem com o meio para o cumprimento de tarefas) e pela fonte maior de observação – o ser humano.

O objetivo da modelagem é primordialmente **tentar mimetizar** o comportamento. Este por sua vez representado pelo conhecimento tem se apresentado como um dos maiores desafios quando a questão é o desenvolvimento de algum artefato que demande ações inteligentes, autônomas ou não.

O que se percebe, baseado na filosofia racionalista, visando justificar a assertiva do parágrafo acima é que todo o processo de teorização se dá a partir da observação de evidências empíricas.

A ciência é o conhecimento estruturado, que tem por objeto, uma (hipotética) realidade universal [18]. Esse é o fato gerador da construção de modelos que "espelhem" a realidade ou um fragmento dela.

O homem espera que a realidade siga uma "linearidade" (o que facilitaria o desenvolvimento de modelos dentro dos mais diversos contextos), ou seja, que o comportamento tenha uma funcionalidade regular, que existam leis e modelos explicativos do mundo [17].

No entanto ao contrário do que se espera quanto à linearidade do comportamento humano, como resposta resta a negação ou revisão de fatos tidos como verdade.

Os modelos são desenvolvidos através de processos de indução, ou seja, o pesquisador a partir de suas observações cria padrões que, de acordo com o comentado acima podem e devem ser refutados à medida que haja mudança de contexto.

Isso prova que estes modelos não podem, apesar das observações no campo do empirismo ter este traço apenas de conjecturas nem de inflexibilidade.

> "O conhecimento científico avança, portanto através da análise teórica de evidências empíricas e a formulação de novas hipóteses dentro dos paradigmas, a fim de validar o paradigma frente às evidências e através da proposição de novos paradigmas que possuam maior poder heurístico que os anteriores" [33]

De acordo com o exposto acima, a modelagem ganha um poder descritivo do mundo real, sensível a ótica do pesquisador e ao seu domínio de observação, esse é o mais significante instrumento no desenvolvimento de um sistema especialista.

O objeto da modelagem de conhecimento contida neste documento é o processo de anamnese acerca da malária. Um sistema especialista será originado a partir deste modelo e colocado no ambiente de serviços da internet. A idéia é que esse modelo possa identificar os avanços no diagnóstico desta doença que acomete uma faixa geográfica equivalente a 1/3 do planeta.

Acredita-se que esteja esclarecida a importância do trabalho de estruturação do conhecimento através de modelos. Como foi visto acima no capítulo 2, estes modelos partem essencialmente da observação inicialmente empírica de características do domínio a ser construído.

A seguir o detalhamento das fases para que se alcance um modelo de conhecimento para um dado problema dentro de um domínio (diagnóstico da malária).

3.1 Ciclo de vida do *WST Web Services Telediagnosis*

Nesta seção será percorrido todo o ciclo de vida de desenvolvimento de um KBS, para a modelagem do conhecimento acerca da malária, e se terá um modelo conceitual que irá permitir desenhar uma base de conhecimento e implementar um sistema especialista, com base na estrutura abaixo:

- Fonte de conhecimento (malária)
  - Conhecimento coletado / bruto (detalhes do ponto de vista clínico da doença)
    - Representação do conhecimento (definição dos objetivos da base de conhecimento e dos fatos que irão compô-la)
      - Modelo conceitual (construção das ontologias do domínio da malária)
        - Design (definição das regras que irão compor o mecanismo de inferência do KBS)

3.1.1 Fonte de Conhecimento - Malária

A malária é uma doença infecciosa marcada por evoluir cronicamente através de manifestações periódicas que via de regra tem um caráter agudo e passa por alguns períodos de latência que podem se confundir com uma cura aparente.

Caracterizada por um processo de **epidemiologia caprichosa**, ou seja, a malária tem o seu “comportamento” epidemiológico associado aos vários elementos naturais do ambiente onde ocorrem as endemias. Esta doença não pode ser analisada puramente pela relação entre células e o parasita.

Primordialmente se encontra explicação para o ciclo de contaminação da malária nos aspectos relacionados ao desequilíbrio natural do ambiente – em face disso a malaria ter a sua maior incidência nos bolsões de miséria do mundo.

A malária nos dias de hoje ainda é considerada a doença mais importante do mundo.Alguns fatos levam esta doença a ocupar este lugar como se observa na história e nas características marcantes desta doença.

3.1.1.1 Breve histórico da doença

A malária teve sua origem provavelmente no Continente Africano ainda na pré-história e se dissipou pelos processos migratórios por todo o Mediterrâneo, Mesopotâmia, Índia e sudeste Asiático.

A transmissão da doença para os outros continentes é um mistério até hoje, mas, o que se supõe é que tenha se dado também por fluxos migratórios na fase das grandes viagens colonizadoras no século XVI [26].

Apesar da malária ter passado por um período envolta em um clima de misticismo, até, por não compreenderem (os povos da época) bem o seu ciclo de transmissão e a sua sintomatologia, no século V a.C. Hipocrates, na Grécia, afastou todo e qualquer tratamento para a doença que não fosse aquele que estivesse a luz da ciência.

Este foi o primeiro a descrever o processo de contaminação e as conseqüências desta contaminação nas populações.

Mais tarde, por volta do século II d.C. alguns médicos gregos e romanos também se empenharam em pesquisas sobre o assunto, visto que, epidemias cíclicas ocorriam naquelas regiões (Grécia, Itália e algumas partes da Europa). A malária era conhecida como "*Febre Romana"*.

Ao longo de um período significativo para a ciência, a malária esteve no esquecimento, ainda que fazendo vitimas pelo mundo todo.

Somente no século XVII através da observação por padres jesuítas quanto ao uso de uma droga pelos índios (da América do Sul) para cura de diversos tipos de febre é que os jesuítas chegaram a uma pista para o desenvolvimento de um tratamento para cura da malária [28].

A descoberta foi imediatamente difundida pela Europa e remédio levou o nome de "*Pó dos Jesuítas"*.

A droga tinha origem a partir da casca de uma arvore chamada "*Cinchona"* que produzia um principio ativo conhecido como "*quinino"*. O quinino foi isolado em laboratório já em 1820.

Infelizmente apesar do progresso, a malária ainda tinha muito campo para descobertas. No século XVIII a Europa ainda não sabia exatamente como se dava o processo de transmissão. Existia a hipótese de que as doenças eram provenientes do mau cheiro de pântanos e alagadiços, o que a levou ser batizada de "*mal-are"*.

Só a partir do século XIX a malária de fato teve um impulso em suas descobertas científicas. Foi neste século que bacteriologistas e patologistas buscando explicação para outras doenças acabaram encontrando algo sobre a malária.

Em 1880 o médico francês Charles Alfonso Laveran, na época trabalhando na Argélia observou e descreveu o parasita da malária em um glóbulo vermelho e em 1897 o médico britânico Ronald Ross, na Índia, acompanhou o ciclo de transmissão da doença e finalmente em 1898 / 1899 os pesquisadores italianos Amico Bignami, Giuseppe Bastianelli e Batista Grassi consolidaram o ciclo de desenvolvimento do parasita no mosquito *Anopheles*[24] e no homem.

3.1.1.2 A Transmissão

A malária é uma doença transmitida por um mosquito do gênero Anopheles, basicamente de duas formas pelas espécies mais comuns A.darlingi, A.aquasalis como mostram as Figuras 3.1 e 3.2: (1) através da picada no homem pelo mosquito infectado com o plasmodium25, via de regra pela fêmea deste mosquito, visto que, ela necessita de sangue para suas funcionalidades no ciclo de ovulação da sua espécie; (2) através do contato com o sangue humano infectado – transfusão de sangue, compartilhamento de seringas.

[24] Anopheles é o gênero do mosquito transmissor da malária.
[25] Plasmodium é o gênero do protozoário unicelular que provoca a malária.

Figura 3.1 Anophelis darlingi

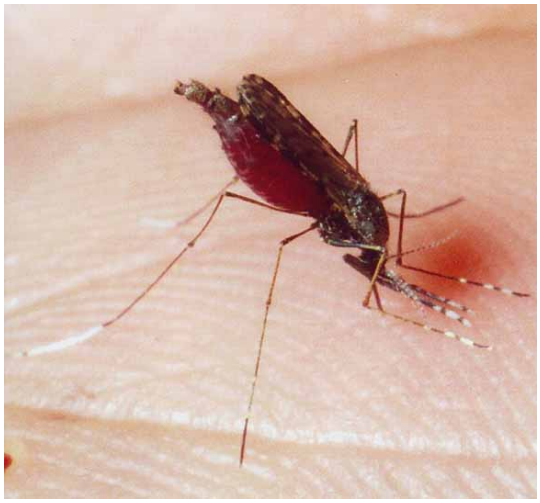


Figura 3.2 Anophelis aquasalis

Este parasita (da malária) pertence à ordem *Coccidiida* ⇒ sub-ordem *Haemosporidiidea* ⇒ Família *Plasmodiidae* ⇒ Gênero *Plasmodium*. As espécies de plasmódio que ameaçam o homem são: [26]

- *Plasmodium vivax* (*Grassi & Feletti*, 1890)
- *Plasmodium falciparum* (*Welch*,1897)
- *Plasmodium malariae* (*Laveran*, 1881)
- *Plasmodium ovale* (*Stephens*,1922)

A Figura 3.3 mostra as células vermelhas infectadas com os dois principais tipos de *plasmodium* (o *vivax* e o *falciparum*).

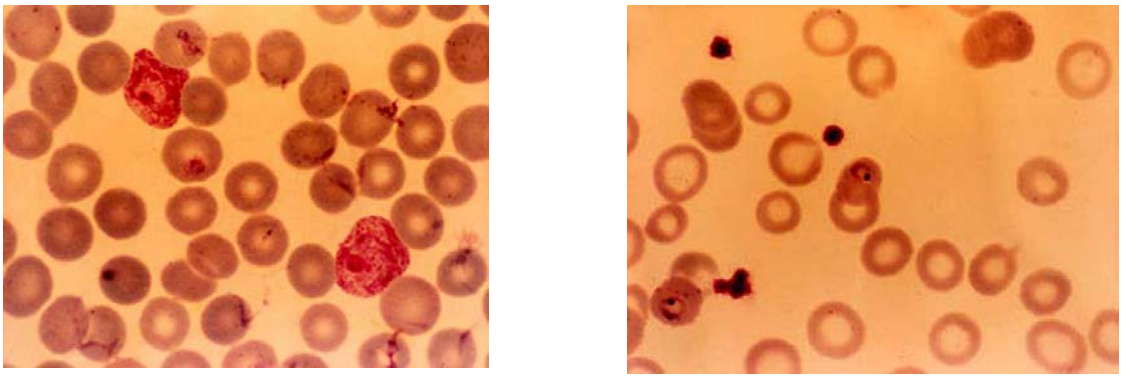


Figura 3.3 Hemácias (glóbulos vermelhos) infectadas por *plasmodium*. A esquerda o *plasmodium vivax* e a direita a forma mais letal da doença o *plasmodium falciparum*. Estes são os tipos mais comuns no Brasil.

O parasita basicamente, nas suas quatro formas tem o mesmo ciclo de reprodução. Uma fase sexuada exógena (*esporogonia*), onde ele se reproduz em alguns gêneros de

Anopheles e outra assexuada endógena (*esquizogonia*) onde a reprodução se dá em seu hospedeiro final – o homem.

É preciso que se pontue estes ciclos em detalhes para que se possa entender profundamente como se dá a transmissão da doença. Primeiro observa-se como o parasita se reproduz no mosquito.

Um mosquito "sadio". Os *anopheles* fêmea ao contrário do macho, que se alimenta de seiva de plantas, se alimenta de sangue para, como já foi comentado acima, compor a sua ovoposição (seu ciclo de reprodução).

Desta forma quando a fêmea do mosquito ingere sangue contaminado (com gametócitos[26]) de um hospedeiro humano inicia-se um processo sexuado no estômago do mosquito que leva a formação do ovo do parasita (zigoto).

Após este processo, o zigoto migra através da camada unicelular do estômago do mosquito alojando-se entre este e sua membrana basal. A partir deste ponto, por esporogonia[27] se dá a multiplicação das formas infectantes (esporozoitas).

Estas formas migram novamente para as glândulas salivares do mosquito sendo inoculadas através da picada em um potencial novo hospedeiro humano. A Figura 3.4 demonstra o vetor de contaminação da malária.

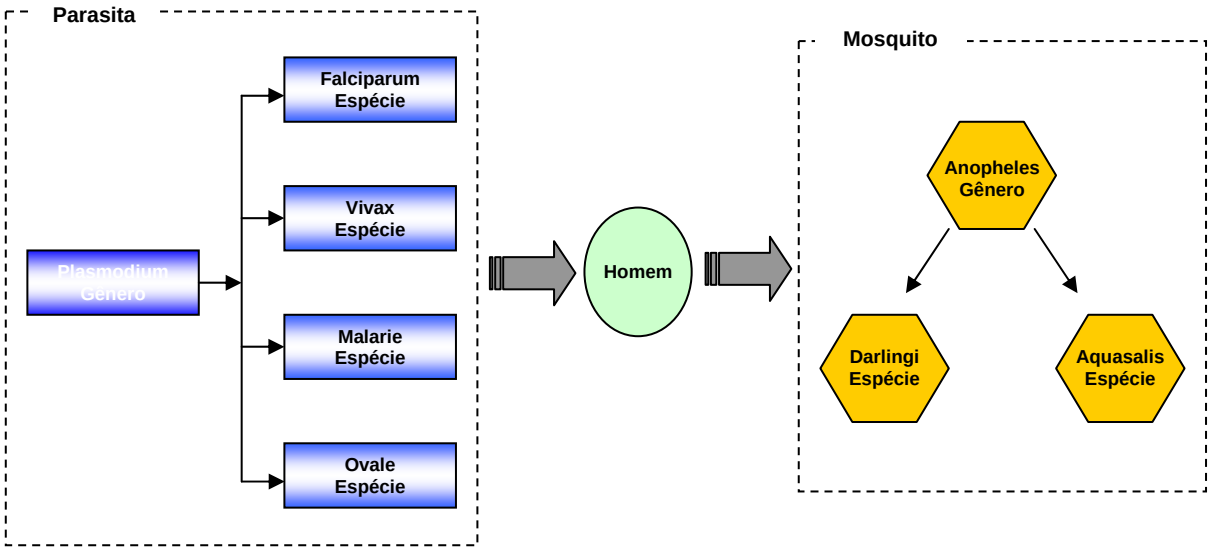


Figura 3.4 Vetor de contágio da malária.

[26] Gametócitos são as formas sexuadas do parasita.
[27] Esporogonia é o processo de reprodução de formas infectantes conhecidas como esporozoitas.

O hospedeiro humano, após ser picado pelo mosquito infectado com a forma esporozóita passa aproximadamente 30 minutos com esta forma em circulação livre pela corrente sanguínea.

Neste momento alguns esporozóitas são fagocitados[28], porém a grande maioria encontra o fígado e as células hepáticas, a partir deste estágio se dando a primeira divisão assexuada do plasmódio (esquizogonia tecidual).

Após este processo de reprodução, as células do fígado se rompem e os plasmódios têm acesso ao sangue invadindo os glóbulos vermelhos onde novamente se multiplicam assexudamente (esquizogonia eritrocítica[29]) em ciclos variáveis de 24 a 72 horas, cada parasita produzindo 8 a 32 novos exemplares.

A partir de três ou quatro ciclos destes iniciam-se os primeiros sintomas da doença. É este período abordado acima, que é conhecido como fase de incubação da doença e dura em média 15 dias.

3.1.1.3 Sintomatologia

A malária é taxionomizada como grave e não complicada. Via de regra os casos considerados graves são causados pelo *plasmódio falciparum* e os demais pelo *vivax, malarie e ovale*.

Isso não significa que pacientes que contraiam malária a partir das formas menos agressivas do parasita não corram o risco de evoluir para quadros graves.

As sintomatologias dos dois casos são: febre, calafrios, sudorese e dores na cabeça e no corpo. Esta sintomatologia básica é conhecida como tétrade da malária.

Outras manifestações podem ocorrer visto que a malária é uma doença que acomete todo o corpo humano (náuseas, vômitos, fraqueza, fadiga, diarréia, dores articulares e abdominais, icterícia e palidez acentuada) [28].

A malária grave tem os mesmos sintomas e manifestações citadas acima para os casos identificados como *vivax, malarie e ovale,* porém em face da severidade da forma como atua o *P.falciparum*, as seqüelas, caso o paciente não receba um cuidado especial, podem leva-lo a comprometimento cerebral, renal, pulmonar e disfunções na coagulação sangüínea.

[28] Fagocitose é um processo de alimentação de muitos protozoários unicelulares, que consiste no englobamento de partículas sólidas pela célula...[28]
[29] Eritrócitos são os glóbulos vermelhos.

3.1.1.4 Diagnóstico

Como foi comentado acima mais que uma doença que se tenha que levar em consideração os aspectos clínicos a malária tem todo um cunho social. Este ajuda criar alguns critérios que indicam no diagnóstico, a possível área onde o paciente contraiu a doença. A seguir o que significam no diagnóstico os termos: [26]

- **Suspeito** é todo paciente com febre e outros sintomas **procedentes de área endêmica**;
- **Confirmado** é o caso suspeito com o parasita em **sangue periférico;**
- **Autóctone** é o caso contraído na área em que reside o paciente;
- **Importado** é o caso contraído fora da área;
- **Introduzido** é o caso derivado de um importado.

Apesar da letalidade da malária em muitos casos e do seu caráter prevalente existe diagnóstico seguro e tratamento extremamente eficaz para a doença.

Dois procedimentos determinantes são tomados nos casos suspeitos de contaminação:

1. Investigação clínica anamnética e observação da procedência do suspeito;
2. Diagnóstico parasitológico simples através da coleta de sangue da polpa digital (gota espessa).

O resultado deste último exame identifica duas informações patológicas: (a) o tipo de *plasmodium* e (b) a carga parasitária representada pelo numero de cruzes.

3.1.1.5 Tratamento

A iniciativa central no combate a malária sempre se estabeleceu em duas frentes: (1) o combate ao mosquito; (2) a cura da doença instalada no homem. O *quinino* é a principal droga para a cura da doença e em 1942, Paul Muller desenvolver um inseticida a base de **dimetil difenil tricloretano DDT**.

Confirmado o diagnóstico as doses do *quinino* são ministrada a cada caso dependendo da forma parasitária do *falciparum*. Existem tabelas com valores pré-determinados para atender às faixas de casos da doença.

3.1.2 Representação do conhecimento

Para que se chegue ao modelo conceitual de qualquer domínio é necessário após ter passado pelas etapas referentes à coleta de conhecimento, refinar este conhecimento em busca dos objetivos que nos proporcionarão a identificação dos fatos acerca deste.

Observe na Figura 3.5 os passos a serem seguidos e as suas respectivas descrições nas seções 3.1.2.1 e 3.1.2.2, uma vez coletados os conhecimentos necessários para implementação de um modelo.

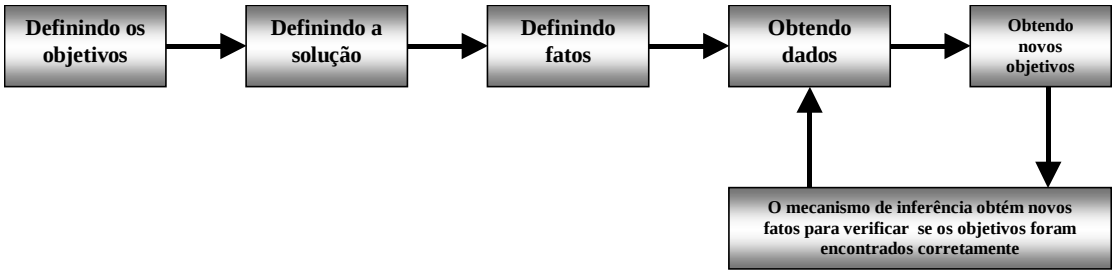


Figura 3.5 Componentes de um sistema de IA baseado em regras

3.1.2.1 Objetivo

O diagnóstico da malária se dá através do exame laboratorial (sangue espesso) e da investigação clínica (anamnese). O escopo desta pesquisa é colocar a disposição em regiões inóspitas um artefato inteligente como já foi comentado acima na proposição do projeto.

Com base nesta intenção o objetivo primordial neste primeiro momento é **exatamente identificar se a entidade paciente a partir de uma anamnese superficial, que será submetida a um mecanismo de inferência, está ou não com a malária**.

No entanto que se entenda que os objetivos estarão alcançados, ou seja, ter-se-á uma solução se as respostas satisfizerem ao contexto da doença.

- Será que a entidade paciente está com malária?

3.1.2.2 Fatos

Os fatos são o ingrediente essencial em um sistema de IA[11]. Para que alguém esteja com malária um conjunto de sintomas e/ou eventualidades pode ter acontecido. Estes elementos darão sustentação ao nosso objetivo. Então abaixo, vejam como isso é representado:

- O paciente apresenta febre alta e intermitente;
- A temperatura esta entre 38 e 40 graus;
- Ele esteve em área endêmica a menos de 30 dias;
- O paciente apresenta sudorese, calafrios, vômitos, fadiga e diarréia;
- Este apresenta os olhos amarelados e icterícia;
- O paciente teve contato com pessoas infectadas a menos de 3 anos.

Estes são alguns fatos entendidos como característicos de quem esta com malária. A partir deles algumas perguntas podem ser feitas para que se obtenha dados necessários a construção de nossas regras do sistema.

Observa-se que acima se tem a característica básica para a sintomatologia da malária agora que se proceda a uma anamnese no paciente:

1. Tem febre?
2. A febre é regular?
3. Qual a temperatura? Aproximadamente 40 graus?
4. Esteve em área de incidência da doença?
5. Há quanto tempo?
6. Sente calafrios?
7. Sudorese?
8. Náuseas?
9. Vômitos?
10. Fraqueza?
11. Fadiga?
12. Diarréia?
13. Dores articulares, abdominais?
14. Urina escura?
15. Icterícia?
16. Olhos amarelados?
17. Aumento do volume do fígado?
18. Alteração do volume do baço?
19. Teve contato a menos de três anos com pessoas infectadas?

Esta anamnese trará os dados necessários a serem submetidos ao processo de *pruning*[30] no mecanismo de inferência. Estes dados irão compor a agenda deste mecanismo.

3.1.3 Modelo Conceitual

No modelo conceitual será visto através de ontologia a representação dos elementos principais da coleta de conhecimento acima, organizados em classes e seus relacionamentos.

[30] Pruning = Poda – Eliminação dos fatos que não fazem parte do contexto de aplicabilidade das regras.

Será usado para elaboração desta etapa de construção de uma ontologia, o software Protege [21].

3.1.3.1 Hierarquia de classes para o projeto Malária

Tem-se para construção de uma ontologia de domínio simplificada da malária, visando um diagnóstico anamnético da doença no ambiente da tecnologia de serviços *web*, as três entidades principais deste referido domínio: o mosquito, o parasita e o homem. Estas três entidades compõem basicamente todo vetor de contágio da doença e dá origem ao diagrama de classes da Figura 3.6:

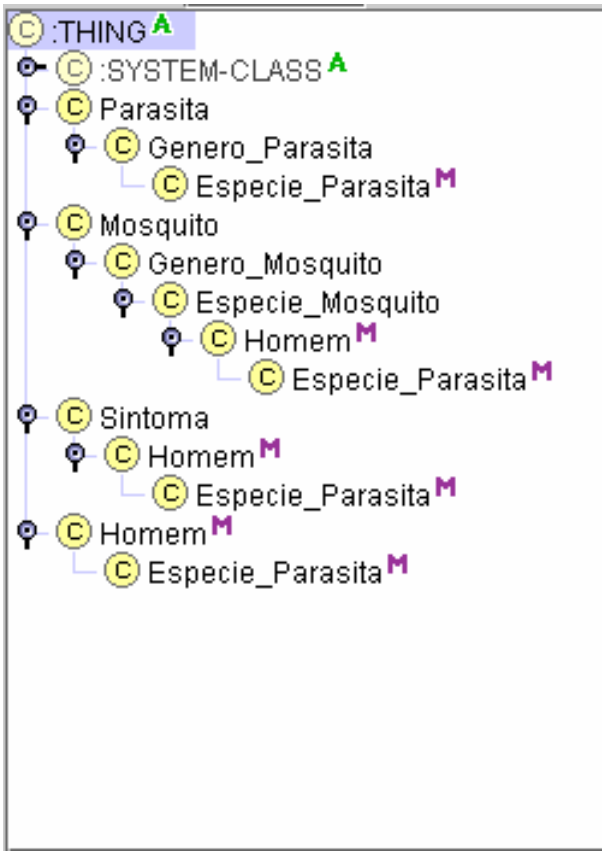


Figura 3.6 Hierarquia de classes para o vetor de contágio da malária

O diagrama acima demonstra dois aspectos dentro do domínio: (1) a relação do vetor de contágio representada pelas classes Parasita ⇒ Homem ⇒ Mosquito ⇒ Homem e; (2) em seguida uma outra relação de sintomatologia entre a classe Parasita ⇒ Homem ⇒ Sintoma.

A superclasse Parasita dá origem a duas sub-classes filhas – gênero e espécie – do parasita, Figuras 3.7 e 3.8, visto que, as características de causa e efeito da doença dependem da espécie do plasmodium contraído.

A sub-classe Gênero_Parasita possui um *slot* - n_gênero_parasita – que irá conter o nome do gênero do parasita. Abaixo desta sub-classe se encontra a sub-classe Espécie_Parasita que por sua vez possui o *slot* n_espécie_parasita – que irá conter o nome da espécie do parasita.

Esta última sub-classe (n_espécie_parasita) além do referido *slot* acima, recebe por herança os *slots* definidos nas classes Homem, Mosquito e Sintomas, visto que, existe uma relação na vetorização de contágio da doença direta com o parasita. Veja nas Figuras 3.7 e 3.8 os diagramas com os *slots* criados para a classe Parasita e suas sub-classes:

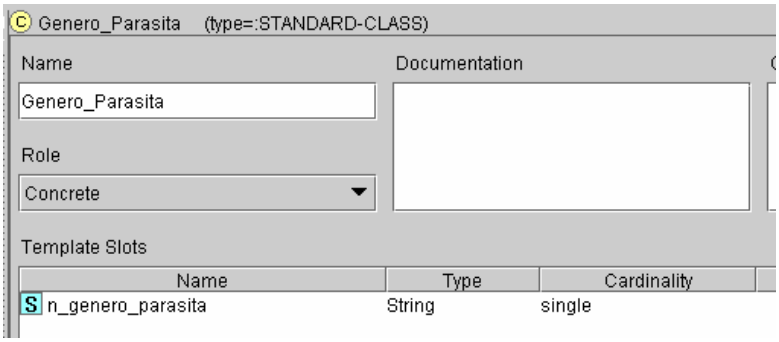


Figura 3.7 *Slot* definido para a classe Parasita e sua sub-classe Gênero_Parasita.

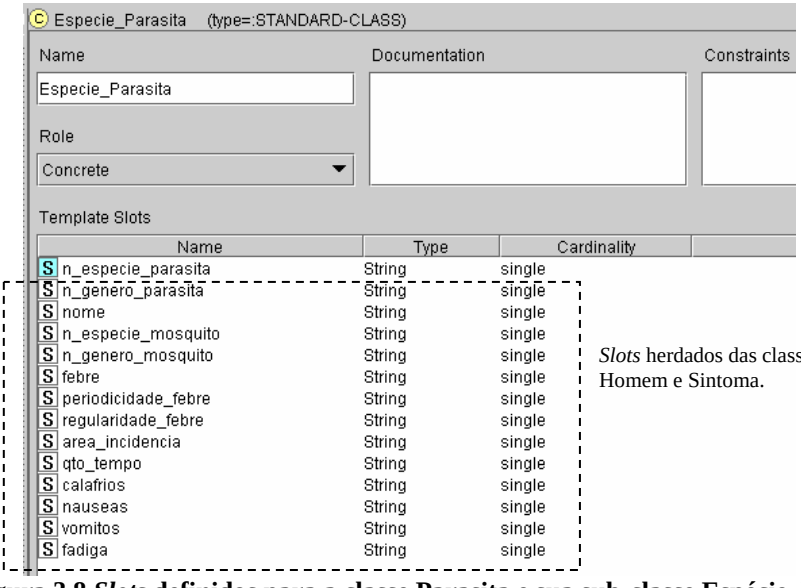


Figura 3.8 *Slots* definidos para a classe Parasita e sua sub-classe Espécie_Parasita.

A classe Mosquito possui as seguintes sub-classes: Gênero_Mosquito (Figura 3.9), Espécie_Mosquito (Figura 3.10), Homem (Figura 3.11) (uma sub-classe na verdade associada a classe Mosquito pela relação do vetor de contágio) e Espécie_Parasita (Figura 3.12) (idem a relação da classe Homem).

A sub-classe Gênero_Mosquito possui o *slot* n_genero_mosquito que irá conter o nome do gênero do mosquito (*Anopheles*). Por sua vez a sub-classe Espécie_Mosquito possui os *slots* n_especie_mosquito que irá conter o nome da espécie do mosquito e herda da classe Gênero_Mosquito o *slot* que conterá o nome do gênero do mosquito.

A classe Homem é herdada pela sub-classe Espécie_Mosquito a partir da associação que se dá através do vetor de contágio o qual já foi comentado acima. É herdado ainda através da classe Homem a sub-classe Espécie_Parasita trazendo com essa relação além dos *slots* peculiares de cada classe e sub-classe associada, todos os *slots* da superclasse Sintoma, visto que, cada espécie de parasita pode causar manifestações características daquela espécie.

A seguir na figura 3.10, os diagramas que definem os *slots* para esta parte da ontologia:

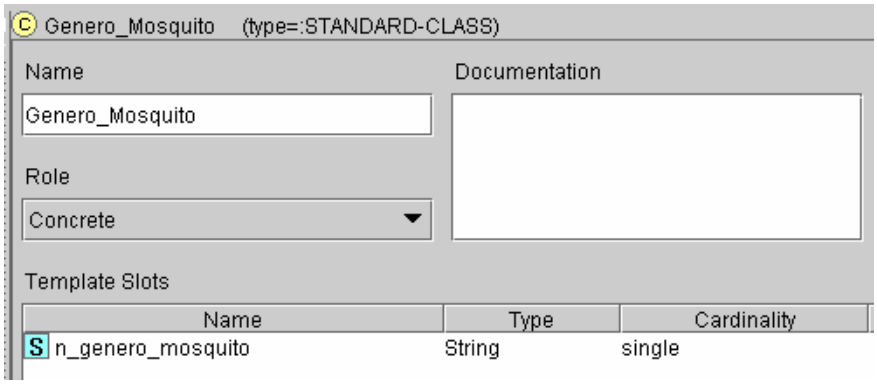
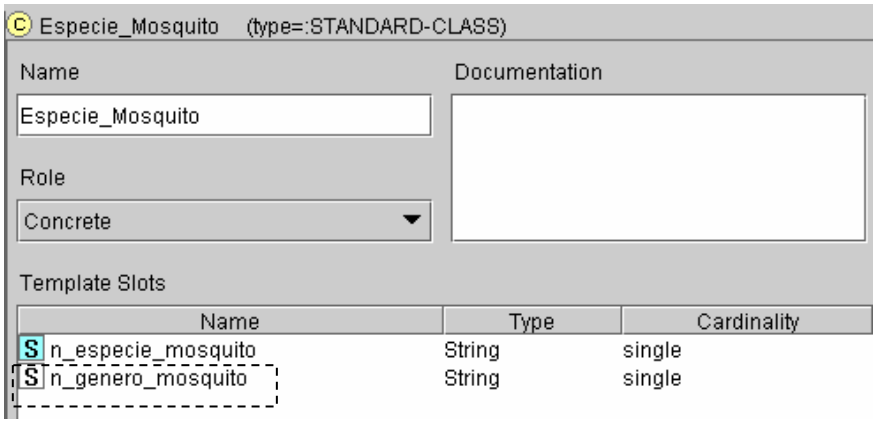


Figura 3.9 sub-classe Gênero Mosquito



Slot herdado da sub-classe Genero_Mosquito.

Figura 3.10 sub-classe Espécie Mosquito

Slots herdados das sub-classes gênero e espécie do mosquito e da superclasse Sintoma.

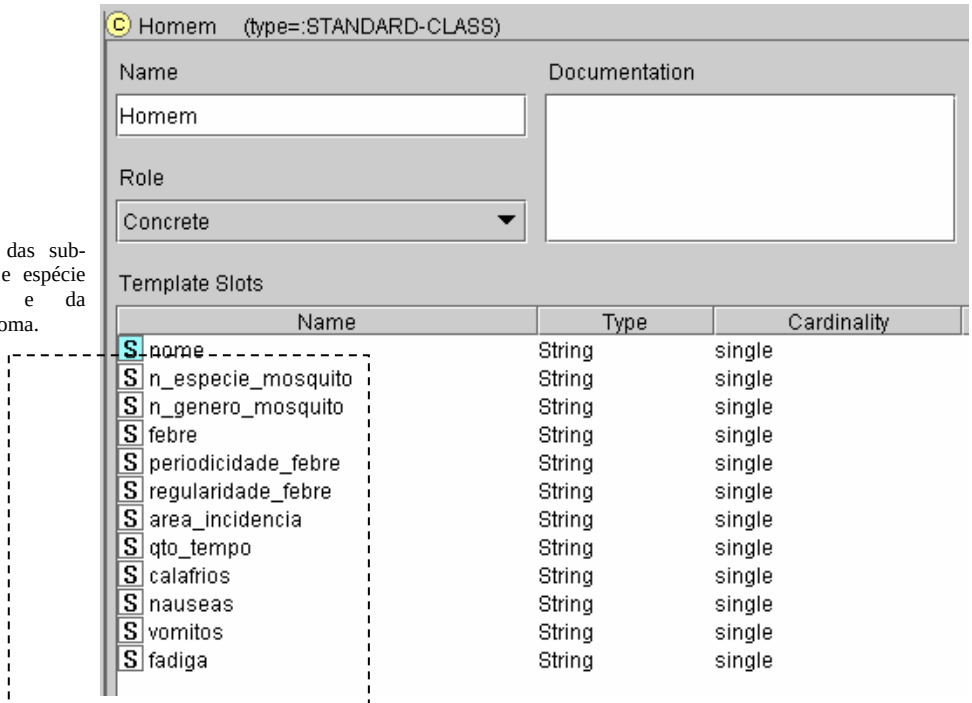
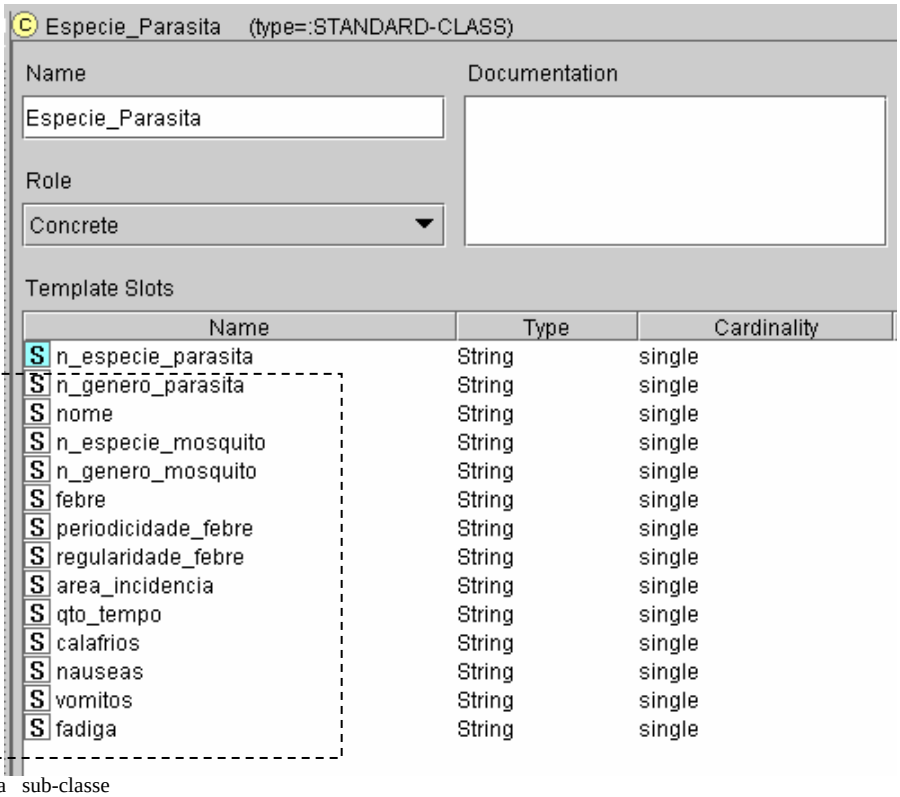


Figura 3.11 sub-classe Homem



Slots herdados da sub-classe Espécie_Parasita através da superclasse Homem.

Figura 3.12 sub-classe Espécie Parasita

Na superclasse Sintoma (Figura 3.13) encontra-se os slots que irão compor a anamnese sobre a malária. Esta superclasse (Sintoma) será associada à superclasse Homem que por sua vez esta associado à sub-classe Espécie_Parasita.

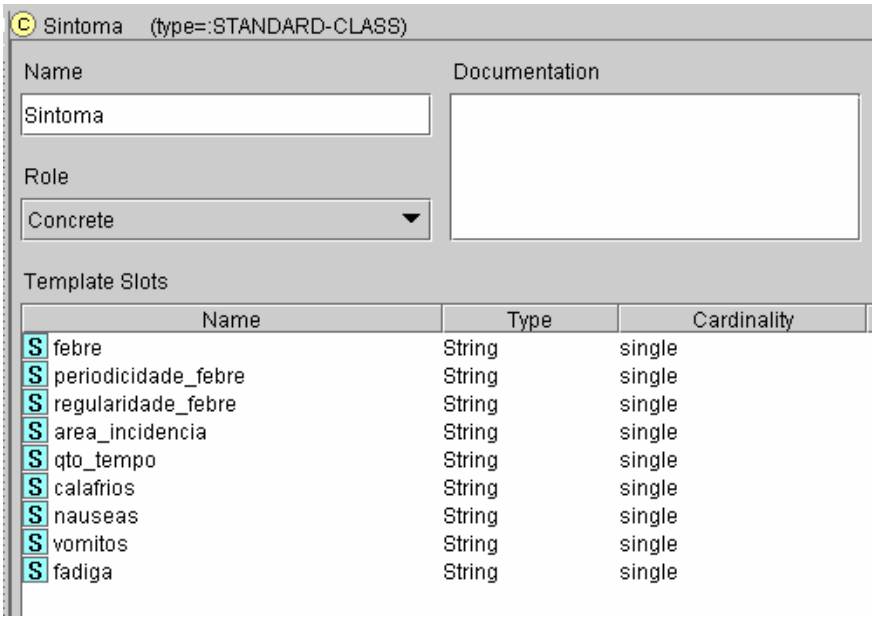


Figura 3.13 superclasse Sintoma

Finalmente, a superclasse Homem que possui apenas um *slot* nome, que conterá o nome do paciente e herdará da sub-classe Espécie_Parasita todos os *slots* que caracterizam o plasmodium responsável pela contaminação.

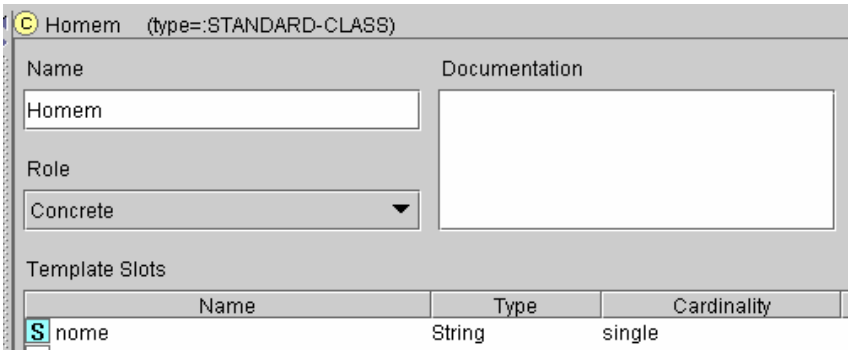


Figura 3.14 superclasse Homem

Algumas instâncias são criadas acerca das classes que representam o domínio de contágio da malária.

Na superclasse Parasita, sub-classe Gênero_Parasita temos a instância *Plasmodium*. Na sub-classe Espécie_Parasita tem-se as instâncias *Vivax, Ovale, Falciparum e Malarie* como mostram as Figuras 3.15 e 3.16

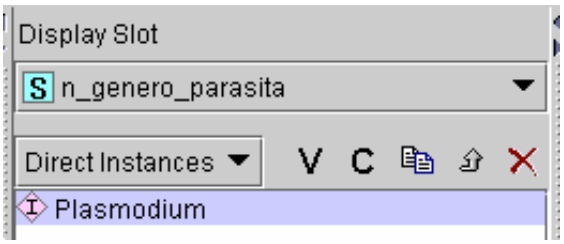


Figura 3.15 Instância da superclasse Parasita, sub-classe Gênero_Parasita

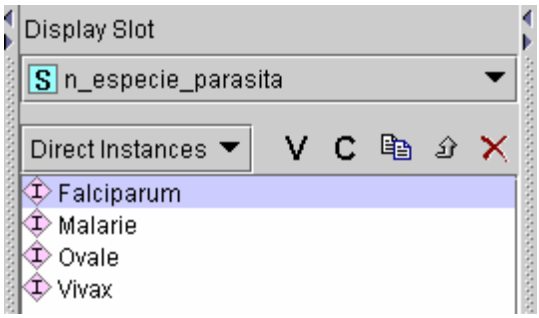


Figura 3.16 Instâncias da superclasse Parasita, sub-classe Espécie_Parasita

Na superclasse Mosquito, subclasse Gênero_Mosquito temos a instância *Anopheles*. Na sub-classe Espécie_Mosquito tem-se as instâncias *Aquasalis e Darlingi* como se vê nas Figuras 3.17 e 3.18.

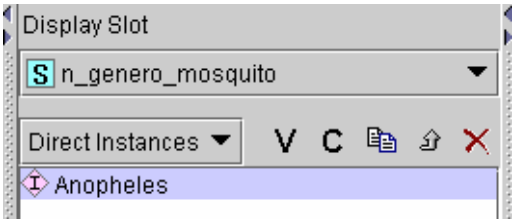


Figura 3.17 Instância da superclasse Mosquito, sub-classe Gênero_Mosquito

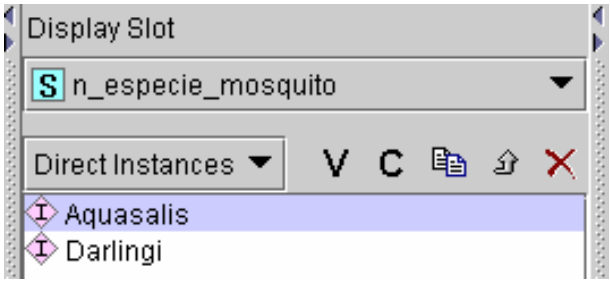


Figura 3.18 Instância da superclasse Mosquito, sub-classe Espécie_Mosquito

Na superclasse sintoma (Figura 3.19) encontra-se as duas situações necessárias para a simulação de diagnóstico no ambiente de serviços *web – positivo e negativo*. Este resultado será dado como resposta a partir da conclusão das regras que irão se basear no questionário de anamnese.

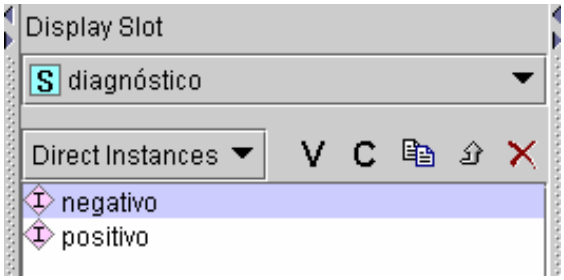


Figura 3.19 Instâncias da superclasse Sintoma

A última instância definida para compor a base conhecimento é a da superclasse Homem (Figura 3.20) – *Ana e Antonio*.

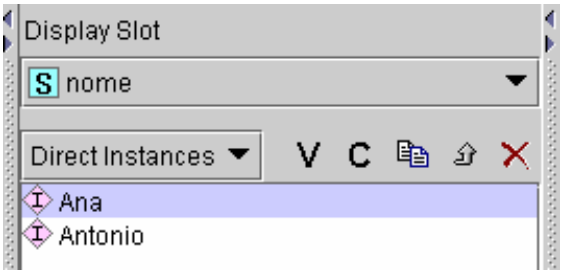


Figura 3.20 Instâncias da superclasse Homem

Uma vez representada toda a base de conhecimento através de ontologia no software Protege [21], a seguir na Figura 3.21, através de gráficos de relacionamentos entre as classes do domínio, demonstra-se como se dá o contágio e a sintomatologia.

Como já foi comentado no início desta modelagem, as três entidades principais para esta demonstração são: o parasita, o homem, o mosquito e o sintoma. A partir destes elementos tem-se duas relações: (a) uma de contágio, onde se verifica o vetor de transmissão da doença e; (b) outra de sintomatologia, onde se explora as características para o diagnóstico da doença.

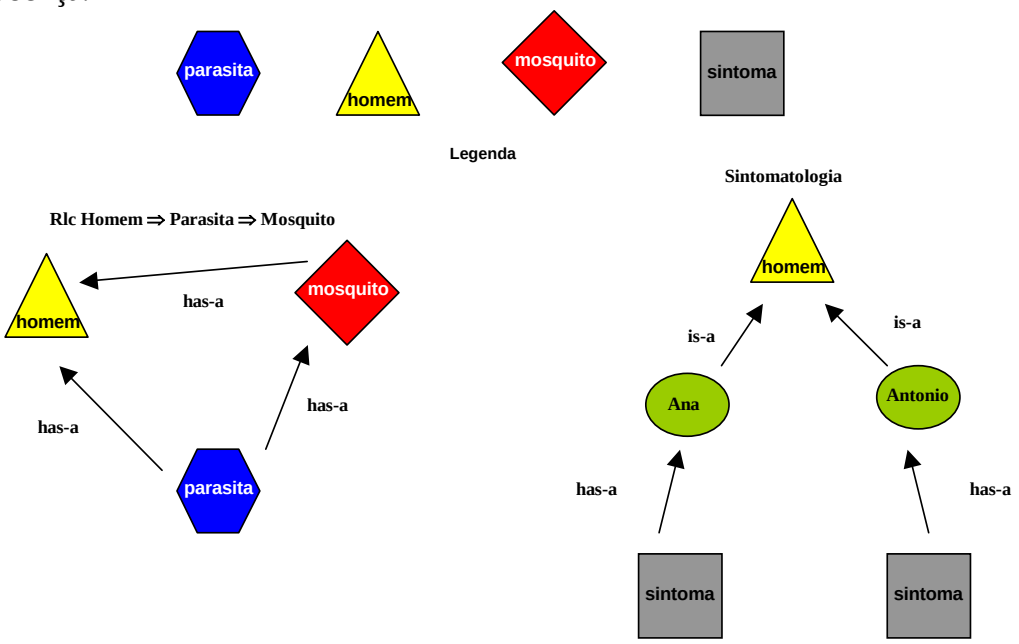


Figura 3.21 Relação semântica entre as entidades do domínio

A partir dos dados coletados acima seguem as regras de produção do sistema escritas em pseudocódigo e em seguida usando a linguagem JESS[31]:

Regra 1: SE o paciente tem febre e a febre é regular e varia entre 38 e 40 graus ENTÃO verificar se o paciente esteve em área endêmica.
Regra 2: SE o paciente esteve em área endêmica e a mais de 30 dias ENTÃO verificar o contato com pessoas infectadas.

Regra 3: SE o paciente teve contato com pessoas infectadas a menos de 3 anos ENTÃO verificar sintomas.

Regra 4: SE o paciente apresenta os sintomas (dor no corpo, cefaléia, icterícia...) ENTÃO submete-lo ao exame de gota espessa do sangue periférico.

Regra 5: SE o exame for positivo ENTÃO o paciente é portador do parasita da malária.

```
(deftemplate pergunta
    (slot texto)
    (slot tipo)
    (multislot valido)
    (slot ident))

(deftemplate resposta
    (slot ident)
    (slot texto))

(do-backward-chaining resposta)

(defrule MAIN::faixa-etaria
    (declare (auto-focus TRUE))
    (explicit (resposta (ident etaria) (texto crianca)))
     =>
    (acao-recomendada "Use o Diagnostico em Criancas")
    (halt))

(defrule MAIN::malaria
    (declare (auto-focus TRUE))
    (resposta (ident S) (texto sim))
    (resposta (ident S1) (texto sim))
    (resposta (ident S2) (texto sim))
    (resposta (ident S3) (texto sim))
    (resposta (ident S4) (texto sim))
    =>
    (acao-recomendada "malaria")
    (halt))

(deffacts MAIN::perguntas
    (pergunta
      (ident etaria)(tipo multi) (valido adulto crianca)
```

[31] JESS Java Expert System Shell é um pacote escrito em java para desenvolvimento de sistemas especialistas. Tal pacote possui uma linguagem própria para construção de bases de regras.

```
      (texto "Qual e a sua faixa etaria?"))
    (pergunta
      (ident S) (tipo multi) (valido sim nao)
      (texto "O paciente tem febre?"))
    (pergunta
      (ident S1) (tipo multi) (valido sim nao)
      (texto "A febre varia entre 38 e 40 graus?"))
    (pergunta
      (ident S2) (tipo multi) (valido sim nao)
      (texto "O paciente esteve em área endêmica a menos de 30
dias?"))
    (pergunta
      (ident S3) (tipo multi) (valido sim nao)
      (texto "Teve contato com infectados a menos de 3
anos?"))
    (pergunta
      (ident S4) (tipo multi) (valido sim nao)
      (texto "Sente dor no corpo, cefaléia, icterícia,
etc.?"))

   (MAIN::ask etaria) )

(deffunction acao-recomendada (?acao)
              (printout t "O diagnostico provavel e: " ?acao
crlf))

(reset)
        (run)
```

Todo o processo de modelagem do conhecimento para o sistema especialista para telediagnóstico no ambiente de serviços *web*, assim como a abordagem dos procedimentos para a modelagem de um KBS estão descritas neste capítulo.

O próximo passo é modelagem do sistema especilista a partir do conhecimento do domínio da malária - diagnóstico. Este assunto será abordado no capítulo seguinte.

Capítulo 4 O Sistema Especialista

Neste capítulo será visto:

- A modelagem e o desenvolvimento do sistema MEDIC;
- A modelagem e o desenvolvimento do sistema WST;
- A integração dos dois sistemas (MEDIC & WST).

4 O SISTEMA ESPECIALISTA

O avanço natural das pesquisas que buscam investigar os processos e as relações entre os domínios de conhecimento deu origem à quebra de paradigma na lógica “de ver o mundo e expressá-lo”.

A IA sempre teve como um dos seus objetivos primordiais a compreensão e o detalhamento do conhecimento, com vista, ao desenho do comportamento humano, uma das respostas para estes problemas é o sistema especialista.

Este capítulo aborda o processo de construção do sistema especialista proposto. De acordo com o que já foi comentado anteriormente, trata-se de uma solução que irá efetuar um diagnóstico acerca da malária a partir de uma anamnese.

O sistema como um todo terá três fases, como se pode ver no diagrama da Figura 4.1, já detalhadas no capítulo 1:

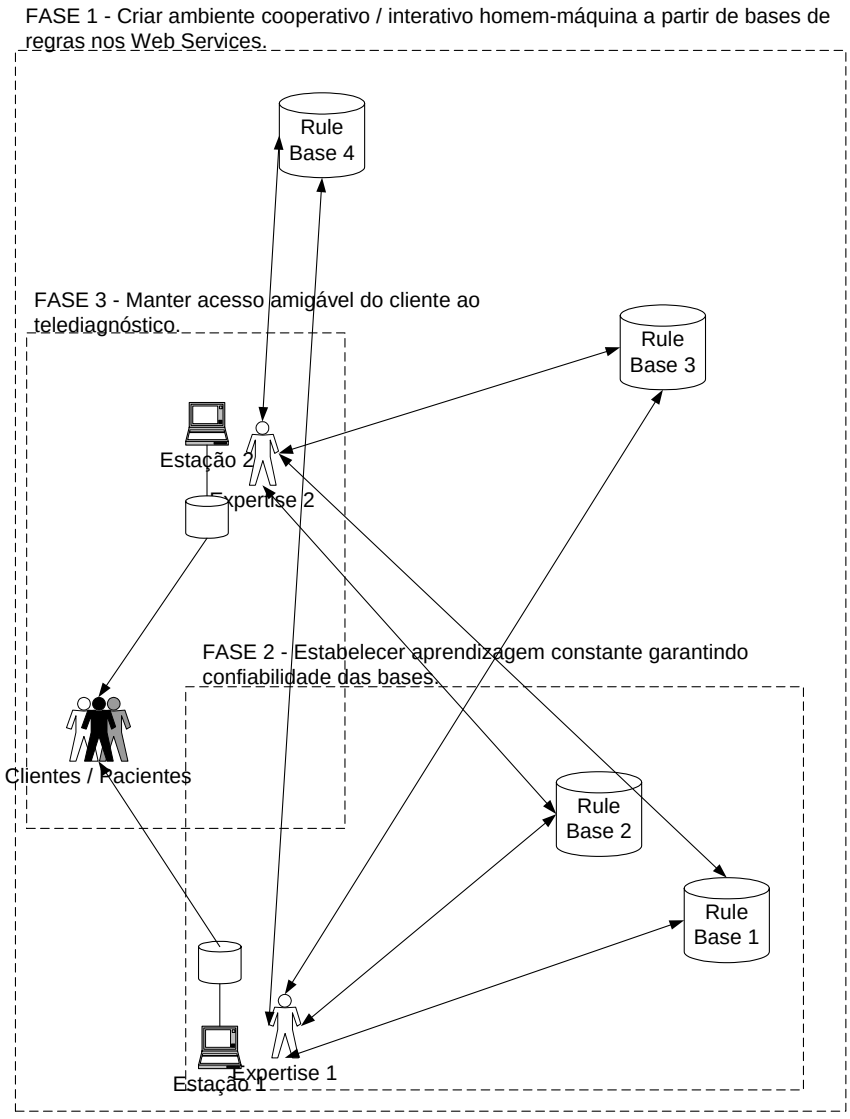


Figura 4.1 Diagrama geral do *WST Web Services Telediagnosis*

A primeira fase: criar um ambiente cooperativo / interativo homem-máquina a partir de bases de regras nos *Web Services* será desenvolvida neste trabalho, ficando as fases 2 e 3 para trabalhos futuros.

Apesar da modularização é importante que se comente o projeto como um todo para se situar na proposta de desenvolvimento.

A primeira fase parte do pressuposto da necessidade de uma solução computacional para o problema e sugere um sistema especialista com característica colaborativa (uso e compartilhamento da anamnese por vários usuários) e interativa (disponibilização no ambiente de serviços da internet).

A segunda fase (Estabelecer aprendizagem constante garantindo confiabilidade das bases), visto que, o sistema tem a característica colaborativa, trata da atualização das bases de regras e da modelagem de novos domínios acerca da malária.

A terceira fase. Fase final do projeto. Tem como principal objetivo permitir que o sistema seja operacionalizado por usuários com os mais diversos perfis. A idéia inicial é que se crie uma aplicação de interface com este usuário, que utilize o processamento de linguagem natural para criação de fatos que irão alimentar as bases de regras do sistema.

4.1. Fase 1 Criar um ambiente cooperativo / interativo homem-máquina a partir de bases de regras nos Web Services

As seguintes etapas (desenvolvidas no processo de modelagem do sistema especialista no capítulo 3) serão pontuadas até que se chegue ao início da fase de desenvolvimento do artefato que irá contemplar este primeiro módulo:

- Identificação do domínio;
- Identificação dos *expertises* deste referido domínio;
- Definição do escopo das bases de regra do domínio definido;
- Modelagem e validação desta base.

Em um primeiro momento estas etapas foram descritas no capítulo 3 modelagem do conhecimento.

Em seguida as etapas que serão abordadas neste capítulo:

- Modelagem das aplicações MEDIC (sistema legado) e *WST Web Services Telediagnosis* (serviço *web*);
- Integração da aplicação MEDIC (sistema legado) à aplicação *WST Web Services Telediagnosis* (serviço *web*).

Como visto acima, as etapas de modelagem das aplicações e a integração da aplicação *web* a um sistema legado, será o escopo do desenvolvimento da primeira fase do sistema.

Algumas técnicas, metodologias e ferramentas serão usadas com esta intenção – modelagem e implementação do artefato. A idéia é que todo o sistema seja desenvolvido usando ferramentas *freeware*. No entanto, visando simular a flexibilidade do projeto, será usado para o processo de modelagem do sistema MEDIC ferramentas proprietárias.

Para o desenvolvimento da aplicação MEDIC serão usadas as seguintes ferramentas:

- Visio 2000 (para modelagem do banco de dados e do sistema);
- Power Design (para modelagem do banco);
- SQL Server (para implementação do banco de dados);
- Visual Basic 6 (para implementação da aplicação *desk*).

Para o desenvolvimento da aplicação *WST Web Services Telediagnosis* – que será integrada a aplicação MEDIC, serão usadas as seguintes ferramentas freeware:

- Poseidon (para modelagem da aplicação);
- Java / JESS (para implementação da aplicação);
- Firebird (para implementação do banco de dados).

4.1.1. Sistema MEDIC

Acredita-se que a maioria das organizações que tratam a malária hoje no mundo (i.e. FUNASA, FUNAI, universidades públicas) tem algum tipo de artefato de software que processam os dados clínicos acerca desta doença.

Tal artefato via de regra mantém em uma base de dados, o cadastro dos pacientes, suas anamenses, assim como, outros procedimentos médicos.

A seguir a documentação parcial de um artefato de software que contempla as aplicações necessárias (cadastro de paciente, marcação de consulta, anamnese, etc.) para que se ilustre esta etapa do projeto.

O software MEDIC desenvolvido utilizando a linguagem Visual Basic e o banco de dados SQL Server é um artefato estruturado, desta forma a sua documentação seguirá a metodologia de desenvolvimento segundo Ed. Yourdon [29].

Por questões de escopo do trabalho, serão abordados nesta documentação somente os módulos indispensáveis ao desenvolvimento (demonstração) do sistema especialista.

A documentação do sistema MEDIC se inicia com o ciclo de vida da Figura 4.2. Este em particular é o mais simples possível, visto que, de acordo com a organização que eventualmente desenvolve sistemas, adaptações são feitas caso a caso.

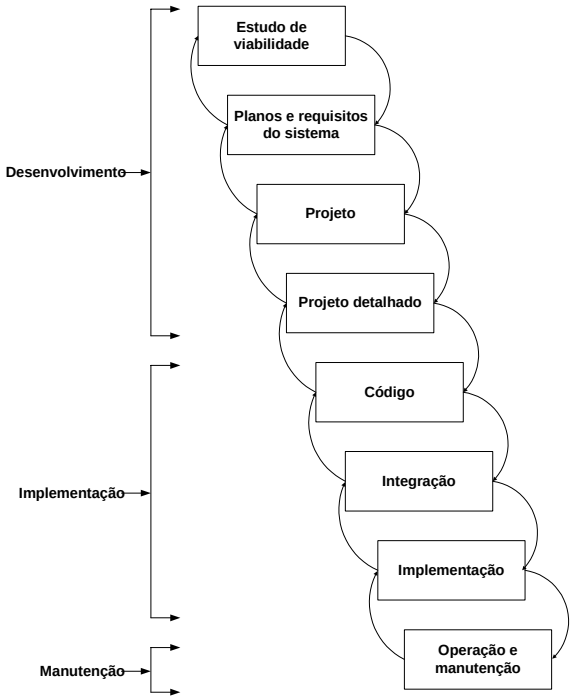


Figura 4.2 Ciclo de vida Sistema MEDIC

Existem três momentos essenciais neste ciclo de vida que podem ser agrupados:

- O desenvolvimento – que envolve desde a solicitação, passando pelo estudo de viabilidade do desenvolvimento até o desenho de uma solução;
- A implementação – envolve a codificação, os testes e a aceitação pelo usuário;
- A manutenção – as etapas planejadas para possíveis correções ou aperfeiçoamentos (não será abordada, visto que, foge ao escopo da pesquisa). Uma vez que o trabalho é o desenvolvimento de um sistema especialista.

A elaboração do sistema MEDIC será iniciada pelo módulo que agrupa as principais tarefas de desenvolvimento.

O sistema MEDIC será um artefato que contemplará todo processo ambulatorial de um paciente. Este sistema fará a anamnese local e poderá fazê-la usando o sistema especialista que estará disponível como serviço *web*, por essa razão este deve estar conectado a um *link* com a internet.

A seguir seguem algumas funcionalidades do sistema MEDIC e o diagrama de contexto deste na Figura 4.3:

1. Controlar o cadastro de pacientes;
2. Gerenciar a marcação de consultas;
3. Fazer anamnese clínica;
4. Emitir receituário e atestado médico;
5. Controlar exames laboratoriais.

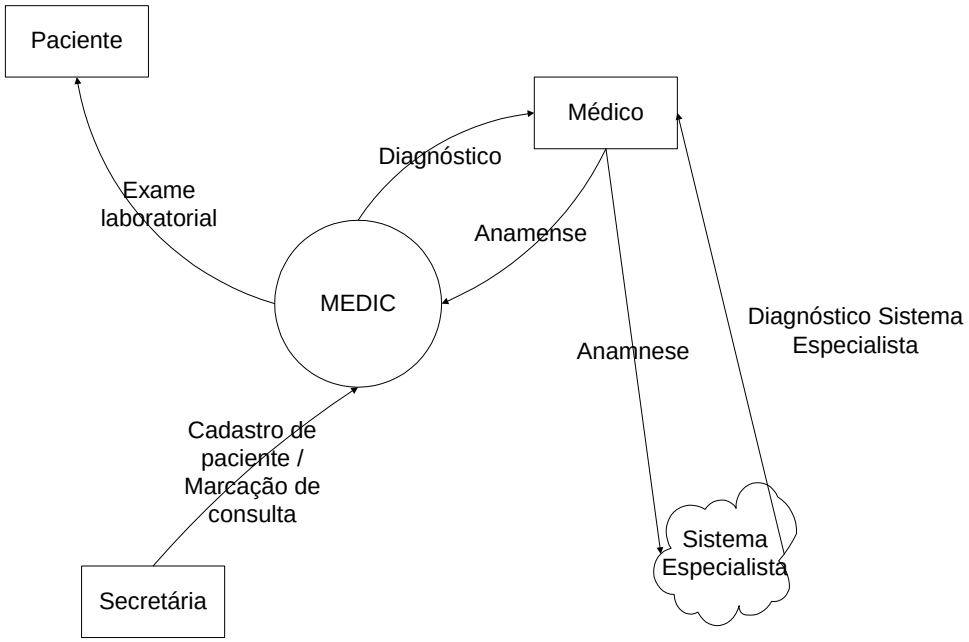


Figura 4.3 Diagrama de contexto do Sistema MEDIC

Baseado no objetivo do sistema MEDIC e nos seus requisitos operacionais, que são controlar o processo ambulatorial de um paciente – anamnese e conseqüente diagnóstico, segue na Figura 4.4 o diagrama de fluxo de dados nível 0, que demonstra as principais funcionalidades: cadastro, marcação e anamnese (local e via sistema especialista).

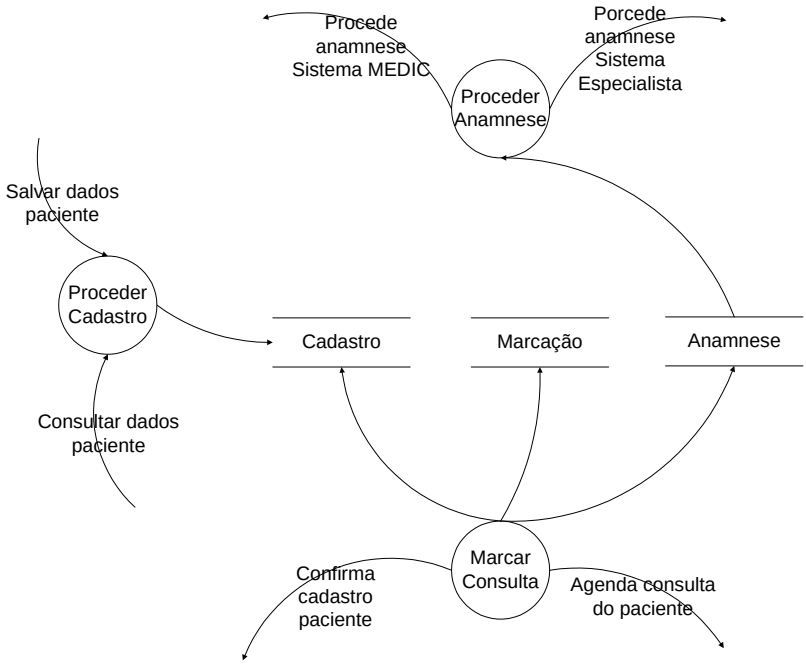


Figura 4.4 Diagrama de fluxo de dados nível 0 Sistema MEDIC

De acordo com o comentado acima sobre o sistema MEDIC e baseado no diagrama de contexto, apenas os módulos de cadastro de paciente, marcação de consulta e anamnese foram abordados neste documento.

A intenção é que se perceba o processo de atendimento de um paciente ambulatorial.

Segue os passos em uma seqüência de ações, desde a entrada dos dados cadastrais do paciente até o processo de submissão das informações da anamnese ao mecanismo de inferência do sistema especialista, no ambiente de serviços *web*.

1. O paciente reporta os dados pessoais ao agente de saúde que, por sua vez, o insere em um formulário do sistema;
2. Após inseri-los, o agente de saúde procede a marcação da consulta selecionando em um outro formulário o paciente cadastrado e o médico responsável pelo seu atendimento;
3. Por sua vez, o médico, dá ocasião do atendimento, seleciona o paciente no formulário de anamnese e preenche os dados de anamense, procedimentos médicos e CID[32].

[32] O CID é o Código Internacional de Doenças, taxonomia obrigatória no atendimento médico.

4. Finalmente o médico ou agente de saúde enviam estes dados a partir de um controle (i.e. botão) neste mesmo formulário, através de uma mensagem XML[33] ao servidor que esta publicado o serviço *web*, que por sua vez acionará o mecanismo de inferência do sistema especialista.
5. Após este procedimento o médico ou agente de saúde terão como resposta do sistema especialista uma mensagem analisada diagnosticando com base no conjunto de conhecimentos a patologia da malária no paciente.

Deste ponto em diante será demonstrada a modelagem do banco de dados que suportara as informações do sistema. O modelo relacional foi construído usando a ferramenta CASE[34] Power Design e em destaque na cor verde é chamada atenção para as tabelas que são manipuladas pelas aplicações abordadas acima, como demonstra a Figura 4.5. São elas:

a. CC01PACIET – dados cadastrais do paciente;
b. CC04MEDICT – dados cadastrais do médico ou agente de saúde;
c. CC05AGENDT – agenda de marcação de consultas;
d. CC07CONVET – convênios médicos usados na marcação de consultas;
e. CC06CONSUT – tabela que armazena as anamneses;
f. CC02CIDT – tabela para cadastro do CID Código Internacional de Doenças;
g. CC03EXAMET – cadastro dos exames médicos (tabela básica);
h. CC15EXARET – exames requisitados ao paciente;
i. CC14EXAPAT – histórico de exames do paciente.

[33] XML eXtended Markup Language é uma linguagem de marcação de texto para o ambiente da internet.
[34] CASE Computer Aided Software Engeneering. Ferramenta de produtividade para modelagem e geração de banco de dados.

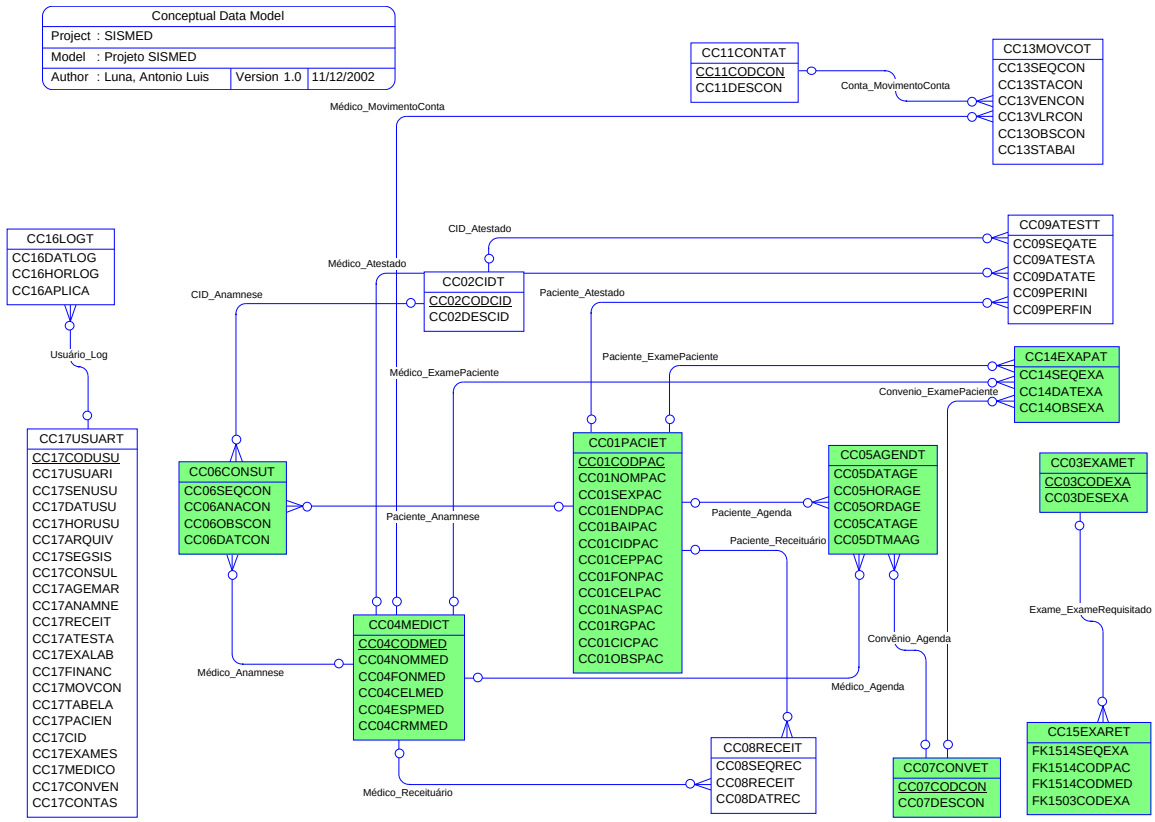


Figura 4.5 Modelo conceitual do banco de dados do sistema MEDIC

Procede-se assim a próxima etapa na construção deste sistema – MEDIC - que será a codificação das aplicações propostas assim como a integração do mesmo ao ambiente da internet pela aplicação de anamnese.

O sistema é iniciado através de um *login* por uma tela de segurança (Figura 4.6) onde o usuário, previamente cadastrado informa seu nome de usuário e uma senha numérica de 6 dígitos.

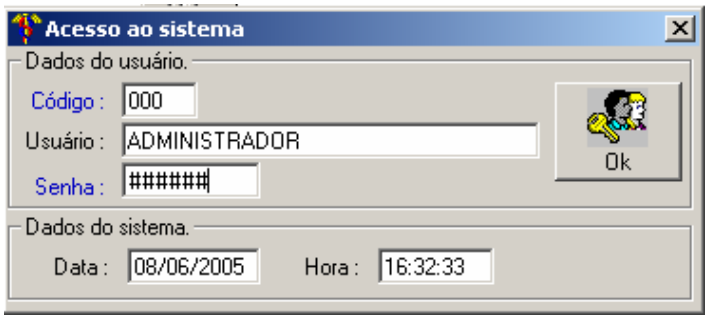


Figura 4.6 Formulário para *login* no sistema MEDIC

As informações são verificadas nas tabelas de segurança do sistema e o usuário recebe permissão para acesso aos módulos (aplicações do sistema). Em seguida um

formulário principal do tipo MDI[35] (Figura 4.7) contendo controles e menus de acesso às aplicações será exibido para que se proceda o processo de operacionalização do sistema.

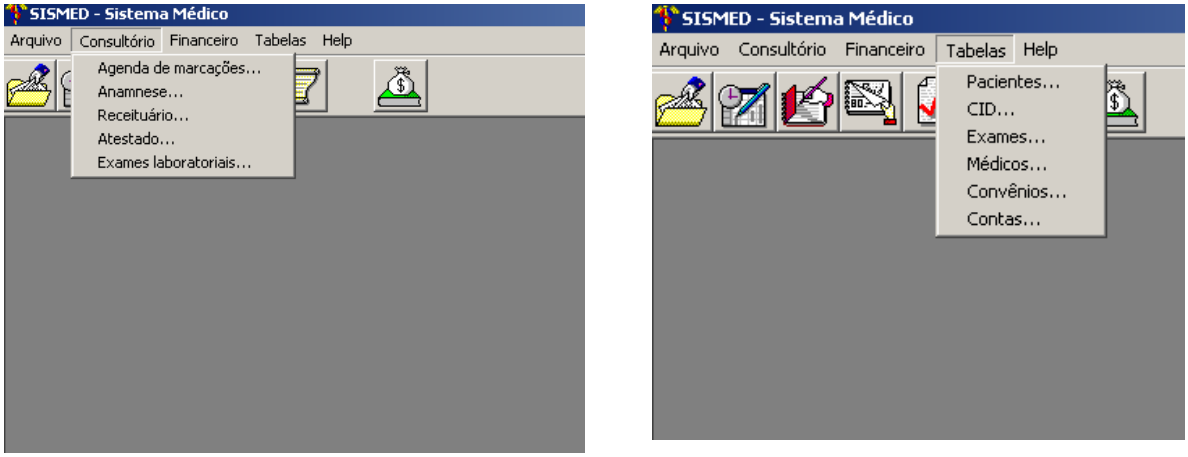


Figura 4.7 Formulário principal do sistema com os menus de acesso às aplicações do sistema MEDIC

Todo o sistema segue um padrão de implementação do código, desta forma, os eventos básicos contidos nas aplicações – como novo registro, salvar, atualizar, excluir, tem a mesma estrutura assim como existem os mesmos controles em todos os formulários.

O primeiro procedimento será o cadastro dos médicos ou agentes de saúde e dos pacientes assim como das demais tabelas básicas do sistema que irão dar suporte ao processo de anamnese.

Para cada registro uma chave numérica de três posições será gerada pelo próprio sistema visando que se evite desta forma a duplicidade deste registro. Na Figura 4.8, o formulário de cadastro dos médicos ou agentes de saúde.

[35] MDI Multiply Document Interface – modelo de formulário eletrônico que agrega formulários “filhos” dentro dele.

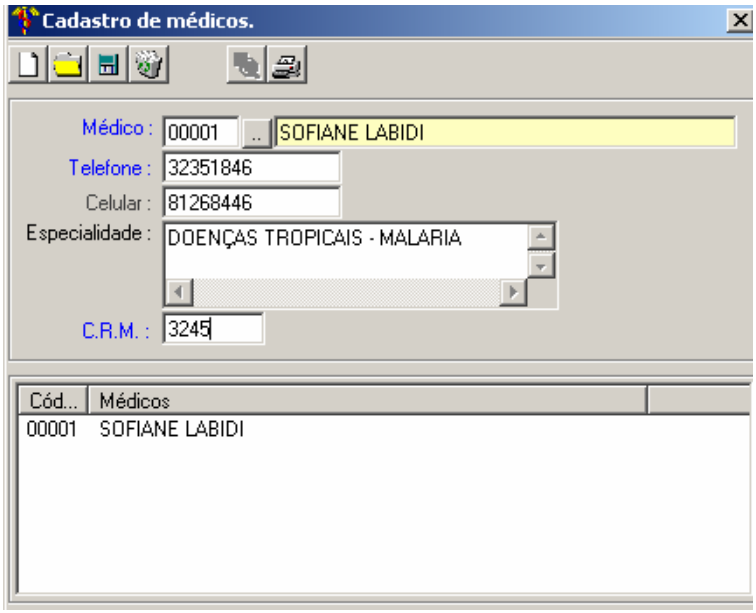


Figura 4.8 Formulário para cadastro de médicos ou agentes de saúde no sistema MEDIC

Para que se possa visualizar a padronização de código comentada, segue no Apêndice G um exemplo para o evento salvar os registro neste formulário.

Cadastrado o médico e partindo-se do pressuposto que o usuário secretária já foi cadastrado no sistema e recebeu as suas devidas permissões operacionais, a próxima etapa será o cadastramento do paciente por este usuário. O mesmo procedimento para cadastramento do médico é executado agora para o paciente em seu formulário próprio na Figura 4.9.

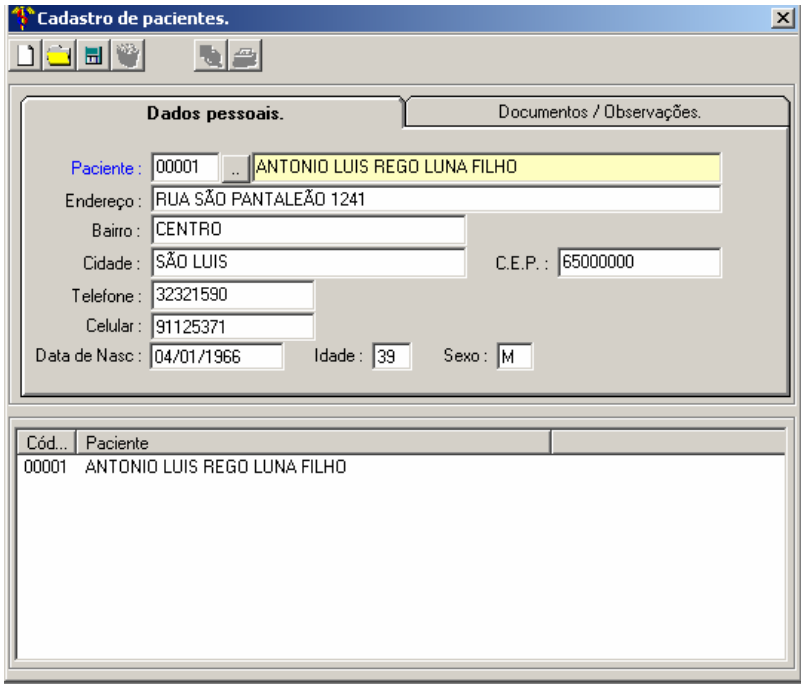


Figura 4.9 Formulário para cadastro de pacientes no sistema MEDIC

Após a inserção dos dados do paciente se procede ao agendamento (Figura 4.10) do atendimento. Este agendamento deve fornecer primordialmente, além das informações de praxe que pede o formulário, o médico, ou seja, a chave deste.

A chave deste, em associação à chave do paciente formarão a chave de acesso às informações do agendamento.

Na realidade, médico e paciente neste sistema – chaves – sempre formarão uma associação para acesso a qualquer informação clínica.

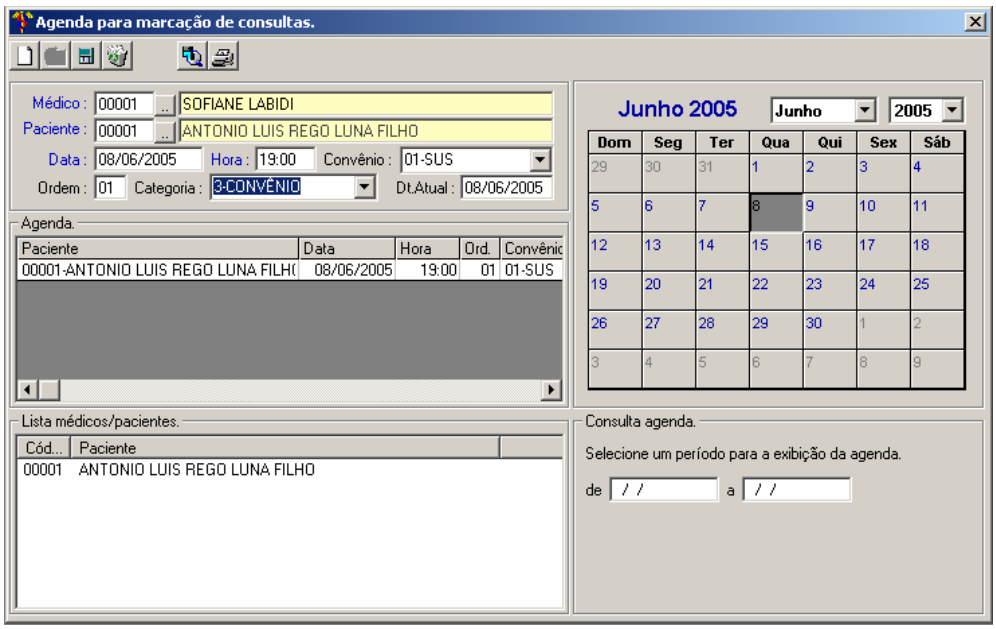


Figura 4.10 Formulário para agendamento de pacientes no sistema MEDIC

Em seguida ao agendamento, e como última etapa no processo de atendimento usando o sistema MEDIC para o diagnóstico do paciente (local) o formulário de anamnese (Figura 4.11) será preenchido pelo médico com os dados que serão reportados pelo paciente acerca da sintomatologia e dos procedimentos/condutas clínicas.

A partir deste formulário, através de um controle (i.e., botão), duas opções estarão à disposição do médico para que este obtenha uma segunda opinião acerca do que foi constatado em seu contato local com as observações do paciente: (1) o envio do conteúdo do campo anamnese através de uma mensagem XML ao servidor de serviço *web*; (2) o preenchimento de um formulário HTML[36] que será enviado da mesma forma ao servidor de serviço *web* que contém o mecanismo de inferência.

[36] HTML Hiper Text Markup Language – Linguagem de marcação de texto com a qual se constrói páginas para a internet.

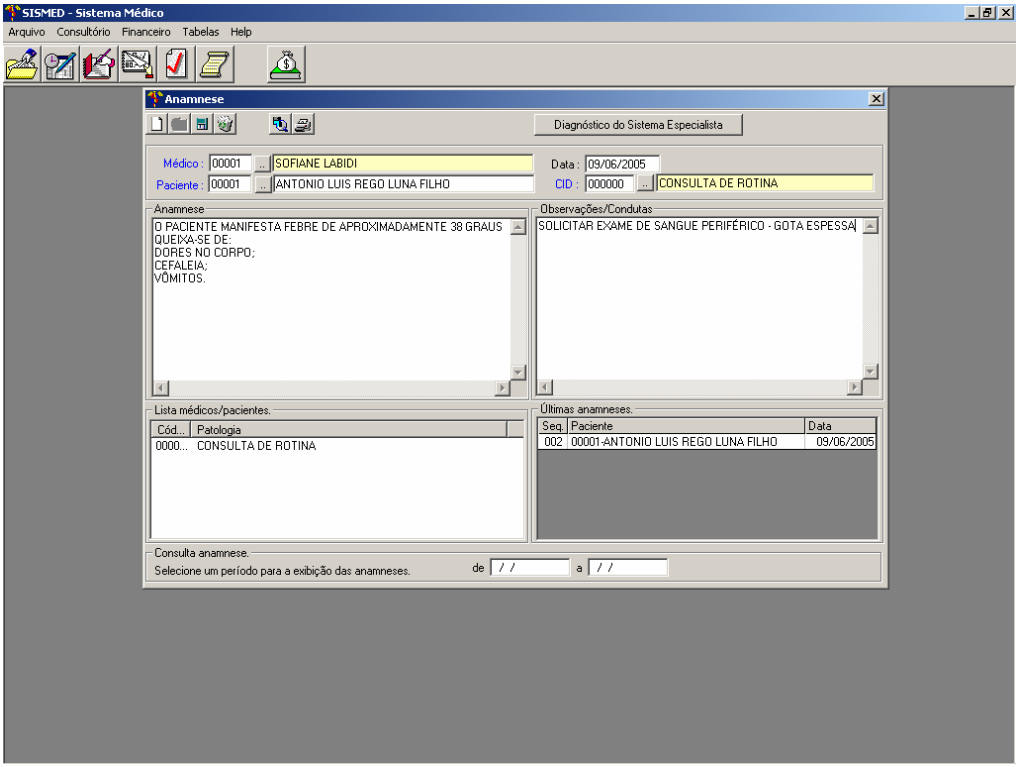


Figura 4.11 Formulário para anamnese e acesso ao sistema especialista no sistema MEDIC

Um browser desenvolvido usando a linguagem Visual Basic (mesma linguagem da aplicação local) foi incorporado à aplicação visando facilitar o acesso do usuário (médico ou agente de saúde) ao site que disponibiliza o serviço de telediagnóstico. Neste browse, caso o usuário deseje acessar outros serviços disponíveis na internet um campo **endereço** pode ser preenchido com o endereço eletrônico a ser acessado como demonstra a Figura 4.12.

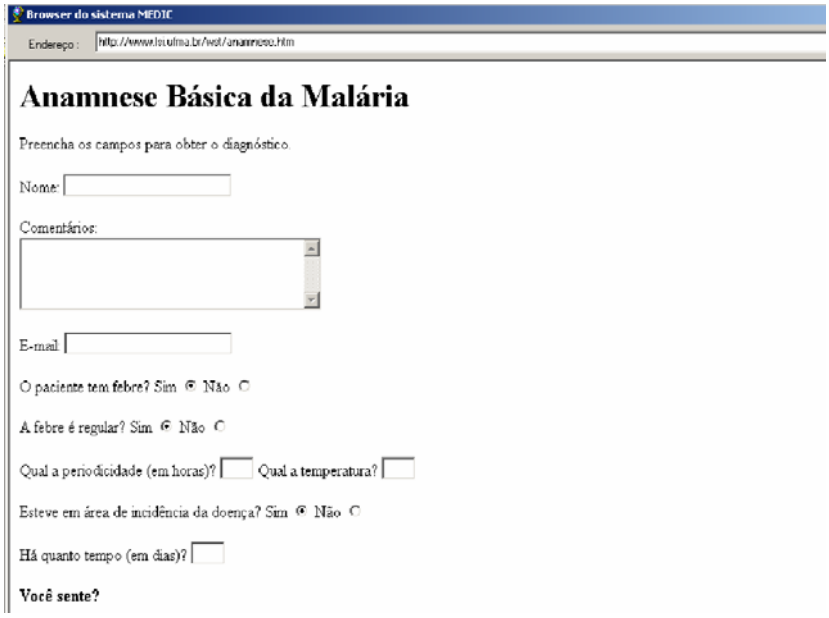


Figura 4.12 Formulário para o browser de acesso ao sistema especialista

A seguir o código principal deste formulário que disponibiliza o site de anamnese para o serviço *web*.

```
On Error Resume Next

 Me.WindowState = 0

 Me.endereco = "http://www.lsi.ufma.br/wst/anamnese.htm"

 Combo1.Text = endereco
 Combo1.AddItem (Combo1.Text)
  WebBrowser1.Navigate endereço
```

4.1.2 Sistema WST- Web Services Telediagnosis

O que esta sendo nomeado como sistema *WST Web Services Telediagnosis* na realidade é uma página HTML com um formulário que irá permitir que o usuário proceda ao telediagnóstico. No entanto a pretensão é que se desenvolva um sistema completo para que este possa ser disponibilizado da mesma forma através de serviços *web*.

Deste ponto em diante faremos a análise e documentação deste artefato baseado essencialmente na funcionalidade deste formulário, visto ele ser o cerne do sistema de telediagnóstico.

Como foi comentado acima, algumas metodologias e ferramentas de análise e documentação serão usadas.

No projeto do sistema MEDIC a metodologia de análise estruturada foi aplicada visando tornar o mais real possível a proposta de desenvolvimento, visto que, a maioria dos sistemas legados, que operam nas organizações seguem este padrão.

No *WST - Web Service Telediagnosis*, sistema para o ambiente *web*, será adotado o paradigma de orientação a objeto. A UML[37] será abordada em toda especificação do projeto e as ferramentas usadas serão totalmente *freeware*, inclusive o ciclo de vida para o desenvolvimento deste sistema, segue as mesmas etapas do sistema anterior, alterando-se apenas a forma de condução destas, onde no sistema MEDIC todas elas foram concluída até a construção do artefato, e neste será usada a abordagem incremental, onde cada incremento percorre todo o ciclo de etapas até que se alcançe a completa elaboração do artefato. O diagrama da Figura 4.13 ilustra esta abordagem.

[37] UML Unified Modeling Language é uma linguagem para modelagem e documentação de projetos de software.

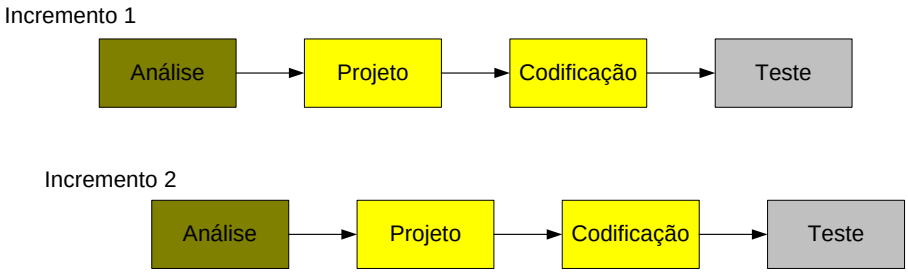


Figura 4.13 Ciclo de vida em abordagem incremental para o WST

Solicitação do sistema: Existe a necessidade pelos médicos e agentes de saúde de uma determinada organização que presta serviços assistências à malária nas regiões tropicais do país de uma segunda opinião acerca do diagnóstico desta doença.

Sabe-se que inúmeros fatores externos podem contribuir com o resultado positivo ou negativo deste diagnóstico o que recomenda o desenvolvimento de um sistema especialista.

Um cadastro do paciente e coleta dos dados da sintomatologia deste, assim como, o armazenamento de um histórico evolutivo desta sintomatologia em um banco de dados irá permitir que se acompanhe a doença de maneira mais acurada e tome providencias eficazes.

Para a especificação do problema acima – ainda que prototipadamente - dois conjuntos de diagramas serão usados pela UML neste projeto como mostra a Figura 4.14:

- **os diagramas estruturais** aqueles que modelam os aspectos estáticos do software e suas relações (diagramas de classes, objetos, componentes);
- **e os diagramas comportamentais** que são aqueles que levam em conta as características dinâmicas do software como os diagramas de caso de uso, seqüência, colaboração, etc.

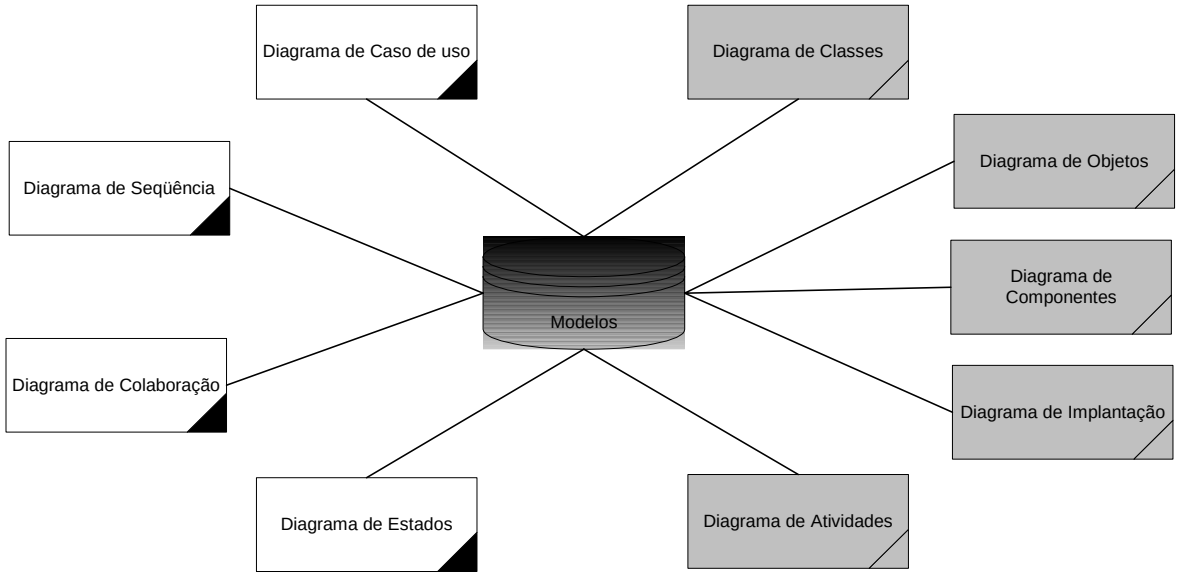


Figura 4.14 Diagramas da UML Unified Modeling Language

Pode-se identificar na solicitação do sistema alguns requisitos funcionais que irão também dar origem a casos de uso: (a) cadastrar de clientes; (b) coletar a sintomatologia reportada na consulta realizada no sistema MEDIC; (c) elaborar um histórico evolutivo da doença por paciente.

Far-se-á a especificação da coleta da sintomatologia de acordo com o modelo abordado pela UML, visto que, a abordagem é essencialmente de um telediagnóstico com base na anamnese do paciente. Os outros requisitos serão abordados oportunamente quando da ocasião do desenvolvimento do sistema como um todo.

O próximo passo é a definição dos atores e dos casos de uso, que para este projeto será apenas o caso de uso referente a anamnese do paciente. O procedimento para que se identifique estes elementos da especificação é basicamente responder alguns questionamentos, como segue:

1. Que usuários utilizam o sistema para cada tarefa?
2. Que usuários são necessários e indispensáveis para que a mesma tarefa seja realizada?
3. Que sistemas outros utilizam ou utilizaram este sistema para realização desta tarefa?
4. Quais são os sistemas outros que dependem deste para realização das suas tarefas?
5. Que grupos de usuários outros dependem deste sistema para realização desta tarefa ou das tarefas de seus sistemas?

Estes questionamentos identificam os atores e indicam os casos de uso associados a eles com base nas funcionalidades da referida tarefa.

Com base nas respostas ao que foi questionado acima se identifica para o escopo do projeto o médico e o agente de saúde como atores e como caso de uso a ser especificado tem-se o cadastro da sintomatologia. Desta forma tem-se a Tabela 4.1 a seguir para este caso de uso e o seu respectivo diagrama na Figura 4.15.

Caso de uso	Cadastrar Anamnese	
Descrição	O caso de uso permite que o ator submeta a anamnese do paciente ao sistema especialista	
Ator	Médico ou Agente de Saúde	
Interação entre ator e sistema	**Médico ou Agente de Saúde**	**Telediagnóstico**
	O caso de uso é iniciado quando o médico ou agente de saúde pressiona no formulário de anamnese do sistema *desk* MEDIC o botão que irá permitir o acesso ao serviço *web* de telediagnóstico.	
		Um browser a partir do sistema *desk* MEDIC acessa um endereço do telediagnóstico previamente passado a um método da aplicação *desk* .
	O médico ou agente de saúde preenche o formulário *web* contendo as informações acerca da sintomatologia do paciente e submete este formulário através do botão enviar.	
		O sistema envia esta requisição que é submetida a uma base de regra localizada em um servidor de serviço *web*.
		As regras são aplicadas aos fatos enviados e o sistema retorna uma mensagem ao usuário confirmando ou negando o diagnóstico.
Exceções	**Ex 01**	Todos os campos do formulário devem ser preenchidos, caso contrário, o mecanismo de inferência não irá levar em consideração alguns fatos que podem ser fundamentais para um diagnóstico acurado.

Tabela 4.1 Caso de uso cadastro da anamnese

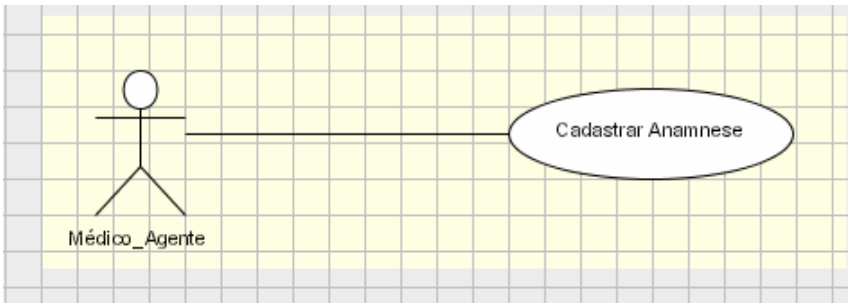


Figura 4.15 Diagrama de caso de uso

Com base no caso de uso cadastrar anamnese abaixo a Tabela 4.2 que especifica esta classe e seu o diagrama de classe na Figura 4.16.

Classes	Atributos	Operações
Anamnese	Nome	Submeter_anamense
	Comentário	
	Email	
	Febre	
	Febre_reg	
	Periodicidade	
	Área_incid	
	Tempo	
	Calafrios	
	Sudorese	
	Vômitos	

Tabela 4.2 Diagrama de caso de uso

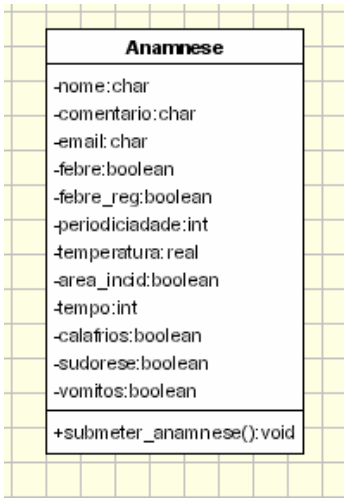


Figura 4.16 Diagrama de classe

Em seguida o diagrama de seqüência (Figura 4.17) que ilustra a interação entre o usuário médico ou agente de saúde e o sistema objetivando o telediagnóstico, e o código Java gerado pelo software Poseidon após a definição destas especificações. O código é só um exemplo da eficácia desta ferramenta, não representando desta forma a codificação final das classes envolvidas, visto que, o pacote JESS será implementado na estrutura definida.

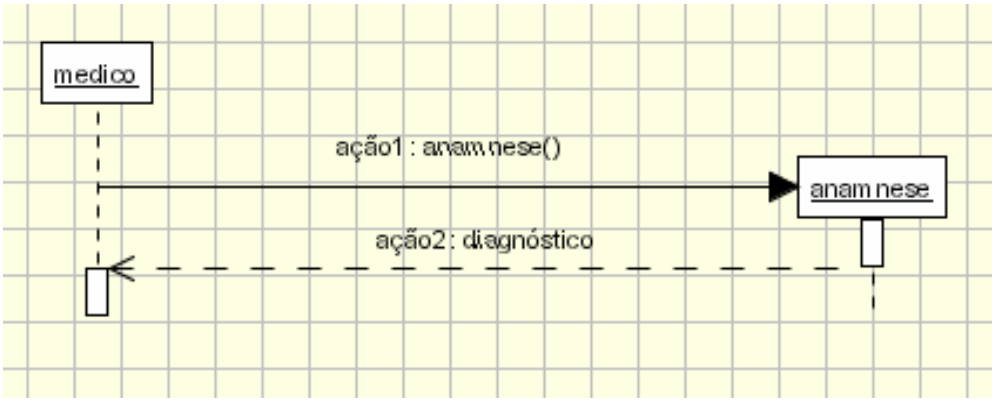


Figura 4.17 Diagrama de seqüência para a operação de telediagnóstico

Finalmente, de acordo com o que foi modelado, se procede à implementação do código que servirá de protótipo para a proposição do projeto, que é um telediagnóstico.

As aplicações para o ambiente da internet (comércio eletrônico, entretenimento, segurança, etc.) tem se tornado uma tendência. As aplicações para este ambiente com suporte de inteligência artificial – sistemas especialistas – por sua vez tem se tornado um desafio.

Existem algumas maneiras de se implementar o exposto acima no ambiente da internet, particularmente usando o JESS como mostra a arquitetura da solução na Figura 4.18. Isso pode ser conseguido do lado do cliente através da execução de *applets* Java, ou executando a partir de uma solicitação deste cliente um objeto do lado do servidor via a tecnologia dos Servlets ou JSP[38].

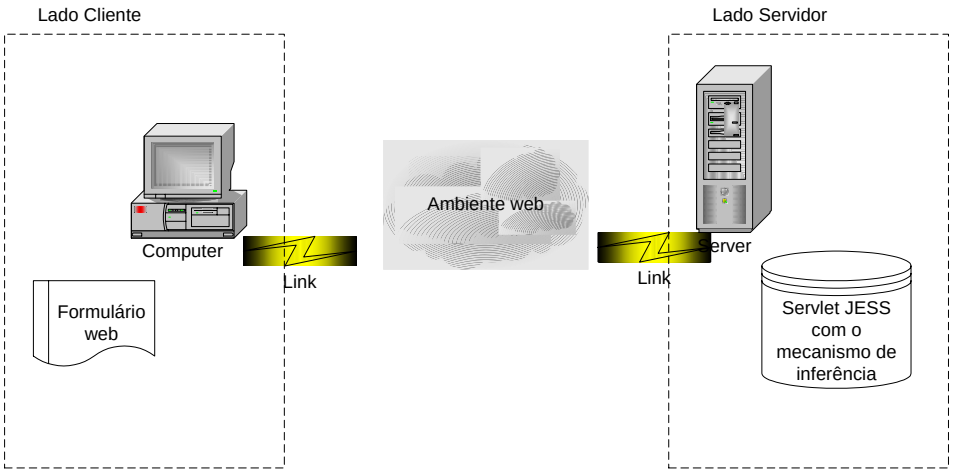


Figura 4.18 Arquitetura para funcionamento do mecanismo de inferência na *web*

[38] JSP Java Server Pages é uma tecnologia para manipulação de conteúdo na *web* na arquitetura cliente-servidor. Esta tecnologia se assemelha aos Servlets, tornando-os diferentes apenas pela característica de manipulação de conteúdo estático deste último.

É importante entender que, como foi comentado no capítulo 2 Estado da Arte, o mecanismo de inferência do JESS pode ser executado a partir do seu próprio *prompt* jess>, a partir de um arquivo com extensão .clp onde se define a estrutura da base fatos / regras inclusive sendo chamada de uma classe Java ou lançando-se mão de uma linguagem própria do pacote JESS que se "encaixa" à linguagem Java.

Para que seja possível este último caso – JESS / Java – é necessário a utilização da biblioteca do JESS onde será encontrado o objeto – Rete – que contém os métodos a serem instanciados na construção da(s) classe(s).

```
import jess.*;
...
Rete engine = new Rete();
```

O objeto Rete possui dois métodos fundamentais para construção desta(s) classe(s), são eles:

- o *addDefTemplate(DefTemplate)* equivalente no JESS ao *deftemplate* – responsável pela contrução dos slots necessários aos dados para a base regras;
- e o *assertFact(Fact)* equivalente ao *assert fact* no JESS – responsável pela declaração dos fatos.

Segue abaixo parte do código da classe que contém o servlet responsável pelo processamento dos fatos sobre as regras acerca da malária e o código HTML do formulário de anamnese que será disponibilizado como serviço *web*.

```
import jess.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Malaria extends HttpServlet
{
      public void doGet(HttpServletRequest request,
      HttpServletResponse response)throws IOException,
      ServletException
      {
            try
            {
            response.setContentType("text/html");

            PrintWriter out = response.getWriter();

            Rete engine = new Rete();

            Deftemplate p = new Deftemplate("pergunta", "As
            perguntas", engine);
            p.addSlot("texto", Funcall.NIL, "STRING");
            p.addSlot("tipo", Funcall.NIL, "STRING");
            p.addSlot("valido", Funcall.NIL, "STRING");
            p.addSlot("ident", Funcall.NIL, "STRING");
            engine.addDeftemplate(p);

            Deftemplate r = new Deftemplate("resposta", "As
            respostas", engine);
```

```
                    r.addSlot("texto", Funcall.NIL, "STRING");
                    r.addSlot("ident", Funcall.NIL, "STRING");
                    engine.addDeftemplate(r);

                    Fact f = new Fact("pergunta", engine);
                    f.setSlotValue("ident",new Value("S",
                    RU.STRING));
                                .
                                .
                                .
                    engine.assertFact(f);
                    engine.executeCommand("(facts)");

                    String rule = (defrule MAIN::malaria
                        (declare (auto-focus TRUE))
                        (resposta (ident S) (texto sim))
                        (resposta (ident S1) (texto sim))
                        (resposta (ident S2) (texto sim))
                        (resposta (ident S3) (texto sim))
                        (resposta (ident S4) (texto sim))
                        =>
                        (acao-recomendada "malaria")
                        (halt))";
                    engine.executeCommand(rule);
                    engine.executeCommand("(run)");


                    engine.addOutputRouter("page", out);
                    }

                    catch (JessException je)
                    {
                          throw new ServletException(je);
                    }
              }
        }

        private void print(String message, Rete engine)
        throws JessException
        {
              engine.executeCommand("(printout t "O diagnostico
        provavel e: " ?acao crlf))");
        }
```

```
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Sistema Especialista para Telediagnóstico usando a Tecnologia
de Web Services.</title>
   </head>

   <body>
```

```
      <?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

      <h1>Anamnese Básica da Malária</h1>

      <p>Preencha os campos para obter o diagnóstico.</p>

      <form method = "post" action = "/cgi-bin/formmail">

         <p><label>Nome:
               <input name = "name" type = "text" size = "25" />
            </label>
          </p>

         <p><label>Comentários:<br />
               <textarea name = "comments" rows = "4"
                  cols = "36"></textarea>
            </label></p>

         <p><label>E-mail:
               <input name = "email" type = "password"
                  size = "25" /></label></p>

         <p>
            <label>O paciente tem febre?</label>
         <label>Sim
                <input name = "febre" type = "radio"
                  value = "Sim" checked = "checked" /></label>
         <label>Não
                <input name = "febre" type = "radio"
                  value = "Não"  /></label></p>

                        .
                        .
                        .
                        .
         <p>
            <input type = "submit" value =
               "Diagnóstico" />
            <input type = "reset" value = "Limpar" />
         </p>

      </form>

   </body>
         </html>
```

Não basta simplesmente codificar o formulário de envio da solicitação de anamense ao servidor, nem tão pouco criar um *servlet* através de uma classe Java para que a aplicação funcione. É preciso estabelecer todo um ambiente de funcionamento para esta arquitetura.

Primeiro é necessário um servidor *web* instalado – o servidor usado neste projeto é o Tomcat do projeto Jakarta (http://jakarta.apache.org/tomcat/). A versão utilizada é a 4.1.29 que contempla o *servlet* e parte da especificação da JSP.

É importante que se note ao baixar a versão do servidor *web* a especificação para os *servlets* e para a JSP. Cada uma delas tem características no projeto que já foram ou não especificadas. Basta que se escolha devidamente na página acima. Esta informação encontra-se explícita. Após o *download* e a descompactação do arquivo, o usuário terá a seguinte estrutura de pastas como mostra a Figura 4.19:

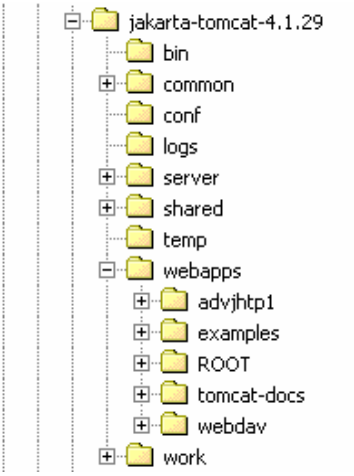
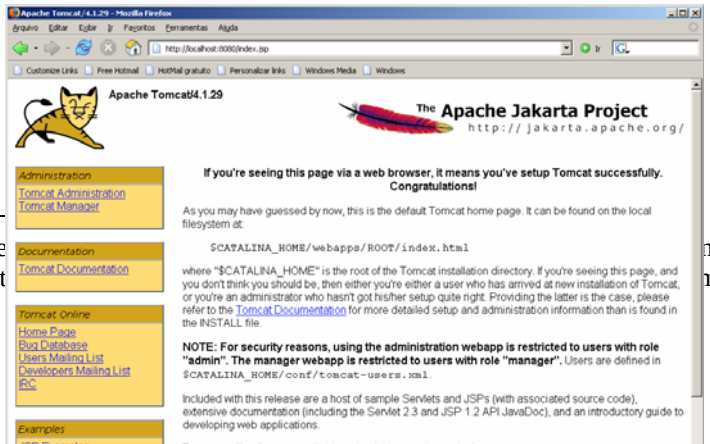


Figura 4.19 Estrutura de pastas para o servidor de página Tomcat

Concluída esta etapa o usuário deverá criar as variáveis de ambiente JAVA_HOME direcionada para a pasta de instalação do JDK[39] e TOMCAT_HOME direcionada para pasta de instalação do servidor Tomcat. Feito isso o servidor poderá ser iniciado a partir de C:\jakarta-tomcat-4.1.29\bin\startup. O servidor é executado a partir da porta lógica TCP8080. `http://localhost:8080`

Esta linha de instrução deverá exibir a página de documentação do Tomcat (Figura 4.20). O *host* do **localhost** indica que o navegador *web* esta pronto para as solicitações de página.



[39] JDK Java Deve nto. Nesta pasta entre outras aplicações encont mento da plataforma Java.

Uma vez instalado, configurado e testado o servidor de página, deve-se apenas inserir a aplicação de telediagnóstico na estrutura de pastas deste servidor.

O servidor Tomcat tem uma estrutura de pastas padrão conforme a descrição abaixo seguindo como sub-pastas da pasta ***web*apps**:

- raiz de contexto – pasta referente ao aplicativo *web*;
- *WEB*-INF – pasta que conterá a descrição do aplicativo *web*. Na verdade o arquivo de descrição *web*.xml;
- *WEB*-INF/classes – conterá os arquivos de classes da aplicação *web*, inclusive os *servlets*;
- *WEB*-INF/lib – conterá os repositorios de arquivos Java (os arquivos .jar).

A estrutura de pastas para a aplicação de telediagnóstico será a seguinte:

```
malaria
   servlets
          anamnese.html
   WEB-INF
          web.xml
          classes
                 servlets
                        Malaria.class
```

Criado a estrutura de pastas que irá abrigar a aplicação deve-se editar o contêiner *servlet* . Através deste *container* **server.xml –** encontrado em C:\jakarta-tomcat-4.1.29\conf, se descreverá a configuração do servidor (uma das configuração é o *context* da aplicação) a raiz de contexto para a aplicação do projeto de telediagnóstico.

```
<Context path = "/malaria"
                 docBase = "webapps/malaria"
                 reloadable = "true">
</Context>
```

Desta feita o usuário deve digitar `http://localhost:8080/malaria/servlets/anamnese.html` e o resultado será o acesso a página

HTML com o formulário da anamnese visto anteriormente no desenvolvimento da integração entre as aplicações MEDIC e *WST Web Services Telediagnosis*.

As etapas percorridas até aqui, descrevem a construção e integração de duas aplicações – sistema legado (MEDIC) e *WST Web Services Telediagnosis* (sistema especialista para telediagnóstico). O próximo passo a ser discorrido será a disponibilização desta aplicação *web* como serviço.

Capítulo 5 O Sistema Especialista no Web Services

Neste capítulo será visto:

- O processo de configuração dos clientes e servidores de serviços web;
- A publicação da aplicação WST como serviço web.

5 O Sistema Especialista no Web Services

O universo das redes de computadores em face de sua evolução tem viabilizado alguns processos inicialmente senão impossíveis, impraticáveis do ponto de vista das funcionalidades requeridas para o sucesso destes. Pode-se ter como exemplo a interoperabilização de aplicações de forma transparente ao usuário.

Das iniciativas em disponibilizar serviços através das antigas BBSs, passando pelos padrões como CORBA até chegarmos às facilidades da internet através dos *Web Services*, a intenção foi sempre aproximar necessidades comuns em um ambiente descentralizado.

Um número crescente de soluções e aplicabilidades para este fim (i.e. comércio eletrônico, sistemas especialistas os mais variados, suporte a informações críticas para tomadas de decisão) tem se tornado realidade graças aos esforços de organizações como W3C[40] que especificam protocolos que permitem esta aproximação como o SOAP,WSDL e a linguagem XML.

O propósito deste capítulo é exatamente implementar com o suporte destas tecnologias e de algumas outras, o sistema especialista desenvolvido no capítulo anterior. Isso é possível em função da troca de mensagem entre um cliente e um fornecedor de serviços a partir da padronização de alguns conceitos que serão visto adiante.

5.1. O escopo do Sistema Especialista como Web Service

O escopo do sistema especialista abordado neste projeto é o telediagnóstico da malária utilizando o JESS Java Expert System Shell – um pacote Java para construção de mecanismo de inferência.

Uma aplicação *desktop* simula o lado cliente onde as consultas são realizadas e o processo de anamnese acerca da doença é efetuado pelo médico ou agente de saúde. Estes profissionais por sua vez disporão do serviço inteligente no ambiente da internet garantindo um diagnóstico acurado.

Vale ressaltar que a intenção maior da contribuição do projeto é prover áreas endêmicas de difícil acesso, porém onde se tem disponível os recursos da informática, de uma segunda opinião médica acerca do assunto.

Toda a solução como já foi comentada no capítulo 4, possui três fases, no entanto neste capítulo será abordado especificamente a implementação da **fase 1 – Criar um**

[40] W3C World Wide *Web* Consortium responsável pela padronização dos principais protocolos que permite a interoperabilização de serviços no ambiente da internet.

ambiente cooperativo / interativo homem-máquina a partir de bases de regras nos *web Services* como serviço *web*.

O desenvolvimento do sistema especialista através da tecnologia de *servlets* agora por sua vez será então disponibilizado como serviço aos clientes.

5.2. Vantagens e desvantagens do Sistema Especialista como Web Service

A especificação dos requisitos do sistema especialista encontra-se no capítulo 4 seção 4.1. Existe um cenário de atendimento aos casos de malária nas regiões endêmicas e indubitavelmente inóspitas neste país que se configura como precário e via de regra inoperante.

Questões as mais adversas, que inclusive fogem a abordagem do projeto dão este encaminhamento às dificuldades de erradicação da doença. No entanto, como já foi comentado em vários pontos deste trabalho a tecnologia hoje oportunizada pela internet permite que os mínimos processos computacionais presentes na maioria das ações de prevenção e combate a doença nestas regiões possam evoluir para soluções eficazes como os serviços *web*.

Pontualmente se tem o seguinte aspecto positivo e negativo para esta solução com base no cenário considerado genérico e também pela confrontação dos aspectos peculiares do *WST Web Services Telediagnosis* em relação aos sistemas mencionados na Tabela 1.1. Observe desta feita a Tabela 5.1:

SE[41] características	HEALTH Net	LEPIDUS	SEDIL	WST
1.O SE foi desenvolvido para o ambiente da internet ?	Sim	Sim	Não	Sim
2. O SE permite que suas bases de conhecimento sejam disponibilizadas no ambiente de serviços *web* ?	Não	Não	Não	Sim
3. O SE contemplará aprendizagem automática pelas bases de regras ?	Não	Não	Não	Sim
4. O SE tem suporte a linguagem Java ? (Portabilidade)	Sim	Não	Não	Sim

Tabela 5.1 Aspectos comparativos entre sistemas especialistas e *WST*

Vantagens

- Padronização dos mecanismos de acesso às informações através do SOAP. Isso significa que cada sistema (MEDIC) de fornecedores (postos de

[41] SE Sistema Especialista

saúde) diferentes, até mesmo de instituições de combate a doença diferentes possam usufruir da interoperabilidade do sistema, o que descarta a complexidade do projeto de integração;

- O processo de padronização acima mencionado permite também que a expansão para novos clientes usuários do serviço possa se efetuar de maneira prática. Apenas disponibilizando as interfaces comuns ao serviço;

- A praxe é que processo de integração de sistemas diferentes, utilizando plataformas diferentes tenha um custo consideravelmente alto. As tecnologias disponíveis na internet e mais especificamente a tecnologia de *web Services* viabiliza esta iniciativa computacional com uma arquitetura extremamente simples e flexível.

Desvantagens

- Essencialmente o único aspecto negativo do que se propõe o projeto, na verdade não perpassa por questões tecnológicas ou de cunho financeiro, mas por questões de iniciativa política, visto que, o maior empreendedor no combate a malária hoje no país é o governo e as ações destes variam a cada ciclo de mandato. A outra questão é na verdade cultural. Existe uma resistência constatada aos sistemas especialistas, na verdade mais que aos sistemas especialistas às aplicações que se propõem a uma segunda opinião médica, porém este é um paradigma a ser quebrado.

A seguir nas Figuras 5.1 e 5.2 os diagramas que ilustram as situações de atendimento atual e que se propõe o sistema especialista para telediagnóstico no *web Services*.

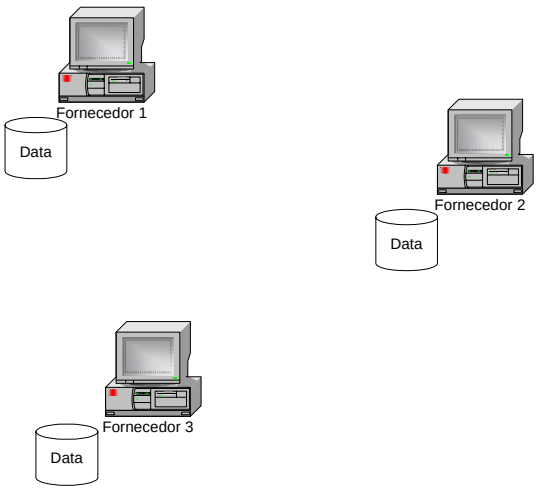


Figura 5.1 Situação do sistema atual

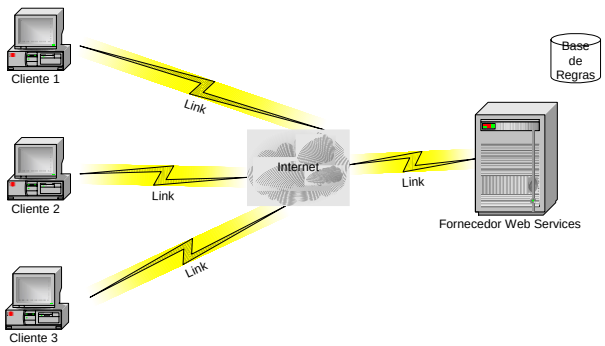


Figura 5.2 Situação proposta *WST*

No sistema atual todos os postos de saúde são na realidade são fornecedores de um serviço de diagnóstico local sem segunda opinião médica. O que significa que cada fornecedor deste serviço tem seu próprio sistema de controle ambulatorial.

A situação proposta , parte da premissa que apesar de cada fornecedor do serviço de anamense possuir seu próprio sistema de anamense, alguns elementos desta são com,uns a todos, e a partir desta filosofia se inicia a padronização e a implementação deste fornecedor como um provedor de serviço *web*.

5.3. Requisitos de software e configurações

Serão efetivamente necessários para implementação do sistema especialista como serviço *web* algumas ferramentas e alguns padrões de configuração tanto do lado do cliente (fornecedor da interface) quanto do lado do servidor (fornecedor do serviço). São elas:

- Java Development Kit 1.4.1_02
- Apache Tomcat versão 4.1.29
- Kit de ferramentas SOAP 2.3.1
- *Web Services* Toolkit WSDP 1_4

Certo de que o objetivo desta implementação é apenas um protótipo, alguns detalhes técnicos serão de fato levados em consideração no artefato final. Isso implica em afirmar que o primeiro ponto a ser abordado será a arquitetura que este projeto irá na prática funcionar.

Tem-se alguns fornecedores de interface e teoricamente um provedor de serviço comum a todos eles, logo para que se possa simular este ambiente em um único computador, instâncias para todos estes (fornecedores) devem ser criadas no servidor de página – Tomcat – como demonstra o diagrama da Figura 5.3.

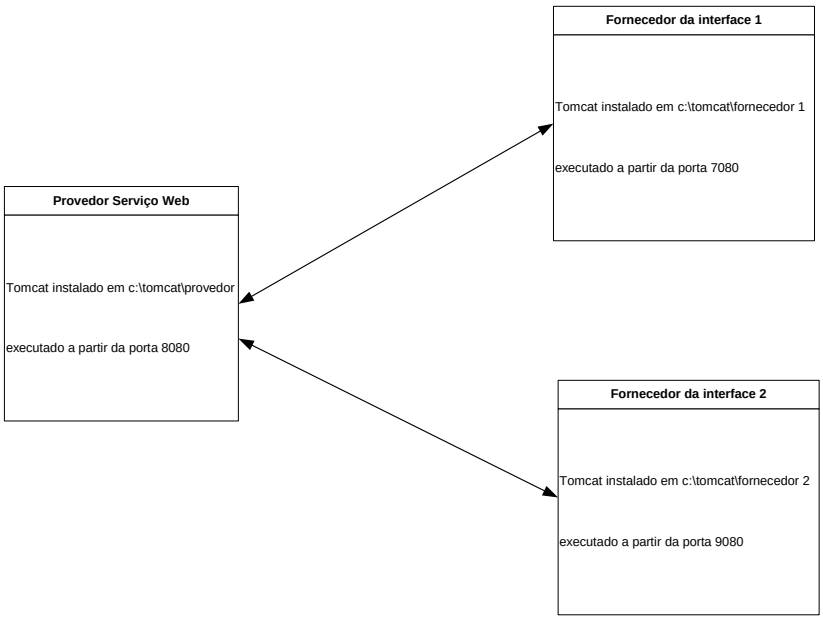


Figura 5.3 Arquitetura do Tomcat para as instâncias dos fornecedores

A próxima ação a ser executada será a configuração dos servidores (instâncias do Tomcat) e do protocolo SOAP para cada fornecedor de interface e provedor de serviço.

Tomcat para o provedor de serviço *web*

- Diretório de instalação do Tomcat `c:\tomcat-Provedor`*`Web`*;
- Neste diretório de instalação do Tomcat, na pasta **\conf** no arquivo **config.bat** editar a linha set `CP=%CLASSPATH% ; %CP`
- Adicionar o SOAP ao diretório do Tomcat `c:\tomcat-Provedor`*`Web`*`\SOAP 2.3.1`
- Finalmente no diretório `c:\tomcat-Provedor`*`Web`*`\conf\server.xml` editar o HTTP Connection para a porta 8080, padrão para o Tomcat.

Instalado e configurado o diretório do Tomcat no provedor do serviço. Ainda neste (provedor) deve ser criado o ambiente de desenvolvimento do serviço *web* onde todo a aplicação de serviço ficara residente. Para isso é necessário:

- Criar um diretório `c:\java`*`webServices`*`\provedorServico`*`Web`*;
- Por sua vez dentro de `c:\java`*`webServices`*`\provedorServico`*`Web`* deverá ser criado um arquivo **config.bat** com o conteúdo abaixo descrito no Apêndice H:

Os fornecedores 1 e 2 são exatamente os usuários (clientes) do serviço *web* que será disponibilizado pelo provedor. São chamados assim neste projeto, visto que, na verdade fornecem cada um a sua própria interface do seu sistema de anamnese para que esta por sua vez seja padronizada pelo provedor de serviço *web*.

Tomcat para o fornecedor 1 e 2

- Diretório de instalação do Tomcat `c:\tomcat-FornecedorX`;
- Neste diretório de instalação do Tomcat, na pasta **\conf** no arquivo **config.bat** editar a linha set `CP=%CLASSPATH% ; %CP`
- Uma cópia da distribuição SOAP deve ser feita para `c:\tomcat-FornecedorX\`*`web`*`apps` ;
- Finalmente no diretório `c:\tomcat-FornecedorX\conf\server.xml` editar o HTTP Connection para a porta 7080 e 9080 respectivamente para os fornecedores 1 e 2.

Da mesma forma que foi criado um ambiente de desenvolvimento e execução para o provedor de serviço, o mesmo deve ser feito agora para os fornecedores 1 e 2.

- Um diretório chamado `c:\java`*`webServices`*`\FornecedorX` deve ser criado;

Nota: Toda a configuração do provedor de serviço é semelhante ao fornecedor da interface. Apenas deve ser alterado o seguinte:

```
set TOMCAT_HOME=C:\tomcat-FornecedorX
set JWS_HOME=C:\java webServices

…… o restante da configuração segue a semelhança do provedor do serviço.
```

Com estes procedimentos encontram-se agora configurados os ambientes de funcionamento do sistema especialista para telediagnóstico. A próxima etapa é a construção das aplicações para troca de mensagens neste ambiente.

A primeira iniciativa para esta implementação é a identificação dos requisitos para o serviço *web* realizar a anamnese. Tipos de interfaces e dados serão trocados entre os dois sistemas no processo de provimento do serviço. Com base na padronização discorrida acima acerca da interface que estará disponível aos fornecedores 1 e 2 se faz necessário então a definição detalhada do(s) método(s) e parâmetros existentes nesta.

```
public interface Anamnese
```

```
{
    String getDiagnostico(String inputReq);
}
```

O arquivo acima deve ser compilado usando-se o `javac Anamnese.java`. O método `getDiagnostico` tem como entrada uma *string* XML contendo as informações da anamnese. Por sua vez é retornado uma string com o diagnóstico.

O último passo na preparação do ambiente é a organização da aplicação no SOAP para os fornecedores 1 e 2. Logo em seguida, a conversão das classes pelo WSDL para o serviço de anamnese.

Finalmente procede-se a implementação da página *web* vista no capítulo 4 que irá prover o acesso ao serviço de telediagnóstico no ambiente de *web Services*.

6 CONCLUSÃO

Não resta dúvida que além do escopo tecnológico, na verdade exista o cunho social da proposta deste projeto. Embora se deva reconhecer que é uma contribuição a somar com inúmeras que agregam ao combate da malária e que almejam estar salvando vidas. Necessário se faz pontuar alguns aspectos (técnicos e sociais) a serem analisados, para que se conclua a verdadeira efetividade da proposta até este ponto dissertada.

É fato que tecnologias viabilizam iniciativas e conseqüentemente viabilizam processos facilitando o cotidiano humano. Fato também é que saúde pública – objeto deste projeto enquanto aplicado a cura da malária – portanto é assunto de abrangência e complexidade incomensurável.

Duas tecnologias tornaram concreta a fase inicial deste projeto como se observa na implementação do protótipo: (a) JESS Java Expert System Shell; (b) *Web Services*. Embora a primordialidade das referidas tecnologias, algumas outras deram suporte a esta concretização como as ontologias e o as metodologias de modelagem do conhecimento da inteligência artificial.

Sistemas especialistas têm sido aplicados aos mais variados seguimentos de solução de problemas, inclusive no apoio ao diagnóstico médico. No entanto parece inusitada uma solução que disponibilize estes sistemas como serviços descentralizados e de maneira totalmente transparente tanto para os usuários como para os desenvolvedores no ambiente da internet.

O JESS, um pacote Java desenvolvido pelo SANDIA Laboratories nos Estados Unidos, lança neste projeto um desafio bem sucedido, pelo menos do ponto de vista da prototipação, pela sua aplicação ao ambiente de *Web Services*.

É dito desafio, visto que, apesar da possibilidade de desenvolvimento de sistemas especialista para o ambiente da *web* contemplada pelo JESS, ainda não se constatam muitas propostas que o apliquem a um ambiente de interoperabilização, nem tampouco que o permita interagir com sistemas legados de maneira simples e funcional através de um *servlet* como foi implementado neste projeto.

O outro aspecto técnico positivo a ser manifestado é com relação à segurança de toda a interoperabilização. A tecnologia de *Web Services* esta se difundindo muito rapidamente, e por ser um padrão aberto, necessita de uma atenção especial quanto às questões de segurança, até por conta da sua infra-estrutura pública, sujeita a ataques. Todo um processo de autorização, autenticação, validação, encriptação e certificação podem ser implementados usando-se este ambiente inclusive com a interação das plataformas onde

funcionam as aplicações legadas, visto que, as maiores "portas" para invasões, comprovadamente são vulnerabilizadas por falhas de aplicação como: entradas não validadas, falhas de injeção de comando ou SQL e tratamento de erros inadequados. Ao contrário do que se imagina com relação a infra-estrutura do próprio ambiente.

Web Services na implementação da fase 1 do protótipo deste projeto se manifestou extremamente flexível, funcional e portável.

Que se comente de maneira analítica, a partir deste ponto os aspectos sociais da prototipação do projeto.

Alguns questionamentos levam a busca de respostas dadas inicialmente pelo processo proforma[42] de prototipação e validação da solução *WST Web Services Telediagnosis*.

1. **Existe atendimento informatizado pelos organismos de saúde que tratam de doenças tropicais nas regiões propostas pelo projeto (regiões inóspitas)?** A FUNASA e o Núcleo de Doenças Parasitárias da Universidade Federal do Maranhão hoje são os dois organismos à frente do combate a doença (malária). Estas duas instituições contam com postos de atendimento onde o acompanhamento ambulatorial – que procede anamnese clínica – possuem computadores em sua grande maioria ligados de alguma forma a redes de computadores.

2. **Que espécie de atendimento informatizado específico existe nestes núcleos?** Foi identificado o seguinte cenário: computadores que a partir de *softwares* como Microsoft Excel, Word ou similares processam e armazenam informações provenientes de pacientes infectados com o *plasmodium*. Identificados ainda alguns software específicos para área médica como o descrito no capítulo 4 que realizam o mesmo trabalho.

3. **As informações coletadas por estes artefatos de software são sociabilizadas e de que forma?** Toda a informação processada nestes núcleos de alguma forma é compartilhada, seja através de correio eletrônico ou a partir de algum tipo de rede de comunicação. No entanto não há nenhum cunho colaborativo ordenado como propõe o projeto *WST- Web Services Telediagnosis*, apenas informação transferida para posterior tratamento.

4. **Qual o nível técnico dos usuários que hoje operacionalizam este ambiente informatizado?** Via de regra heterogêneo. Pessoas com

[42] Proforma visto que o projeto é uma proposta ainda não em operação. Na verdade uma contribuição em forma de protótipo dada como positiva e aceita.

conhecimento de informática operacional e pessoas com visão de tecnologia de informação.

5. **Quais os investimento para área de TI nestes núcleos?** Escassos. Efetivamente apenas computadores com acesso a internet.

6. **Quais as perspectivas comuns destes núcleos quanto a soluções informatizadas?** Além dos investimentos, a necessidade primordial de padronização nos processos de informação e compartilhamento e compartilhamento destas.

Com base nas respostas acima se conclui também pontualmente alguns aspectos sobre a solução proposta pelo projeto *WST- Web Services Telediagnosis* tanto do ponto de vista técnico quanto da realidade presente nestas instituições:

1. O *WST- Web Services Telediagnosis* é um sistema que apesar de possuir uma certa complexidade o que demanda uma mão-de-obra técnica para sua implementação é a solução mais próxima das espectativas de tratamento de informação em conhecimento.

2. Para que se possa implementar um projeto com este escopo, mais do que a mão-de-obra técnica referida no tópico anterior é necessário um trabalho de formação de TI para aqueles envolvidos no ciclo de operação (conhecimento em sistemas de informação a nível teórico), apesar da interatividade do sistema, afinal um sistema especialista envolve o conhecimento sobre um domínio.

3. Os recursos necessários (hardware, software e pessoal) à implementação desta solução em se comparando com o que existe como cenário atual são bem menores, tornando assim o projeto viável sob este aspecto.

4. A transparência dos processos contemplados pelo WST através dos mecanismos de inferência e a interoperabilidade que o ambiente de *web Services* provém minimizam os custos operacionais destes núcleos através inclusive de programas preventivos contra a doença.

Em face do exposto confrontado pontualmente acima se pode concluir que o *WST Web Services Telediagnosis* possui uma diretriz de solução efetiva para os problemas de segunda opinião médica, prevenção à malária, compartilhamento de conhecimento, etc.

Este projeto como já foi comentado está dividido em três fases bem definidas: a fase 1 implementada através de um protótipo neste trabalho, a 2 onde as regras de colaboração manterão atualizadas as bases de regras do sistema especialista e a fase 3 que implementará

interfaces adaptáveis aos mais diversos usuários lançando desta forma um grande desafio a filosofia principal dos *web Services* que é a padronização destas. Estes últimos dois módulos são as sugestões iniciais para trabalhos futuros no contexto deste projeto.

REFERÊNCIAS

[1] AIM Advanced Informatics in Medicine

Disponível em: http://www.qub.ac.uk/ivs/medical/

Acesso em: 23/03/2005

[2] AMORIM, Ricardo José Rocha, **Desenho de um sistema gerenciador inteligente de recursos em um ambiente de aprendizagem cooperativa**, Tese apresentada ao programa de pós-graduação em Engenharia de Produção da Universidade Federal de Santa Catarina como requisito para o grau de Mestre em Engenharia de Produção, 2002

[3] BARBOSA[1,2], Ana Karina, NOVAES[2], Magdala, STAMFORD[1,2], Peter, *et all*, **HealthNet: um Sistema Integrado de Telediagnóstico e Segunda Opinião Médica,** [1]Centro de Informática – UFPE, [2]Grupo de Tecnologias da informação em Saúde (TIS) LIKA - UFPE

[4] BARROS, Leliane Nunes de, SANTOS, Eduardo Toledo, **Um estudo sobre a modelagem do domínio de Geometria Descritiva para a construção de um Sistema Tutor Inteligente,** Depto. de Ciência da Computação, Instituto de Estatística e Matemática - Universidade de São Paulo - Brasil

[5] BLEUMER, Gerrit, **AIM Advanced Informatics in Medicine – Secure Environment for Information Systems in Medicine (SEISMED),** Universität Hildesheim, Institut für Informatik, 1995

[6] CERAMI, Ethan, **Web Services Essentials, Distributed Applications with XML-RPC, SOAP, UDDI & WSDL**, O'Reilly, 2002

[7] FIOCRUZ Fundação Oswaldo Cruz.

Disponível em: http://www.fiocruz.br/ccs/estetica/malaria.htm

Acesso em: 23 de março de 2005

[8] FRIEDMAN-HILL, Ernest, **JESS in action – Rule based systems in java**, Manning - 2003

[9] HENDRICKS, Mack *et.al*, **Java Web Services Profissional**, Ed. Alta Books – 2002 [10] HAYKIN, Simon, **Neural networks,** 2.ed., Prentice-Hall Inc., 1999, USA

[11] IGNIZIO, James P., **Introduction to Expert System Shell**, Ed. McGraw-Hill – 1991

[12] JEWELL, Tyler, CHAPPELL, David, **Java Web Services**, O'Reilly, 2002

[13] LAKATOS, I. **"Falsificação e metodologia dos programas de investigação científica.** In Falsificação e Metodologia dos Programas de Investigação. Lisboa: Edições 70, 1999.

[14] LEVINE, Robert I, DRANG, Diane E., et. all., **Artificial Inteligency and Expert System** McGraw-Hill – 1998 São Paulo

[15] MATTOS, Merisandra Côrtes de, MARCÍLIO, Carla de Melo, et. all., **Modelagem do conhecimento de um tutor inteligente para apoio ao ensino de eletrocardiograma,** Grupo de Pesquisa em Informática Médica e Telemedicina, Universidade do Extremo Sul Catarinense (UNESC), Brasil

[16] MENDES, Carlos Otavio Schocair, **SEDIL Sistema Especialista de Diagnóstico de Doenças Infecto-parasitárias Regionais**, Tese apresentada ao programa de pós-graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão como requisito para obtenção do grau de Mestre em Engenharia de Eletricidade (área de concentração Ciências da Computação), 1999.

[17] NARDON, Fabiane Bizinella, MOURA, Lincoln de Assis Jr., **Ontologias e bancos de dados dedutivos para integração de informações de saúde**, Escola Politécnica, Universidade de São Paulo - SP

[18] PASSARELLI, Brasilina, **Teoria das Múltiplas Inteligências aliada à Multimídia na Educação: Novos Rumos para o Conhecimento**, Escola do Futuro – USP São Paulo-SP, 1998

[19] PENÍN, M. Lama. **Modelo del conocimiento y arquitetura para la síntesis de un especialista terapéutico en Unidades de Cuidados Intensivos y Coronarios**.2000. Tese (Doutorado em Eletrônica e Informática) – Programa de Doutoramento em Eletrônica e Informática do Departamento de Eletrônica e Computação, Universidade de Santiago de Compostela, USC, Espanha.

[20] PRADO JR., Caio. O que é Filosofia. Brasilia: Ed. Brasiliense, 1984.

[21] **Protégé User Guide**, Versão 2.1.1, Stanford University, 2005
Disponível em: http://protege.stanford.edu/index.html
Acesso em: 12/04/2005

[22] Revista Informédica, 1(6): 5-9, 1994.

[23] RUSSELL, Stuart, NORVIG, Peter, **Artificial Intelligence,** 2.ed., Prentice Hall, 2003.

[24] SILVA[1], Roberto, ROQUE[2], Antonio C., **Lepidus On-Line: Sistema de Apoio à Decisão Médica**, [1]Curso de Eng. Biomédica Universidade de Mogi das Cruzes Mogi das Cruzes-SP, Dpto. De Física e Matemática, Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto, Universidade de São Paulo, Ribeirão Preto – SP

[25] SILVA, Rogers Ferreira da, **Plataforma DeskEaD para aplicações de educação à distância,** Rede Metropolitana de Porto Alegre - MetroPoA – FENG – PUCRS

[26] SILVA, Antonio Rafael da, GONÇALVES, Eloísa da Graça do Rosário, et all. **Malária: diagnóstico, tratamento e controle de cura. (Uma nova estratégia de controle),** UFMA Universidade Federal do Maranhão, 2003.

[27] SOTIRIOU[1], Dimitrios, PANOS[2], George, **A Survey of Telemedicine Services Today**, University of Athens[1] , University of Athens[2] , 1992

[28] SUCEN Superintendência de Controle de Endemias. Disponível em: http://www.sucen.sp.gov.br/doencas/malaria/texto_malaria_pro.htm.

Acesso em: 30 de março de 2005

[29] SUCHMAN, Susan, DAVID, Bellin, **Manual de desenvolvimento de sistemas estruturados,** Makron Books, 1993, São Paulo

[30] SZOLOVITS, Peter, SHROBE, Howard, *et all*, **What is a Knowledge Representation?,** MIT AI Lab USA

[31] TELEMEDICNA DA BAHIA (Empresa)

Disponível em: www.telemedicina.com.br

Acesso em: 18 de fevereiro de 2005

[32] TOFFLER, Alvin, **The Third Wave,** Batan Books – 1990

[33] VICARI, Rosa Maria, PEROTTO, Filipo Studzinski, **Modelagem do conhecimento, Sistemas especialistas e o Projeto SEAMED**, Instituto de Informática, Universidade Federal do Rio Grande do Sul

[34] VYGOTSKY, L. **A formação social da mente.** São Paulo, Martins Fontes, 1987.

[35] WIKIPEDIA A Enciclopédia Livre.

Disponível em: http://pt.wikipedia.org/wiki/Fagocitose

Acesso em: 02 de abril de 2005

[36] WINSTON, Patrick Henry, **Artificial Intelligence**, 3ª Ed., Addison-Wesley Publishing Company, 1992

APÊNDICES

Apêndice A

Código JESS para demonstrar a eficácia no funcionamento das regras sobre os fatos.

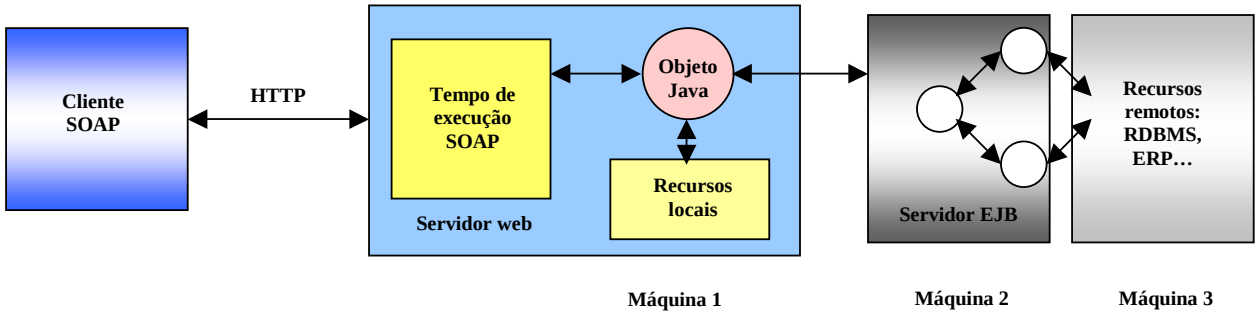
```
Jess> (deftemplate being (slot name))
Jess> (deftemplate mortal extends being)
Jess> (deftemplate immortal extends being)
Jess> (deftemplate monster extends mortal)
Jess> (deftemplate human extends mortal)
Jess> (deftemplate god extends immortal)
Jess> (defrule list-all-humanoids
   ;; fire for all beings, gods, monsters, and humans
   (being (name ?n))
   =>
   (printout t ?n " is a being " crlf))

Jess> (defrule list-all-mortals
   ;; fires only for mortal things
   (mortal (name ?n))
   =>
   (printout t ?n " is mortal " crlf))

Jess> (deffacts beings (human (name Bob)) (monster (name Gollum))
                       (god (name Zeus)))
Jess> (reset)
Jess> (run)
   Zeus is a being
Gollum is mortal
   Gollum is a being
   Bob is a being
   Bob is mortal
5
```

Apêndice B

Arquiteturas do SOAP

Arquitetura distribuída.

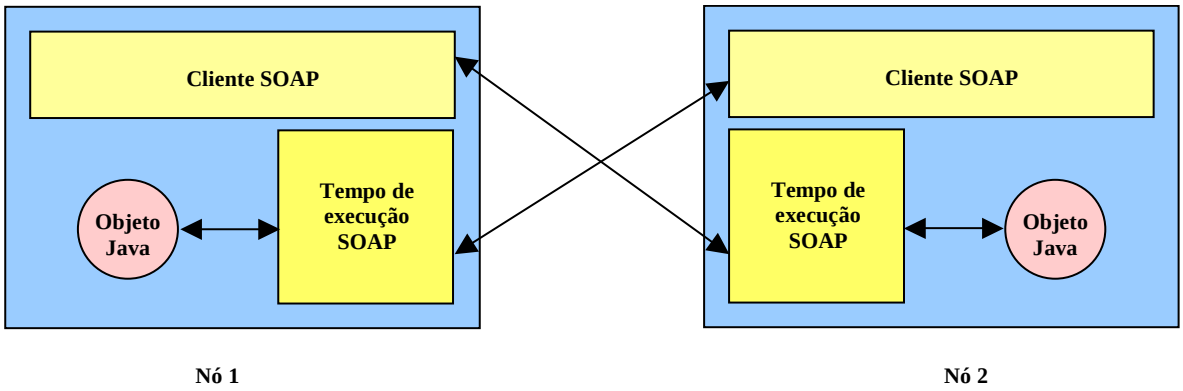
Ídem a arquitetura padrão com a sutil diferença de interagir com outros protocolos distribuídos como o CORBA.



Arquitetura SOAP distribuída

Arquitetura não hierárquica.

Imagine vários nós em uma rede. Cada cliente por sua vez, através de uma mensagem faz uma solicitação de serviço que pode ser aceita ou negada pelo nó que representa o serviço da rede.



Arquitetura SOAP não hierarquica

Apêndice C
Detalhes XML que compõem

uma mensagem SOAP

O envelope: é o elemento de nível mais alto na estrutura de mensagem do SOAP. Dentro dele irão se encontrar o cabeçalho (opcional) e o corpo do da mensagem. O elemento envelope é codificado por: <envelope>, veja abaixo um exemplo:

```
<SOAP-ENV Envelope
    <SOAP-ENV Header></SOAP-ENV Header>
    <SOAP-ENV Body></SOAP-ENV Body>
</SOAP-ENV Envelope>
```

Uma observação interessante é que todos os elementos recebem o prefixo SOAP-ENV associado ao *namespace* http://schemas.xmlsoap.org/soap/envelope.

O cabeçalho: apesar de ser opcional, tem um papel muito útil como os recursos de autenticação de mensagens que são codificados nele, veja o exemplo abaixo:

```
<SOAP-ENV Envelope
    <SOAP-ENV Header>
        <a:authentication
            xmlns:a="http://www.pagina.com/autenticacao">
            <a:username>usuario</a:username>
            <a:password>pwsusuario</a:password>
        </a:authentication>
    </SOAP-ENV Header>
    <SOAP-ENV Body></SOAP-ENV Body>
</SOAP-ENV Envelope>
```

O corpo: é o elemento por assim dizer que contém o núcleo da mensagem no SOAP. É codificado pelo elemento <body>. No corpo do SOAP é onde se codifica a carga útil que será recebida pelo receptor da mensagem, geralmente uma chamada RPC. Veja abaixo:

```
<SOAP-ENV Envelope
    <SOAP-ENV Header>
        <a:authentication
            xmlns:a="http://www.pagina.com/autenticacao">
            <a:username>usuario</a:username>
            <a:password>pwsusuario</a:password>
        </a:authentication>
```

```
        </SOAP-ENV Header>
        <SOAP-ENV Body>
            <cmd:processReboot xmlns:cmd="http://www.pagina.com/cmd">
                <ip xsi:type= »xsd :string »>192.168.1.3</ip>
                <delay xsi:type="xsd:int">30000</delay >
            </cmd:processReboot>
        </SOAP-ENV Body>
</SOAP-ENV Envelope>
```

Apêndice D

Vinculo SOAP com o HTTP

POST /rpcrouter HTTP/1.1

Host 127.0.0.1

Content-type: text/xml; charset=”utf-8”

Content-Length: 559

SOAPAction:

<SOAP-ENV Envelope

<SOAP-ENV Header>

<a:authentication

xmlns:a=”http://www.pagina.com/autenticacao”>

<a:username>usuario</a:username>

<a:password>pwsusuario</a:password>

</a:authentication>

</SOAP-ENV Header>

<SOAP-ENV Body>

<cmd:processReboot xmlns:cmd=”http://www.pagina.com/cmd”>

<ip xsi:type= »xsd :string »>192.168.1.3</ip>

<delay xsi:type=”xsd:int”>30000</delay >

</cmd:processReboot>

</SOAP-ENV Body>

</SOAP-ENV Envelope>

Veja abaixo a resposta a esta solicitação:

http/1.1 200 OK

Content-Type: text/xml; charset=”utf-8”

Content-Length: 320

Apêndice E

Instanciação da classe

javax.wsdl.Definiton

A primeira forma se cria uma instância nova a partir da classe *javax.wsdl.factory.DefinitionFactory.newDefinition( ).*

Caso 1:

```
import com.ibm.wsdl.xml.*;
import javax.wsdl.*;
import javax.wsdl.factory.*;

public class WSDL4JcreateSample
{

  public static void main(String[] args)
  {
    try
    {
      Definition def = DefinitionFactory.newInstance().newDefinition();
}
catch (WSDLException ex)
{
      System.out.println("Exception - fault code : "+ex.getFaultCode());
}
}
```

A segunda forma é se criar uma instância a partir de um documento WSDL existente.

Caso 2:

```
import com.ibm.wsdl.xml.*;
import javax.wsdl.*;
import javax.wsdl.factory.*;

public class WSDL4JreadSample
{

  public static void main(String[] args)
  {
    try
    {
      Definition def = WSDLReader.readWSDL(null, new
InputSource("http://localhost:8080/wrox/wsdl/AddressBook-service.wsdl"));
    }
    catch (WSDLException ex)
    {
      System.out.println("Exception - fault code : "+ex.getFaultCode());
    }
}
```

Apêndice F
Funcionamento do Framework WSIF

Em síntese o funcionamento se dá da seguinte forma:

1. O documento WSDL é lido a partir de um objeto *javax.wsdl.Definiton*
2. Ao usar este objeto acima, um outro objeto *com.ibm.wsif.WSIFDynamicPortFactory* é criado. Fazendo com que todos os tipos usados no XML declarados na WSDL e também nas classes Java associadas, serão por sua vez declarados neste factory.
3. Um objeto *com.ibm.wsif.WSIFPort* é obtido do factory.
4. Todas as mensagens e partes dos documentos WSDL são incluídas nos objetos WSIF (*com.ibm.WSIFMessage* e *com.ibm.wsifWSIFPart*).
5. Finalmente o método *executeRequestResponseOperation( )* no objeto *com.ibm.wsifWSIFPort* é usado para chamar o serviço.

```
import javax.wsdl.*;
import org.xml.sax.*;
import java.io.*;
import com.ibm.wsdl.xml.*;
import com.ibm.wsif.*;

public class WSIFSample {
  public static void main(String[] args) throws Exception {
    Definition def = WSDLReader.readWSDL(null, new
InputSource("http://localhost:8080/wrox/wsdl/AddressBook.wsdl"));

    WSIFDynamicPortFactory dpf = new WSIFDynamicPortFactory(def);
    dpf.mapType( new
QName("http://www.addressbook.com/schemas/AddressBookRemoteInterface",
"Address"), Address.class);

    WSIFPort port = dpf.getPort();

    WSIFMessage input = port.createInputMessage();
    WSIFMessage output = port.createOutputMessage();
    WSIFMessage fault = port.createFaultMessage();

    WSIFPart inputPart = new WSIFJavaPart(java.lang.String.class, "Joe
Smith");
    input.setPart("person", inputPart);

    port.executeRequestResponseOperation("getAddress", input, output,
fault);

    WSIFPart outputPart = output.getPart("result");
    Address address = (Address)outputPart.getJavaValue();
```

```
    System.out.println("Resulting address is:");
    System.out.println("\tStreet  : "+address.getStreet());
    System.out.println("\tCity    : "+address.getCity());
    System.out.println("\tZipcode : "+address.getZipcode());
  }
}
```

O exemplo acima, mostra como é possível manipular um serviço a um nível da WSDL sem levar em conta o protocolo que ele suporta (SOAP ou http, etc.).

Apêndice G

Código padrão para codificação do sistema – evento salvar registro.

O código inicialmente verifica a existência de uma chave para o registro já existente na tabela do médico – CC04MEDICT, tomando assim a decisão automática em considerar o registro como novo, salvando-o, ou atualizando-o caso este já exista na referida tabela.

```
Sub P_Button_Salvar()
On Error GoTo erro

' Testa se campos obrigatórios estão vázios.
   If Trim(txtCodMed.Text) = G_Vazio Or _
      Trim(txtNomMed.Text) = G_Vazio Or _
      Trim(txtFonMed.Text) = G_Vazio Or _
      Trim(txtCRMMed.Text) = G_Vazio Then
      MsgBox GC_CAMPO_OBRIGATORIO, vbInformation + vbOKOnly, CG_NOME_SISTEMA
      Exit Sub
   End If

'Consultar o registro na tabela.
   G_SQL = "SELECT * FROM CC04MEDICT WHERE CC04CODMED = " &
Format(Trim(txtCodMed)) & " "

   Set G_RES = G_CONEXAO.OpenResultset(G_SQL)

'Se existir atualizar senão salvar.
   If Not G_RES.EOF Then
      GV_SALVAR_ALTERAR = GC_CONFIRMA_ALTERACAO
   Else
      GV_SALVAR_ALTERAR = GC_CONFIRMA_INCLUSAO
   End If

'Confirma se deseja salvar o registro.
   lResp = MsgBox(GV_SALVAR_ALTERAR, vbQuestion + vbYesNo, CG_NOME_SISTEMA)

   If lResp = G_No Then
      Exit Sub
   End If

   If GV_SALVAR_ALTERAR = GC_CONFIRMA_INCLUSAO Then
'SQL para inserir registro na tabela.
      G_SQL = " insert into CC04MEDICT ( CC04CODMED, CC04NOMMED, CC04FONMED, " & _
            "CC04CELMED, CC04ESPMED, CC04CRMMED ) " & _
            " values  ( " & Trim(txtCodMed) & ", '" & Trim(txtNomMed) & "', " & _
            "'" & Trim(txtFonMed) & "', '" & Trim(txtCelMed) & "', " & _
            "'" & Trim(txtEspMed) & "', '" & Trim(txtCRMMed) & "') "

      G_CONEXAO.Execute (G_SQL)

      MsgBox GC_SALVAR_SUCESSO, vbInformation + vbOKOnly, CG_NOME_SISTEMA
      Call P_MontaLista_Medicos
   ElseIf GV_SALVAR_ALTERAR = GC_CONFIRMA_ALTERACAO Then
'SQL para atualizar registro na tabela.
      G_SQL = " UPDATE CC04MEDICT " & _
            "SET CC04NOMMED = '" & Trim(txtNomMed) & "' , " & _
            " CC04FONMED = '" & Trim(txtFonMed) & "' , " & _
            " CC04CELMED = '" & Trim(txtCelMed) & "' , " & _
            " CC04ESPMED = '" & Trim(txtEspMed) & "' , " & _
```

```
            " CC04CRMMED = '" & Trim(txtCRMMed) & "'" & _
            "WHERE CC04CODMED = " & Format(Trim(txtCodMed)) & " "

        G_CONEXAO.Execute (G_SQL)

        MsgBox GC_ALTERAR_SUCESSO, vbInformation + vbOKOnly, CG_NOME_SISTEMA
        Call P_MontaLista_Medicos
    End If

'Habilitar botão excluir.
    Toolbar1.Buttons(G_Excluir).Enabled = True

Exit Sub
erro:
MsgBox "A operação resultou no seguinte erro : " & vbCrLf & Err.Description

End Sub
```

Apêndice H
Ambiente de desenvolvimento do
serviço web – arquivo config.bat

```
set TOMCAT_HOME=C:\tomcat-ProvedorWeb
set JWS_HOME=C:\javawebservices
set classpath=%classpath% ; %JWS_HOME%\lib\xerces.jar
set classpath=%classpath% ; %JWS_HOME%\lib\soap.jar
set classpath=%classpath% ; %JWS_HOME%\lib\mail.jar
set classpath=%classpath% ; %JWS_HOME%\lib\uddi.jar
set classpath=%classpath% ; %JWS_HOME%\lib\activation.jar
set classpath=%classpath% ; %JWS_HOME%\lib\jmxc.jar
set classpath=%classpath% ; %JWS_HOME%\lib\jmxx.jar
set classpath=%classpath% ; %JWS_HOME%\lib\log.jar
set classpath=%classpath% ; %JWS_HOME%\lib\jdom.jar

set classpath=%classpath% ; %JWS_HOME%\lib\provedorServicoWeb.jar
set classpath=%classpath% ; %JWS_HOME%\lib\fornecedor 1.jar
set classpath=%classpath% ; %JWS_HOME%\lib\fornecedor 2.jar

set path=%TOMCAT_HOME%\bin ; %PATH%
set path=%PATH%\bin ; C:\wsdp 1_4\bin
```

Apêndice I

Código da interface da aplicação de serviço web e arquivo de mensagem XML

```
import java.io.StringReader;

//JDOM packages required for XML parsing and XML Document creation
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.input.SAXBuilder;
import org.jdom.output.XMLOutputter;

public class Provider1Appointment implements Appointment {

  /** Creates new Provider1Appointment */
  public Provider1Appointment() {
  }


  public String reserveAppointment(String inputReq) {
    System.out.println("Received Appointment");
    System.out.println("--------------------");

    //Parse the Appointment Request
    try {
      SAXBuilder builder = new SAXBuilder();
      Document AppointmentReq = builder.build(new StringReader(inputReq));
      Element AppointmentReqRoot = AppointmentReq.getRootElement();
      String providerId = AppointmentReqRoot.getAttribute
("PROVIDERID").getValue();
      String patientId  =
AppointmentReqRoot.getChild("PATIENTID").getText();
      String apptId     = AppointmentReqRoot.getChild("APPOINTMENT-
ID").getText();

      //Prepare Response
      Element AppointmentRespRoot = new Element("PROVIDER-
APPOINTMENTRESPONSE");
      AppointmentRespRoot.setAttribute("PROVIDERID","1");
      AppointmentRespRoot.addContent(new
Element("PATIENTID").setText(patientId));
      AppointmentRespRoot.addContent(new Element("APPOINTMENT-
ID").setText(apptId));
      AppointmentRespRoot.addContent(new Element("STATUS").setText("OK"));
      Document AppointmentResponse = new Document(AppointmentRespRoot);
      XMLOutputter outputter = new XMLOutputter("  ",true);
      return outputter.outputString(AppointmentResponse);

    } catch (Exception e) {
      System.out.println("ERROR");
      return "ERROR";
    }
  }
}
```

Entrada

```
<?xml version=”1.0” encoding=”UTF-8”?>
<FORNECEDOR-ANAMNESEREQUEST=”1”>
        <FEBRE>sim</FEBRE>
```

```
        <FEBRE-REG>sim</FEBRE-REG>
        <TEMPERATURA>39</TEMPERATURA>
                            .
                            .
                            .
</FORNECEDOR-ANAMNESEREQUEST>
```

Saída

```
<?xml version=”1.0” encoding=”UTF-8”?>
<FORNECEDOR-ANAMNESERESPONSE=”1”>
        <DIAGNOSTICO>positivo</DIAGNOSTICO>
                            .
                            .
                            .
</FORNECEDOR-ANAMNESERESPONSE>
```