UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA METODOLOGIA DE DESENVOLVIMENTO DE PROGRAMAS EM INTELIGÊNCIA ARTIFICIAL: MEDSIA

MAYSA REGINA MEDEIROS

Prof. Dr. JORGE MUNIZ BARRETO

Florianópolis, agosto de 2004.

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MAYSA REGINA MEDEIROS

UMA METODOLOGIA DE DESENVOLVIMENTO DE PROGRAMAS EM INTELIGÊNCIA ARTIFICIAL: MEDSIA

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Prof. Dr. JORGE MUNIZ BARRETO

Florianópolis, agosto de 2004.

UMA METODOLOGIA DE DESENVOLVIMENTO DE PROGRAMAS EM INTELIGÊNCIA ARTIFICIAL: MEDSIA

MAYSA REGINA MEDEIROS

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Conhecimento e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Prof. Dr. Raul S. Waslawick

Banca Examinadora

_____

Prof. Dr. Jorge Muniz Barreto

_____

Prof. Dr. Mauro Roisenberg

_____

Prof. Dr. Jovelino Falqueto

_____

Prof. Dr. João Bosco da Mota Alves

EPÍGRAFE

Caros Amigos e colegas,

"Quero dizer o que disse a Herbert Simon ao telefone, há alguns minutos atrás: estou profundamente sensibilizado pela notícia que um programa de computador tenha finalmente ultrapassado o campeão do mundo de xadrez num jogo completo. Para todos aqueles que como nós assistiram ao nascimento do campo da Inteligência Artificial, este era sem dúvida o problema do grande desafio. O xadrez computacional não é, certamente toda a Inteligência Artificial, mas como o primeiro amor, fica conosco para sempre. Herbert Simon disse-me: "Bem, talvez eu não tenha sido demasiado preciso ao prever o futuro em dez anos, mas fiz tudo o que pude para ser correto num horizonte de cinqüenta anos..."

Edward A. Feigenbaum

OFERECIMENTO

Dedico este trabalho a Deus, por ter me dado força, serenidade, sabedoria e paciência para superar todos estes dias entre as salas de aulas, laboratório e as estradas da vida em busca desta conquista profissional e pessoal. Aos meus pais Arlete Alves, Eleazar Pereira da Silva e Moisés Alves Medeiros, orgulho de minha alma, de um amor sem limites e por simplesmente tudo! Aos meus avós por todo amor, in memoriam.

Ao meu filho que com o seu nascimento, renovou minha vida, me dando forças para transpor todas as barreiras. Ao meu marido Paulo Edison Horikawa, companheiro de todas as horas que me impulsionou a seguir em frente, e pela compreensão pelas horas que estive ausente. Ao Tókio Horikawa e Helena Gontarz Horikawa; Valdir Rui Horikawa e Doralice Adorina Horikawa por todo apoio e carinho ao longo desta caminhada.

Ao Apolinário Stuhler e Clotilde Rohden Stuhler que participaram e contribuíram para o meu crescimento intelectual, por todo apoio, carinho com que me têm prestigiado.

Ao meu orientador Prof. Jorge Muniz Barreto e Solange Quintella Barreto pelas inúmeras sugestões e contribuições realizadas durante o desenvolvimento deste trabalho.

AGRADECIMENTO

Sou profundamente grata a Deus, pela luz, pela força, coragem e perseverança ao longo de todo este período. Aos meus pais Arlete Alves, Eleazar Pereira da Silva e Moisés Alves Medeiros pelo apoio e pelo amor incondicional! Aos meus avós por todo amor, in memoriam. Ao meu filho por motivos de muita ternura, orgulho de minha alma, de um amor sem limites. Ao meu marido Paulo Edison Horikawa, pelo muito amor que me tem prestigiado, pela compreensão e confiança, por toda a força e apoio ao longo deste período, e pelos importantíssimos "Boa Sorte, tudo vai dar certo!".

Ao Tókio Horikawa, Helena Gontarz Horikawa, Valdir Rui Horikawa e Doralice Adorina Horikawa pelo carinho constante a mim direcionado.

Desejo registrar minha profunda gratidão a Apolinário Stuhler e Clotilde Rohden Stuhler que muito colaborou para esta conquista profissional e pelo prazer de suas amizades.

Ao Msc. Ewerson Duarte da Costa que colaborou com este trabalho, pelas informações fornecidas, pela troca de idéias, e pelas suas sugestões.

Ao Prof. Msc. Cristiano Maciel pelas dicas e aconselhamentos ao longo desta etapa, pelo apoio incondicional e pela grande amizade que me tem dedicado.

Ao meu orientador Prof. Dr. Jorge Muniz Barreto e Solange Quintella Barreto pelo valioso conhecimento que me forneceram, pela confiança que sempre imprimiu às nossas discussões, por ter viabilizado trilhar todos os caminhos; pela orientação precisa, pela afetividade, pelos inúmeros momentos de apoio e estímulo.

A todos gostaria de exprimir os maiores agradecimentos e aqui reconhecer o seu importante contributo, este trabalho dedico a vocês.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AC	Aquisição de Conhecimento
ACM	*Association for Computing Machines*
BC	*Knowledge Base*
CE	Computação Evolutiva ou Inteligência Artificial Evolucionária
CEE	*Comission Économique Européenne*
CYGMA	*Cycle de vie e Gestion des Métiers et des Applications*
DM	*Data Mining* (Mineração dos dados)
EC	Engenheiro do Conhecimento
EP	Engenharia de Programas
ESPRIT	*European Strategic Programme in Research in Information Technology*
IA	Inteligência Artificial
IAC	Inteligência Artificial Conexionista
IAH	Inteligência Artificial Híbrida
IAS	Inteligência Artificial Simbólica
IAE	Inteligência Artificial Evolucionária ou Computação Evolutiva
KADS	*Knowledge Acquisition and Development System*
KQML	*knowledge Query Manipulation Language*
KOD	*Knowledge Oriented Design*
MDS	Metodologia de Desenvolvimento de Sistemas
MEDSIA	MEtodologia de Sistemas em Inteligência Artificial
MIKE	*Model-Based and Incremental Knowledge Engineering*

MKSM	*Methodology for Knowledge System Management*
PLN	Processamento de Linguagem Natural
RC	Representação do Conhecimento
SBC	Sistemas Baseados em Conhecimento
SE	Sistemas Especialistas
SEC	Sistemas Especialistas Conexionistas
SES	Sistemas Especialistas Simbólicos
SICDIRE	Sistema Conexionista para Diagnóstico em Reumatologia
SISEADJU	Sistema Especialista de Apoio a Decisão Judicial
UFSC	Universidade Federal de Santa Catarina
UML	*Unified Modelling Language*
VDL	*Vien Development Language*
VDM	*Vien Development Method*

RESUMO

As mais populares metodologias de construção de programas em Inteligência Artificial - IA, KADS, MIKE E PROTÉGÉ, são apresentadas e comparadas. Desta comparação e avaliação das principais limitações são detectadas e uma nova metodologia é sugerida, que será chamada MEDSIA (Metodologia para Desenvolvimento de sistemas em IA). A principal diferença entre esta metodologia proposta e as apresentadas é que ela considera novos desenvolvimentos de métodos para resolver problemas de IA, métodos estes que se tornaram populares por volta do final do século passado e são: abordagem conexionista e a evolucionária. Para ilustrar MEDSIA foram desenvolvidos dois programas: 1 – Sistema de Ajuda a Decisão Jurídica no caso de Pensões Alimentícias e, 2 – Sistema Especialista Conexionista de Diagnóstico Médico no campo da Reumatologia.

ABSTRACT

The most well known techniques, KADS, MIKE and PROTÉGÉ are outlined and compared. From this comparison and evaluation the main limitations are detected and a new methodology is suggested, named MEDSIA. The main difference between this new methodology and the mentioned above is that it considers the new developments of Artificial Intelligence that emerged by the end of last century and are now well known: the connectionist and evolutionary approaches. To illustrate MEDSIA, two case studies are presented: a decision advisor in juridic veredict limited to alimony and a second example of connectionist expert system in rheumatology.

PUBLICAÇÕES

MEDEIROS, Maysa Regina; BARRETO, Jorge Muniz. **Metodologias para IA: Sonho ou Tecnologia?** In: IT CONFERENCE SUCESU – MT 2003 - VI Escola Regional da Sociedade Brasileira de Computação. Cuiabá – MT: SUCESU/SBC/UNIRONDON, Outubro. 2003. ISBN 858844269-8.

MEDEIROS, Maysa Regina; BARRETO, Jorge Muniz. **Estudo Comparativo sobre Metodologias de Desenvolvimento de Sistemas Especialistas.** In: IT CONFERENCE SUCESU – MT 2003 - II SEMINÁRIO DE INFORMÁTICA. Cuiabá – MT: UNIC, Outubro. 2003.

PALESTRA PROFERIDA

MEDEIROS, Maysa Regina; BARRETO, Jorge Muniz. **Considerações sobre Metodologias de Desenvolvimento de Programas em Inteligência Artificial.** In: II JORNADA ESTADUAL DE EDUCAÇÃO MATEMÁTICA - I SIMPÓSIO DA COMPUTAÇÃO. Palestra proferida em: Cáceres – MT, UNIVERSIDADE ESTADUAL DE MATO GROSSO, Junho. 2003.

1. INTRODUÇÃO

1.1. Motivação

Existem muitas idéias contestáveis e conceitos mal formulados a respeito do que é essencialmente a IA. Para a grande maioria das pessoas a IA é apresentada em filmes, romances e outros, quase sempre como ficção científica. Isto leva a crer ser esta uma tecnologia do futuro sem aplicações no mundo atual. Tal fato é um mito. Para desmistificar o mito é bom dar ênfase que a IA permite fornecer paradigmas para o desenvolvimento de sistemas com capacidade de simular o raciocínio.

A importância científica, industrial e econômica da IA (Inteligência Artificial) cresceu muito nos últimos anos e a utilização começa a crescer em muitos campos de aplicação. Alguns países já estão se dedicando a essas aplicações, como por exemplo, a Europa, Japão e EUA. Ainda não se tem uma padronização no processo de desenvolvimento de programas, mas, os sistemas estão se tornando cada vez mais populares.

Será o interesse em IA recente? Não. O desejo de criar artefatos capazes de reproduzir um comportamento inteligente encontra suas origens no passado. Exemplos vão desde o distribuidor de água em Delfos, na Grécia antiga, passam por dispositivos mecânicos de tempos remotos aos quais se atribuía inteligência, até chegar nos sofisticados programas atuais (BARRETO, 2001).

O rápido crescimento de sistemas em IA está tornando-os cada vez maiores e complexos e os recentes avanços vêm fazendo com que estes sistemas sejam vistos como a base de uma futura geração de novos sistemas com inteligência. O fator mais importante é que se espera mais sucesso agora do que antes, pelo fato do grande interesse da comunidade científica. Com este advento é importante que se tenha métodos e ferramentas que sirvam de auxílio nas etapas de desenvolvimento.

Como a crescente implementação tem gerado intensa demanda por profissionais, é necessário que o desenvolvedor tenha habilidade individual em produzir sistemas bem

documentados e com qualidade.

Nas últimas décadas houve o aprimoramento constante das técnicas e metodologias para a engenharia de programas visando acompanhar a evolução natural do conhecimento, incluindo a análise e o projeto estruturado, a análise essencial e a engenharia de informação (FURLAN, 1998).

Parte do trabalho dos pesquisadores envolvidos com a Engenharia de Programas tem sido descrever modelos que compreendam os processos de desenvolvimento dentro de um paradigma conhecido. A existência de um modelo é apontada como um dos primeiros passos em direção ao gerenciamento e a melhoria do processo de *software* (KELLNER, 1989). A fase de projeto é responsável por incorporar requisitos tecnológicos aos requisitos essenciais. Assim, o projetista deverá estar atento aos critérios de qualidade que o sistema terá que atender (ROCHA, *et. al.,* 2001).

Desde a demanda inicial pelo cliente até que finalmente o sistema seja posto em operação, o desenvolvimento de programa envolve diversas fases, caracterizadas por atividades. Para descrever o desenvolvimento de sistemas, diversos modelos de ciclo de vida têm sido propostos pela Engenharia de *Software*, e a seleção de um deles deverá servir à estratégia que se pretende adotar.

É possível dizer que definir o ciclo de vida em um desenvolvimento é o passo mais importante na construção de sistemas. Normalmente é feito com base em um modelo adequado à situação que se quer representar, trazendo como conseqüência o desenvolvimento de sistemas práticos e documentados.

Uma metodologia de desenvolvimento de programa faz o refinamento do ciclo de vida, especificando um conjunto completo, único e coerente, de princípios, técnicas, notações, linguagens de representação e procedimentos.

Este trabalho ressalta a importância em se utilizar metodologias para desenvolvimento de programas em IA. Entretanto, ainda não se tem uma metodologia unificada que nos ofereça métodos eficazes para o auxílio no desenvolvimento. Após cerca de 60 anos de pesquisa entramos em uma nova era onde já é possível determinar com eficácia qual o melhor paradigma a ser aplicado para determinados problemas. E porque não podemos também escolher a melhor metodologia a ser empregada no desenvolvimento de sistema em IA?

1.2. Objetivos

1.2.1. Objetivo Geral

A presente dissertação de mestrado propõe-se a apresentar uma metodologia de desenvolvimento de sistemas em IA tendo como base um estudo bibliográfico sobre as metodologias de desenvolvimento de sistemas de IAS, adaptando-as para as aplicações em Inteligência Artificial de IAC.

1.2.2. Objetivos Específicos

- Introduzir considerações sobre modelos de ciclo de vida de sistemas convencionais as metodologias existentes de Inteligência Artificial.
- Elaborar um modelo de ciclo de vida voltado para a IA.
- Aplicar a metodologia proposta em um Estudo de Caso.
- Desenvolver dois sistemas usando a abordagem simbólica e a abordagem conexionista para validar e ilustrar a metodologia sugerida.
- Auxiliar a comunidade científica, os profissionais e acadêmicos no tocante a conceito, estruturação e retorno que o conhecimento adquirido e apreendido durante o ciclo de vida de um sistema poderá proporcionar.

1.3. Justificativa

A utilização de uma boa prática de engenharia de sistemas tanto na escolha da linguagem de modelagem quanto na especificação, documentação, comunicação e visualização dos artefatos de forma eficiente traz vantagens no desenvolvimento, tais como:

- Sistemas robustos, portáveis e modificáveis.
- Aumenta a confiabilidade do sistema.
- Reduz riscos.
- Cumprimento de prazos.
- Melhora a precisão e consistência do sistema.
- Qualidade do sistema melhorada.

- Manutenção facilitada.
- Redução da taxa de defeitos.

Ao aplicar uma metodologia em um desenvolvimento de programa é possível definir como deve ser feito o sistema, onde se inicia o detalhamento que deve incorporar as características da tecnologia a ser empregada. Visando a verificação da qualidade dos sistemas, alguns fatores a serem considerados são: portabilidade, confiabilidade, eficiência, correção, robustez, facilidade de implantação, clareza e facilidade de manutenção, sendo que, se podem definir os tipos de manutenção tais como: corretiva, adaptativa e preventiva.

Sendo assim, desde o início do desenvolvimento busca-se que o sistema tenha um processo sistemático de desenvolvimento e garantia da qualidade, utilizando técnicas modernas de Engenharia de Programas. E, somando a isso, deve ao longo de todo o ciclo de vida, ser feito à verificação e validação do sistema.

Existem muitos elementos que contribuem para uma empresa de *software* de sucesso; um desses componentes é a utilização de alguma metodologia para desenvolvimento (BOOCH, *et. al.,* 2000). Pensando neste contexto, é possível ressaltar a importância de uma metodologia empregada no desenvolvimento de programas para o domínio de IA.

Com o crescimento de sistemas mais complexos no domínio da IA torna-se cada vez mais importante ter a definição do processo que sistematize o seu ciclo de vida, possibilitando uma maior habilidade de descrever formalmente o comportamento dos modelos. Sendo necessário fazer um acompanhamento de sua execução, guiando os desenvolvedores com uma representação de conhecimento que permita modelar, projetar, avaliar, manutenir, validar, verificar, reusar e acompanhar o processo de desenvolvimento destes programas.

A motivação para este trabalho é encontrada nesta fundamentação de idéias acima demonstrada. A importância da utilização de ferramentas de Engenharia de Programas voltados para a construção de sistemas no domínio de Inteligência Artificial traz qualidade e eficiência pelo fato de ser possível não só desenvolver, mas também estruturar e documentar o programa, evitando assim problemas conceituais que antes não poderiam ser visualizados.

1.4. Estrutura do Trabalho

Visando proporcionar uma visão geral deste projeto o trabalho está organizado da seguinte forma: O Capítulo 1, conceitua os objetivos gerais e específicos e a justificativa sobre o estudo a ser elaborado.

No Capítulo 2, se apresenta uma introdução aos conceitos básicos, onde se fará um breve relato acerca da Inteligência Artificial e suas definições, abordagens, suas origens, seus conceitos, a evolução no que diz respeito à sua história.

No Capítulo 3, o enfoque recai sobre os Métodos e Técnicas de Desenvolvimento de Programas, explanando os modelos existentes na Engenharia de Programas, bem como as seções relativas a Metodologias para Desenvolvimento de Sistemas em IA, Engenharia do Conhecimento, Engenheiro de Conhecimento, Aquisição e Representação de Conhecimento, definições de ciclo de vida, metodologias e ferramentas existentes para desenvolvimento de sistemas voltado para a IA. Este capítulo terá a finalidade de fornecer subsídios para a escolha da nova metodologia proposta.

No Capítulo 4, discute-se sobre a metodologia proposta MEDSIA (MEtodologia de Desenvolvimento de Sistema em IA), mostrando suas fases e atividades. No Capítulo 5, são apresentados dois exemplos para ilustrar a MEDSIA: um de aconselhamento na área de Direito de Família – Investigação de Paternidade e outro para Diagnóstico em Reumatologia.

No Capítulo 6, finalmente, apresentam-se as considerações e conclusões finais do trabalho. Faz-se uma nova síntese de todo o projeto. São discutidos os resultados obtidos apontando suas principais características. É dedicado um cuidado particular à identificação de oportunidades e necessidades de trabalhos futuros, na procura de uma evolução.

No Glossário se apresenta definições de termos usados durante todo este trabalho. Nas Referências Bibliográficas, se apresenta o levantamento bibliográfico que servirá de base para tecer todo o embasamento teórico do projeto.

2. INTELIGÊNCIA ARTIFICIAL

Este capítulo trata sobre IA, suas características e abordagens, apesar de ser largamente explorada em livros, pois alguns aspectos relevantes à pesquisa devem ser ressaltados. O leitor interessado em aprofundar o assunto deve consultar bibliografia adequada, por exemplo: (AZEVEDO, *et. al.*, 2000), (BARRETO, 2001), (BITTENCOURT, 1996), (CHARNIAC & MCDERMOTT, 1985), (CHORAFAS, 1988), (FEIGENBAUM & MCCORDUCK, 1983), (FRIEDERICH & GARDANO, 1989), (GANASCIA, 1993), (HARMON & KING, 1985), (HAYKIN, 2001), (HOLLAND, 1975), (KELLER, 1991), (LANGTON, 1995), (LEVINE, *et. al.,* 1988), (MCGRAW & BRIGGS, 1989), (MINSKY & PAPERT, 1969), (NILSSON, 1982), (RABUSKE, 1995), (RUSSEL & NORVIG, 1995), (SIMONS, 1988), (TAFNER, *et. al.*, 1995), (WINSTON, 1981).

A área de IA tem se destacado na busca por compreender a inteligência englobando diversos campos de conhecimento objetivando simular a inteligência. Fornecendo métodos e técnicas para o desenvolvimento de programas que simulam nas máquinas comportamentos inteligentes, tornando os computadores capazes de tomar decisões como também resolver questões complexas para serem resolvidas por um programa comum (MEDEIROS & BARRETO, 2003).

O objetivo da IA é o estudo e modelagem da inteligência tratada como um fenômeno. A inteligência, propriedade emergente de um número enorme de neurônios, é algo extremamente complexo, resultado de milhões de anos de evolução. Entendê-la não é tarefa fácil, embora existam muitas conclusões relevantes, ainda há muito a ser desvendado, uma vez que não existe uma teoria completa sobre a mente humana e os processos de raciocínio (BITTENCOURT, 1996), (JOHNSON, 2001), (SIMONS, 1988).

IA é a parte da ciência da computação que cria estruturas computacionais inteligentes, tais como a linguagem natural, aprendizado, raciocínio e resolução de problemas (RICH & KNIGHT, 1993).

Os sistemas decisórios, robótica, desenvolvimento de linguagem natural, percepções visuais, prova de teoremas e banco de dados inteligente, dentre outros, são algumas das áreas de aplicação da Inteligência Artificial.

Não é possível se afirmar até onde a IA poderá chegar na intenção de criar máquinas capazes de "pensar". Serão necessários ainda muitos anos de estudo para que se compreenda todos os fatores (conscientes, inconscientes, cognitivos, instintivos, dentre outros) envolvidos no processo do pensar. No entanto, alguns aspectos da inteligência já podem ser implementados e utilizados para auxiliar ou substituir o homem na realização de tarefas. E mesmo quando se chegar à compreensão dos fatores envolvidos nesse processo ainda haverá o desafio de encontrar formas de implementá-los no computador e desenvolver equipamentos adequados para otimizar essa implementação (GANASCIA, 1993).

A IA é simplesmente a transferência das características da inteligência biológica para as máquinas. Os SE (sistemas especialistas) lidam com uma pequena área técnica que pode ser convertida da inteligência humana para a artificial (LEVINE, *et. al.,* 1988), (CHARNIAC & MCDERMOTT, 1985).

A IA fornece métodos e técnicas para o desenvolvimento de programas que simulam nas máquinas comportamentos inteligentes, isto é, tornam os computadores capazes de pensar e tomar decisões (WINSTON, 1981). Por isso, as técnicas de IA necessitam de uma grande quantidade de conhecimentos e de mecanismos de manipulação de símbolos. Esses conhecimentos devem ter a possibilidade de representação, modificação e ampliação (GANASCIA, 1993).

Uma técnica em IA é um método de explorar os conhecimentos que devem ser representados de tal modo que possam (RICH & KNIGHT, 1993):

1. capturar generalizações, ou seja, agrupar situações que partilhem propriedades importantes;
2. ser de fácil compreensão para os usuários que farão a entrada dos dados para a aprendizagem. Em alguns programas de IA os dados são obtidos automaticamente (leitura de vários instrumentos), em outros domínios, a maioria dos conhecimentos que o programa possui deve ser fornecida através da interação com o usuário;

3. ser facilmente modificado para corrigir erros e para refletir mudanças (flexibilidade);
4. ser utilizado em muitas situações, mesmo quando não totalmente preciso ou completo;
5. sirva de ajuda para superar seu próprio volume, auxiliando a limitar as várias possibilidades que em geral tem de ser consideradas.

2.1. Abordagens da Inteligência Artificial

Os programas desenvolvidos através da IA caracterizam-se pelo fato de objetivarem uma interação com o usuário ou seu ambiente (sistema) por meio da simulação de atitudes e reações humanas que envolvem a compreensão, análise, planejamento, tomada de decisão, aprendizado (DAZZY, 1999). Dependendo da abordagem de IA adotada, tanto o modo de manipular o conhecimento quanto como adquiri-lo, armazená-lo, empregá-lo diferem (BARRETO, 2001).

A linha simbólica segue a tradição lógica e tiveram em McCathy e Newell seus principais defensores. Inicialmente, a pesquisa em manipulação de símbolos se concentrou no desenvolvimento de formalismos gerais capazes de resolver qualquer tipo de problemas. Estes esforços iniciais ajudaram a estabelecer os fundamentos teóricos dos sistemas de símbolos e forneceram uma série de técnicas de programação voltadas à manipulação simbólica, por exemplo, as técnicas de busca heurística.

A Inteligência Artificial Simbólica ficou conhecida nas décadas de 60/70 pela aparição dos sistemas especialistas, como também da expansão do uso de linguagens favorecendo o paradigma de Programação em Lógica (*Prolog*) (GIANNESINI, *et. al.,* 1985), (SHERING & SHAPIRO, 1986). Nesta mesma época uns dos grandes desafios à "inteligência" dos computadores eram jogos, se destacando o jogo de xadrez. Os japoneses também entraram na corrida pela criação de ferramentas de "inteligência" artificial, e propuseram a criação dos computadores de 5ª Geração, onde os computadores inteligentes seriam capazes de escutar, falar e raciocinar (FEIGENBAUM & MCCORDUCK, 1983).

Segundo Simons (1988), a Abordagem Conexionista, também denominada de biológica ou ascendente, dá ênfase no modelo de funcionamento do cérebro, dos

neurônios e das conexões neurais. Os pioneiros dessa corrente foram McCulloch, Pitts, Hebb, Rosenblatt e Widrow. Em 1943 surgiu a representação e formalização matemática dos neurônios artificiais, que fez surgir os primeiros modelos de redes neurais artificiais.

A Inteligência Artificial Conexionista sofreu grande impacto quando os cientistas Marvin Minsky e Seymour Papert publicaram o livro *PERCEPTRONS*, no qual criticaram e sustentaram que os modelos das redes neurais não tinham sustentação matemática suficiente para que lhes fosse possível resolver problemas linearmente separáveis (MINSKY & PAPERT, 1969). Apesar das pesquisas nesta área não terem parado, foi apenas na década de 80 que o físico e biólogo do Instituto de Tecnologia da Califórnia, Jonh Hopfield conseguiu recuperar a credibilidade da utilização das redes neurais (SIMONS, 1988).

A Inteligência Artificial Evolutiva é um exemplo de solução do problema bem definido de sobrevivência de uma espécie em ambiente variável. Pode ser encarada como um método de otimização com restrições variáveis e muitas vezes desconhecido (BARRETO, 2001).

Os Sistemas Evolutivos tomando inspiração da Evolução Biológica foram inicialmente introduzidos em 1960, na Alemanha e hoje são conhecidos como Programação Evolutiva. Em meados dos anos 70, Holland (1975) estudou a possibilidade de incorporar os mecanismos naturais de seleção e sobrevivência para a resolução de problemas de IA e apresentou seu Algoritmo Genético tão popular atualmente. A partir dos anos 80, foi viabilizada a prática de sistemas com técnicas de Computação Evolutiva ou Inteligência Artificial Evolucionária. Em 1995, Fogel (1995) apresentou um modelo de Computação Evolutiva.

Este paradigma permite a resolução de problemas considerando várias soluções possíveis envolvendo os indivíduos da população, o problema a resolver e o ambiente. A adaptação seria então a qualidade da solução permitindo considerações sobre o quão inteligente seria uma dada solução comparada com as demais (FALQUETO, *et. al.,* 2001).

A diversidade é gerada por cruzamento e mutações. Os seres mais adaptados sobrevivem (seleção natural). Ou seja, têm maior chance de sobrevivência os mais aptos

e estes conseqüentemente, têm maior possibilidade de gerar descendentes, transmitindo suas características. As características genéticas são herdadas pelas próximas gerações. A Computação Evolutiva trata dos seguintes paradigmas atualmente (PALAZZO, 1997):

- Algoritmos Genéticos.
- Programação Evolutiva.
- Estratégias Evolutivas.
- Sistemas Classificadores.
- Programação Genética.

É possível dizer que todo sistema que integra duas ou mais técnicas diferentes para a solução de um problema é considerado um Sistema Híbrido. Ou seja, a IAH é a combinação de diferentes abordagens: IAS, IAC e IAE. A Inteligência Artificial Híbrida busca explorar as vantagens de cada técnica a fim de vencer as suas dificuldades e limitações. Pesquisas futuras nos levam em direção a sistemas com múltiplas formas de aquisição e representação de conhecimentos, onde sistemas híbridos e sistemas multi-agentes serão uma tendência. Estas possíveis combinações são mais eficientes em determinadas soluções de problemas, suprindo deficiências encontradas nas técnicas tradicionais (MEDEIROS & BARRETO, 2003).

Nas abordagens clássicas de IA, a ênfase da inteligência é baseada em um comportamento individual e o foco de atenção se volta à representação de conhecimento e métodos de inferência. Na Inteligência Artificial Distribuída, é baseada em comportamento social e sua ênfase é para cooperações, interações e para o fluxo de conhecimento entre unidades distintas (OLIVEIRA, 1996).

A IAD se concentra no comportamento inteligente que emerge como produto da cooperação de diversas entidades conhecida como agentes, ou seja, entidades computacionais que interagem entre si. Um sistema em IAD é formado pelos seguintes elementos (ALVARES & SICHMAN, 1997):

- Agentes: entidades ativas do sistema.
- Sociedade: conjunto formado pelos agentes.
- Ambiente: conjunto formado pelas entidades passivas (objetos) do sistema.

- Interação: trocas de informações entre os agentes da sociedade.
- Organização: restrições sociais aplicadas aos agentes; isto garante que os agentes farão aquilo para o qual foram construídos.

Pode-se distinguir duas áreas principais que estão relacionadas (FININ, 1993):

- Resolução Cooperativa e Distribuída de Problemas (RCDP): estuda os métodos de como dividir uma tarefa em vários módulos, que cooperam e partilham o conhecimento acerca do problema e da sua solução.
- Sistemas Multi-Agentes (SMA): modelagem do comportamento de agentes computacionais, "dotados" de inteligência e coordenados entre si, os quais realizam atividades especificas de forma coletiva.

Os agentes podem ser definidos como entidades que emergem num ambiente onde podem agir intencionalmente, com habilidades de: perceber, representar, comunicar com outros agentes autônomos (CASTELFRANCHI, 1990). Agente é algo que obtém o conhecimento do seu ambiente através de sensores e atua nesse ambiente (RUSSEL & NORVIG, 1995).

Agentes podem possuir uma série de propriedades, não necessariamente todas (FERNANDEZ, 1998):

- Autonomia: pode interagir sem a intervenção de humanos.
- Sociabilidade: é capaz de interagir por meio de uma linguagem de comunicação entre agentes.
- Reatividade: é capaz de perceber e reagir a estímulos.
- Iniciativa: pode atuar guiado por objetivos.
- Continuidade: está continuamente ativo.
- Orientação por objetivos: é capaz de resolver tarefas complexas.
- Mobilidade: capacidade de um agente de locomover-se através de uma rede de computadores.
- Adaptabilidade: deve adaptar-se aos hábitos e métodos de trabalho.
- Interoperabilidade: é capaz de integrar duas aplicações executando em plataformas diferentes.

Entre os campos de aplicações de técnicas da IA destacam-se os sistemas a seguir.

2.2. Sistemas Especialistas

Trata-se de programas baseados em conhecimento podendo ter várias finalidades tais como, assistente, crítico, consultor e tutor. Tal conhecimento deve ser obtido de especialistas, e seu comportamento deve imitá-los. Por ter um comportamento semelhante ao de um especialista humano, supostamente inteligente, ele é considerado um programa inteligente sendo geralmente capaz de aprender e tomar decisões (MEDEIROS & BARRETO, 2003).

A importância tecnológica e econômica de um Sistema Especialista produz resultados consistentes desde que seja amparado por especialistas humanos, onde a máquina resolve problemas para cada área específica, utilizando sua base de conhecimento.

Sistemas Especialistas possuem capacidade de acumular conhecimentos, e usar tais conhecimentos para resolver problemas que necessitariam de um especialista. Podem ter várias finalidades como: analisar, concluir, deduzir, induzir, diagnosticar, prever, monitorar, planejar, projetar, interagir e tomar decisões que "melhor lhe convier" baseado em sua base de conhecimento desenvolvido por especialistas, tornando-se independente do especialista no momento de sua execução.

Um sistema especialista é um programa de computador que resolve uma tarefa que requer grande conhecimento e alta capacidade intelectual quando um ser humano a resolve (FENSEL & HARMELEN, 1994), (FRIEDERICH & GARDANO, 1989), (KELLER, 1991).

Grande parte do esforço de desenvolver um sistema especialista se encontra na elicitação do conhecimento, ou seja, como capturar e utilizar o conhecimento de um ser humano em uma aplicação computacional (SILVA, *et. al.,* 2002). Elicitação de conhecimento é um conjunto de atividades realizadas por um especialista para obter material de alguma fonte relevante, analisar e interpretar o material, e colocá-lo sob uma forma pré-codificada (CUNHA, 1995).

Essa é uma tarefa realmente importante, mas para que ela resulte em um bom sistema especialista, o mesmo deve ser desenvolvido utilizando técnicas que considerem todo o universo que o cerca, desde o início do projeto até a morte do programa. Estas

técnicas envolvem o ciclo de vida de um programa, aumentam a sua qualidade e facilitam a sua manutenção (SILVA, *et. al.,* 2002).

Um SE tradicional ao ser construído, conduz a vários problemas. O primeiro deles é o processo de Aquisição de Conhecimento (AC). De fato, a elicitação de conhecimento de um especialista de domínio – a qual é um dos estágios da AC – é uma tarefa árdua. Ela consome um tempo razoável, não só do Engenheiro do Conhecimento como também do especialista em questão (BRASIL, 1996). O acúmulo de conhecimento é o aspecto mais importante de um SE (WATERMAN, 1986).

De acordo com Barreto (2001) de um modo geral, para construir um SE, necessita-se:

- De uma fonte de conhecimento: o especialista.
- Conhecimento deve ser obtido do especialista, transformado em forma conveniente e armazenado no computador.
- Este conhecimento fundamentalmente é de dois tipos: fatos sobre o problema a resolver e regras que mostram como o especialista raciocina pra chegar a uma conclusão. São as regras de raciocínio.
- Freqüentemente é ainda necessário dispor de um mecanismo capaz de gerar explicações sobre como o especialista chegou a uma determinada conclusão. Isto é motivado por casos em que o usuário do sistema não concorda totalmente com a sugestão do sistema especialista: ele quer ver qual o raciocínio que foi seguido para se convencer que o SE tinha razão.

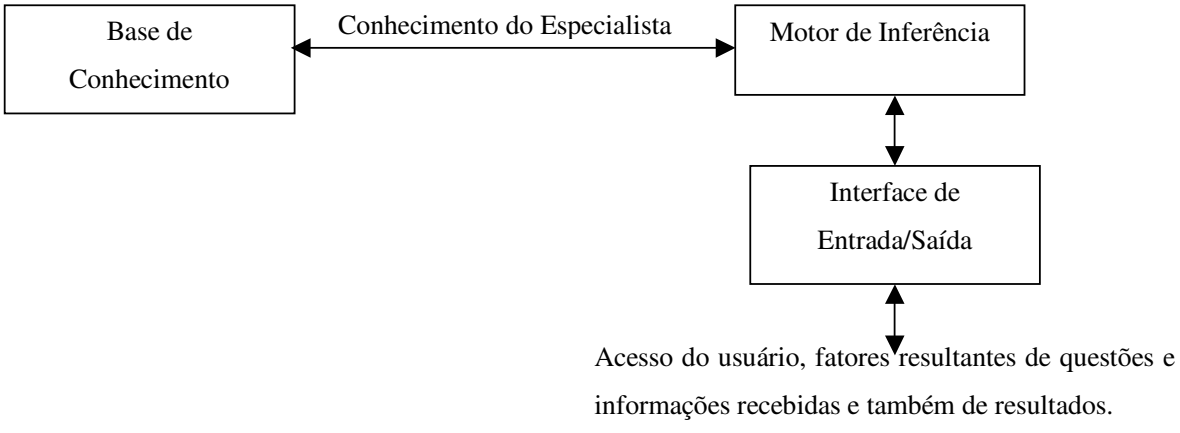

Figura 1: Estrutura de um Sistema Especialista (HART, 1992).

O modelo de elementos básicos é mostrado na Fig. 1: a base de conhecimento, o mecanismo de inferência que manipula o conhecimento, a interface para o usuário que permite acessar o conhecimento.

Base de Conhecimento: permite uma representação para o conhecimento que é requerido. Em geral, são compostas por regras de domínio, fatos e valores atribuídos a cada regra ou fato das bases. Junta-se ao conjunto, de acordo com o modelo: heurísticas e regras que podem levar a melhor caracterização do sistema.

O mecanismo de inferência: meios para que o conhecimento seja manipulado. É uma estratégia de busca sobre determinada base de conhecimento baseada em regras de produção.

A interface de entrada e saída: habilita o usuário para acessar fatos e dados, e habilita o sistema para perguntar questões ou acesso de conselho e ou explicação. É necessário ter cuidado com a forma de linguagem e apresentação, com o objetivo de propiciar facilidade de explicação.

A base de conhecimento é a estrutura de dados onde os fatos e as regras de produção estão armazenadas (PELLEGRINI, 1995). Um fato importante nos SEs é que o raciocínio seguido deve ser o mais transparente possível e capaz de explicar a qualquer momento porque usou determinado procedimento.

Inicialmente os Sistemas Especialistas de 1ª Geração, apresentaram problemas de aquisição de conhecimentos (não era automática e dependia do especialista e/ou Engenheiro do Conhecimento para que fosse adicionado novo conhecimento ao sistema). Surgia assim o conhecido gargalo da aquisição de conhecimento dos sistemas especialistas (CHORAFAS, 1988). Alguns exemplos notáveis destes sistemas são: *Dendral (*identificação química), *Hearsay I e II* (reconhecimento de fala), *HELP* (gerencia admissões e altas)*, ILIAD* (complemento do *HELP*) e *DELTA/CATS (*manutenção de locomotivas).

Os sistemas especialistas de 2ª Geração introduziram a aquisição automática de conhecimentos, e então começamos a ouvir falar de aprendizado de máquinas simbólicas. O auge dos sistemas simbólicos foi atingido em torno da década de 1980, com os programas que dispunham de uma "base de conhecimento" no qual estavam codificadas regras conhecidas como "regras de produção". Alguns exemplos notáveis

destes sistemas são: *Mycin* (diagnóstico de infecção no sangue), o *Prospector* (recomendação de exploração geológica) e o *LUNAR* (interface para geólogos interrogarem sobre as mostras de rochas trazidas pela *Appolo* na missão lunar - o primeiro usado por pessoas que não eram os projetistas do sistema).

2.3. Redes Neurais Artificiais

Redes neurais artificiais (RNAs) depois de um início em paralelo com métodos simbólicos ficou no esquecimento durante vários anos despertando no principio dos anos 80, inicialmente explorados com "alguma" curiosidade em que o primeiro Sistema Especialista Conexionista, desenvolvido por Gallant (1988) começou a se tornar popular.

Redes neurais artificiais (RNAs) possuem a capacidade de aprendizado e a adaptação do comportamento, a partir de um conjunto de exemplos, também considerado como conjunto de treinamento. O conhecimento é armazenado nas conexões sinápticas. A informação é representada e processada paralelamente por neurônios que distribuídos no espaço e ligados através de conexões (sinapses) trocam sinais inibitórios ou excitatórios cooperando entre si em busca de objetivos.

A principal finalidade é de que dado um determinado conjunto de entradas, se produza um conjunto esperado na saída. Uma maneira é treinar a rede através de um algoritmo de treinamento, que irá automaticamente ajustar os pesos da rede. Para esse treinamento, existem basicamente duas técnicas, a do aprendizado supervisionado e aprendizado não supervisionado. O que distingue as duas técnicas é a presença ou não de um professor ou supervisor (HAYKIN, 2001). O professor pode ser um conjunto de dados de treinamento, ou um observador que qualifica o desempenho (DAZZI, 1999).

As RNAs possuem características tais como: adaptação e aprendizado de comportamento, correção dos erros, otimização de performance, interação com o meio, experimentação e descoberta, memória e compreensão dos conhecimentos (MEDEIROS & BARRETO, 2003). As redes neurais artificiais podem ser subdivididas em redes diretas e redes com ciclos ou recorrentes. As redes diretas são aquelas que o fluxo de informação segue em uma única direção, enquanto que, as redes com ciclos possuem topologia apresentando ao menos um percurso fechado (AZEVEDO, *et. al.*, 2000).

A topologia determina o tipo de processamento causando um enorme efeito na operação de uma RNA (ALEKSANDER & MORTON, 1990). A Tabela 1 apresenta um quadro comparativo entre as redes diretas e redes recorrentes (FRANCESCHI & BARRETO, 1999).

Tabela 1: Principais diferenças entre RNAs Diretas e Recorrentes.

Características	**RNAs Diretas**	**RNAs Recorrentes**
Quanto ao estado	Estática – não existe troca de estado.	Dinâmicas – existe troca de estados conforme as entradas.
Quanto à topologia	Não possui ciclos. O fluxo de sinais da rede possui apenas uma direção, da entrada para saída.	Possui ciclos com realimentação. A saída de uma das camadas pode realimentar a entrada de dados (o fluxo pode ser da entrada para a saída, das camadas intermediárias para a entrada, ou da camada de saída para a entrada).
Quanto ao número de camadas	Para solucionar um problema: **LINEAR:** duas camadas uma de entrada e uma de saída. **NÃO LINEAR:** tem que ter no mínimo uma camada intermediária.	Como existem ciclos com realimentação, pode haver ou não camada intermediária, dependendo do caso.
Quanto à construção	**Treinadas** ou de forma **supervisionada** (como é o caso do algoritmo de treinamento *Backpropagation*) ou **não-supervisionada** (como é o caso dos mapas de *Kohonen*).	**Construídas:** exemplos de redes neurais construídas são BAM (Bidirectional Associative Memory) - Kosko e Hopfield. **Treinadas:** normalmente são mais complicadas, porque o ajuste dos pesos deve ser feito nos dois sentidos, para frente e para trás. Exemplo deste tipo de rede ART (Adaptive Resonance Theory – Grossberg & Carpenter.

Fonte: FRANCESCHI, A.S.M., BARRETO, J.M. Desenvolvimento de Agentes Autônomos para Gerência de Redes de Computadores. Tese de Doutorado (CPEEL, UFSC), Florianópolis, 1999.

No próximo capítulo será abordado um estudo sobre a engenharia de programas e as metodologias existentes com a finalidade de ter subsídios para a escolha do modelo a ser utilizado.

3. MÉTODOS E TÉCNICAS DE DESENVOLVIMENTO DE PROGRAMAS

Apesar das metodologias serem largamente exploradas em livros, a maioria das metodologias aqui mencionadas se concentram apenas em método algorítmico. É necessário tecer estas considerações, uma vez que, alguns aspectos relevantes ao estudo são fundamentais. O leitor interessado em aprofundar o assunto deve consultar bibliografia adequada, por exemplo: (BOEHM, 1986), (BOOCH, *et. al.*, 2000), (BRUSSO, *et. al.*, 2001), (FURLAN, 1998), (LARMAN, 2000), (PETERS & PEDRYCZ, 2001), (PRESSMAN, 2002), (ROCHA, *et. al.*, 2001), (SOMMERVILLE, 2003), (YOURDON, 1989).

3.1. Histórico da Engenharia de Programas

No desenvolvimento de sistemas complexos, uma abordagem sistemática deve ser adotada. Várias abordagens têm sido propostas, todas objetivando a produção de bons sistemas. Mas o que é um bom sistema? Segundo Jacobson (1992), um ponto de vista tanto da parte do usuário como do desenvolvedor deve ser observado: que um bom sistema deve ser correto, rápido, confiável, fácil de ser usado, eficiente.

As atividades e informações que são evidenciados pelo ciclo de vida envolvidos no processo de desenvolvimento de programas utilizam as linguagens de representação cuja eficácia já foi comprovada na área de Engenharia de Programas. As habilidades desenvolvidas podem ser um dos melhores meios de simplificar a tarefa de aquisição do conhecimento. Quando se tem uma estrutura adequada para construir um programa fica mais fácil a compreensão dos problemas antes do término do sistema, tornando o desenvolvimento um projeto sólido.

A engenharia de programas ainda é uma disciplina relativamente nova. A noção de “engenharia de programas” surgiu pela primeira vez em 1968, em uma conferência organizada para discutir a chamada “crise de software”. Essa crise resultava diretamente da introdução (naquela época) do poderoso *hardware* de computador de terceira

geração. Sua capacidade tornava viáveis aplicações de computador até então inimagináveis. O *software* resultante era bem maior e mais complexo que os sistemas anteriores (SOMMERVILLE, 2003).

A engenharia de programas integra métodos, ferramentas e procedimentos para o desenvolvimento de *software* de computadores. Uma série de diferentes paradigmas foi proposta, cada uma exibindo potencialidades e fragilidades, mas todos tendo uma série de fases genéricas em comum (PRESSMAN, 1995). É caracterizada por um desenvolvimento de programa prático, ordenado e medido. O principal objetivo é produzir sistemas satisfatórios que respeitem prazos e orçamentos (PETERS & PEDRYCZ, 2001).

A experiência inicial de construção de sistemas mostrou que a abordagem informal do desenvolvimento de *software* não era o bastante. Projetos importantes sofriam atrasos às vezes, de alguns anos. Por apresentarem custos muitos maiores do que os inicialmente previstos, eles não eram confiáveis, eram de difícil manutenção e tinham desempenho inferior. Os custos de *hardware* caiam, enquanto os de *software* subiam rapidamente. Novas técnicas e novos métodos eram necessários para controlar a complexidade inerente aos grandes sistemas de *software* (SOMMERVILLE, 2003).

Visando melhorar a qualidade dos produtos de "*software*" e aumentar a produtividade, surgiu a área de pesquisa denominada Engenharia de Programas. Busca organizar esforços no desenvolvimento de ferramentas, metodologias e ambientes de suporte ao desenvolvimento de sistemas (FALBO & MENEZES, 1998).

O gerenciamento de projetos de *software* também é uma tarefa de fundamental importância. Não é visto como uma etapa clássica do processo, uma vez que ele acompanha a todas as etapas de concepção à obtenção do sistema (PRESSMAN, 1995), (YOURDON, 1989). A crescente demanda relacionada ao *software* tem ressaltado uma série de problemas relacionados ao processo de desenvolvimento de sistemas, dentre eles estão: alto custo, complexidade, dificuldade na manutenção (DEMARCO, 1995).

O Engenheiro do Conhecimento é uma figura central, tanto na construção do sistema, vista sob o aspecto técnico, como na aquisição do conhecimento. Pessoa de tal importância deve ter um preparo especial e até algumas qualidades inatas (ROLANDI,

1986). O Engenheiro do Conhecimento deve ter uma educação ampla e ser, de forma geral, bem informado (REITMAN, 1989), conforme explanação a seguir.

3.2. Engenheiro do Conhecimento

O Engenheiro do Conhecimento é o profissional responsável pela estruturação e construção de um sistema. Ele extrai conhecimento de alguma fonte, interpreta e representa em tipos e estruturas convenientes.

Algumas habilidades necessárias para um Engenheiro do Conhecimento são: boa comunicação, inteligência, tática, diplomacia, empatia, paciência, versatilidade, inventividade, conhecimento de domínio e programação, dentre outros (OSHIRO, *et. al.*, 2001).

Não existe uma regra ou um ciclo de vida padrão para o desenvolvimento de sistema, se pode usar um ciclo pré-definido por um determinado autor e moldá-lo conforme seu trabalho ou ainda criar seu próprio ciclo. É de responsabilidade do Engenheiro do Conhecimento definir qual a metodologia mais indicada para o desenvolvimento.

3.3. Aquisição do Conhecimento

Aquisição de conhecimento tenta responder às seguintes questões: como introduzir o conhecimento na máquina? Como tratar consistência e redundância?

A psicologia tem um papel fundamental no processo de aquisição do conhecimento. A forma de tratamento, os aspectos psicológicos, como comportamento, comunicação e percepção, determina a qualidade da interação entre o especialista e o Engenheiro do Conhecimento (DUARTE & FALBO, 2000), (GAMMACK & YOUNG, 1985).

Na Filosofia, ontologias são usadas com o intuito de desvendar o significado das coisas no mundo, procurando descrever a natureza das coisas. Na IA, por sua vez, utilizam-se ontologias para descrever domínios já consagrados, como Medicina, Engenharia e Direito, onde é possível saber o significado projetado das coisas. Assim, o que se busca com uma ontologia em IA é firmar um acordo sobre o vocabulário do

domínio de interesse, a ser partilhado por agentes que conversam sobre ele (FALBO & TRAVASSOS, 1995), (FALBO & MENEZES, 1998), (FALBO, *et. al.*, 1999).

Ontologias são descrições dos conteúdos do conhecimento do domínio, podendo ser construídos e usados para melhorar o processo da engenharia de conhecimento. A escolha da ontologia é usada para organizar a base de conhecimento e também aumentar o reuso do programa. Pode-se dizer que ontologia é um vocabulário de conceitos, relações e regras.

Pode-se dizer que aquisição de conhecimento é um processo de extração, transformação e transferência de informação de uma fonte de conhecimento para um programa de computador. É um estágio fundamental no desenvolvimento de um sistema inteligente, e também o mais problemático (OSHIRO, *et. al.,* 2001).

A técnica "entrevista estruturada" ou seja, aquisição de conhecimento explícito é uma das técnicas mais usadas para elicitar o conhecimento de um Especialista do Domínio (GAMMACK & YOUNG, 1985), (MCGRAW & BRIGGS, 1989). Um exemplo de um formulário de aquisição de conhecimento é mostrado na Fig. 2:

Formulário de Aquisição de Conhecimento

Sessão de AC: Engenheiro de Conhecimento:

Tópico da sessão: Data de sessão de AC:

Local da sessão: Tempo total:

Esp Dom/Fonte de Conhecimento:

Tipo de sessão: [ ] Entrevista [ ] Rastreamento de processos [ ] Simulação

[ ] Estudo de casos [ ] Análise de tarefas [ ] Outros .....

Principais metas da sessão:

Resumo da sessão:

Figura 2: Formulário de Aquisição de Conhecimento (MCGRAW & BRIGGS, 1989).

Para solucionar os problemas encontrados na Inteligência Artificial, é preciso uma grande quantidade de conhecimento e também alguns mecanismos para a manipulação desse conhecimento, a fim de criar soluções para novos problemas (RICH & KNIGHT, 1993).

A aquisição de conhecimento é, sem dúvida, o aspecto crucial no desenvolvimento de sistemas e, portanto, faz-se necessário o uso de uma abordagem apoiada em alguma forma de reuso (FALBO & TRAVASSOS, 1995).

3.4. Representação do Conhecimento

Segundo Barreto (2001), representação do conhecimento tenta responder às seguintes questões: como o conhecimento é armazenado na máquina? Qual a capacidade do armazenamento de conhecimentos de uma certa máquina? Vários autores dividem os paradigmas de representação de conhecimento em quatro grandes grupos:

- Regras de produção.
- Redes semânticas.
- Molduras.
- Roteiros.
- Lógicas de primeira ordem.

É impossível representar o mundo real, ou mesmo uma parte dele, em sua completa riqueza de detalhes. Para representar um certo fenômeno ou uma porção do mundo, a que chamamos domínio, é necessário concentrar a atenção em um número limitado de conceitos, suficientes e relevantes para se criar uma abstração do fenômeno em estudo (FALBO & TRAVASSOS, 1995).

Desta forma, várias conceituações diferentes são possíveis para um mesmo domínio e, para capturar a conceituação de uma porção do mundo, é necessário utilizar alguma representação (FALBO & TRAVASSOS, 1995). Quanto melhor e mais perfeita a representação, do problema, da base de dados, e da solução, melhor o programa compreenderá o que se pretende dele.

3.5. Ciclo de Vida de Sistemas

Ciclo de vida de sistemas pode ser considerado como uma representação de conhecimento desde a idéia de sua concepção, construção, implementação, implantação, maturidade, declínio, manutenção, uso; após sua entrada em operação até que se finaliza quando o sistema não está mais disponível para uso.

As origens dos ciclos de vida devem-se ao ambiente de desenvolvimento tais como Linguagem de 3ª e 4ª Geração, tipo de programa comercial, industrial, técnico-científico, monitoração, controle e simulação de processos, aplicativo, básico, utilitário, tutor, dentre outros. O ciclo de vida define as fases pelas quais passa o sistema, bem como seu relacionamento entre elas.

O modelo do ciclo de vida do programa é um importante componente do processo de desenvolvimento de aplicações. Compreende a análise do problema e definição dos requisitos, o desenho da solução, a codificação, os testes e a manutenção, bem como as atividades, métodos e ferramentas necessárias para o desenvolvimento. Segundo Pressman (1995) ciclo de vida de *software* é hoje considerado como "modelo de processo de *software*".

Um amplo conjunto de tarefas que abrange análises de requisitos, projeto, construção de programas, teste e manutenção e ferramentas em conjunto com as fases formam o modelo de processo de *software* (PRESSMAN, 2002). O ciclo de vida representa as diversas etapas pelas quais passa um projeto de desenvolvimento tais como: crescimento, evolução e morte.

E para auxiliar o desenvolvimento do sistema existem vários modelos de ciclo de vida descritos na literatura, tais como: Modelo Cascata, Prototipação, Espiral e Iterativo Incremental (FALBO & TRAVASSOS, 1995).

3.5.1. Modelo Cascata

O modelo de ciclo de vida clássico foi proposto por Royce em 1970. Às vezes é chamado modelo cascata e requer uma abordagem sistemática, seqüencial ao desenvolvimento do *software*, que se inicia no nível do sistema e avança ao longo da análise, projeto, codificação, teste e manutenção.

É modelado em função do ciclo da engenharia convencional. O paradigma do ciclo de vida abrange as seguintes atividades: Análise e Engenharia de Sistemas, Análise de Requisitos de *Software*, Projeto, Codificação, Testes e Manutenção (PRESSMAN, 1995).

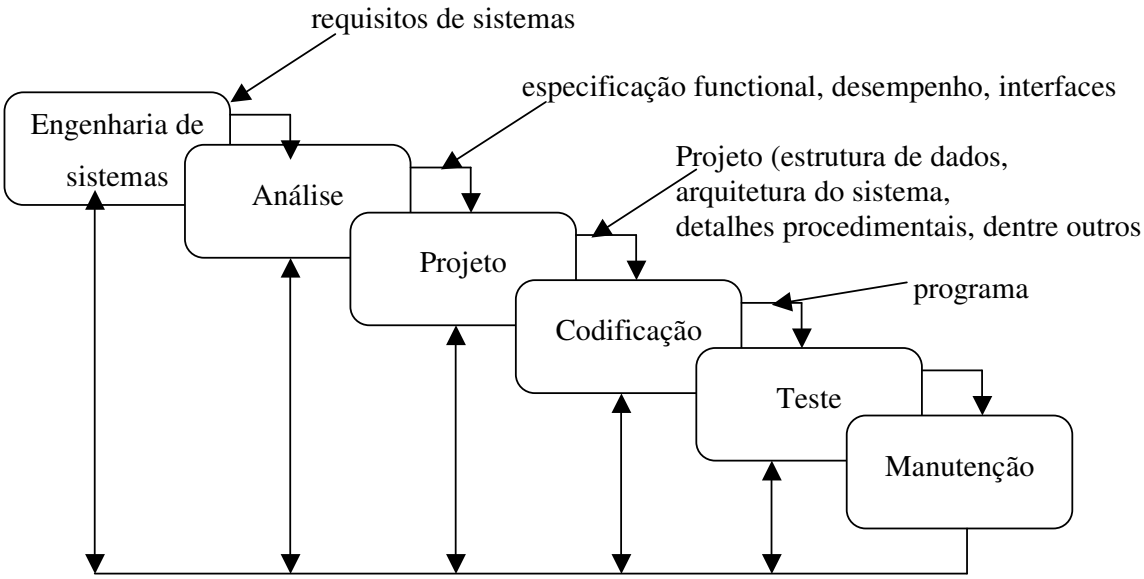

Figura 3: O ciclo de vida clássico (PRESSMAN, 1995).

As fases são executadas seqüencialmente, conforme Fig. 3, cada fase é executada uma vez, embora um retorno à fase anterior seja permitido para correção de erros. A entrega do sistema completo ocorre ao final da fase de Testes. É um modelo muito utilizado e eficiente para problemas pequenos e bem definidos. Mas é questionável sua eficácia para aplicações maiores e complexas.

O modelo de ciclo de vida clássico continua sendo o modelo procedimental mais amplamente usado pela engenharia de *software*. Embora tenha fragilidades, ele é significativamente melhor do que uma abordagem casual ao desenvolvimento de *software* (PRESSMAN, 1995). As características principais de um modelo em cascata são: fases bem definidas e bem delimitadas; validação e verificação somente no final do processo e fortemente documental (ROYCE, 1970).

Outra característica singular de um ciclo de vida de desenvolvimento em cascata é que ele envolve único passo de Análise, de Projeto e de Construção. O desenvolvedor faz toda a análise, todo o projeto, a seguir, a codificação e, finalmente, todos os testes. Entre algumas de suas falhas temos (LARMAN, 2000):

- sobrecarga dos passos iniciais da complexidade que deve ser atacada;
- “feedback” (retorno, realimentação) retardado;

- especificações congeladas prematuramente, enquanto o ambiente de negócios muda.

3.5.2. Prototipação

A prototipação é um processo que capacita o desenvolvedor a criar um modelo do *software* que será implementado. O modelo pode assumir uma das três formas (PRESSMAN, 1995):

1. retrata a interação homem-máquina de uma forma que capacita o usuário a entender quanta interação ocorrerá;
2. um protótipo de trabalho que implementa algum subconjunto da função exigida do *software* desejado; ou
3. um programa existente que executa parte ou toda a função desejada, mas que tem outras características que serão melhoradas em um novo esforço de desenvolvimento.

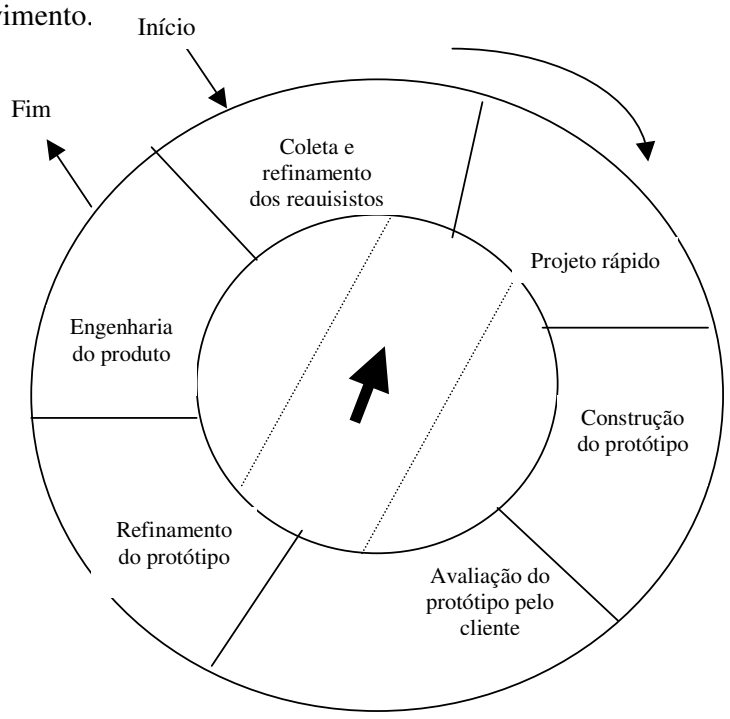

Figura 4: Prototipação (PRESSMAN, 1995).

Idealmente, o protótipo serve como um mecanismo para identificar os requisitos de *software* conforme Fig. 4. Se um protótipo de trabalho for construído, o desenvolvedor tentará usar fragmentos de programas existentes ou aplicará ferramentas

(por exemplo, geradores de relatórios, gerenciadores de janelas, dentre outros) que possibilitem que programas de trabalho sejam gerados rapidamente (PRESSMAN, 1995).

Prototipação é uma boa ferramenta para tratar riscos. Se o risco for considerado inaceitável, pode parar o projeto. A prototipação permite detectar problemas e assim reduzir custos e melhorar a qualidade do sistema.

3.5.3. Modelo Espiral

O modelo espiral foi proposto por Boehm em 1986 e desenvolvido para abranger as melhores características tanto do ciclo de vida clássico como da prototipação, acrescentando ao mesmo tempo, um novo elemento – a análise de riscos – que falta a esses paradigmas (PRESSMAN, 1995).

É capaz de descrever como um produto se desenvolve para formar novas versões e como uma versão pode ser incrementalmente desenvolvida, partindo-se de um protótipo até um produto completo (JACOBSON, 1992).

Uma maneira simplista de analisar este modelo é considerá-lo como um modelo cascata onde cada fase é precedida por uma análise de risco e sua execução é feita evolucionariamente ou incrementalmente. Com cada iteração ao redor da espiral (iniciando-se ao centro e avançando para fora), versões progressivamente mais completas do *software* são construídas (PRESSMAN, 1995). Cada setor da espiral corresponde a um estágio do desenvolvimento conforme mostra a Fig. 5.

No primeiro ciclo se inicia com a coleta inicial de requisitos e planejamento do projeto, onde ocorre o comprometimento dos envolvidos e o estabelecimento de uma estratégia para alcançar os objetivos. No segundo ciclo realiza-se a avaliação de alternativas, identificação e executa-se uma análise de risco. No terceiro ciclo ocorre o desenvolvimento do produto. Neste quadrante pode-se considerar o modelo cascata (BRUSSO, *et. al.,* 2001).

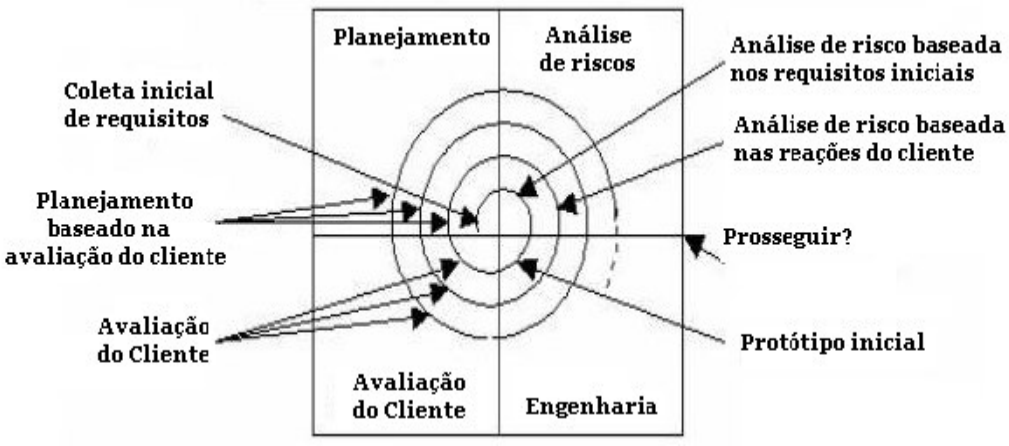

Figura 5: O modelo Espiral Completo (BRUSSO, *et. al.*, 2001).

Durante o primeiro giro ao redor da espiral, os objetivos, alternativas e restrições são definidos e os riscos são identificados e analisados. Se a análise dos riscos indicar que há incertezas nos requisitos, a prototipação pode ser usada no quadrante da engenharia para ajudar tanto o desenvolvedor quanto o cliente na decisão de prosseguir ou não. O paradigma do modelo espiral para a engenharia de programas atualmente é a abordagem mais realística para o desenvolvimento de *software* em grande escala. O modelo espiral usa a prototipação como um mecanismo de redução de riscos, mas, o que é mais importante, possibilita que o desenvolvedor aplique a abordagem de prototipação em qualquer etapa da evolução do produto (PRESSMAN, 1995).

3.5.4. Iterativo Incremental

Uma definição do desenvolvimento iterativo é que o modelo é um processo planejado para voltar a uma área repetidamente, cada vez complementando o sistema. A definição do desenvolvimento incremental é adicionar funcionalidades a um sistema durante vários ciclos de liberação de produto. Uma versão incremental é composta de múltiplos ciclos de desenvolvimento iterativos.

O modelo Iterativo Incremental procura usar a flexibilidade e modularidade da orientação a objetos, sendo que a divisão em fases e atividades, tem-se um maior controle sobre custos e riscos. A identificação é realizada no começo do ciclo de vida (LARMAN, 2000). Apresenta um ciclo de vida que consiste de várias iterações, cada

iteração incorpora atividades de modelagem. Baseia-se no aumento e no refinamento sucessivo de um sistema através de: requisitos, análise e projeto, implementação, testes e instalação.

O autor Boehm (1986) defende a questão que o processo de *software* é muito mais iterativo e cíclico, sugerindo os processos em ciclos contínuos e repetidos, desenvolvidos a cada ciclo. Um processo é aquele que envolve o gerenciamento de seqüências de versões executáveis. Um processo incremental é aquele que envolve a integração contínua da arquitetura do sistema para a produção dessas versões, de maneira que cada nova versão incorpora os aprimoramentos incrementais em relação às demais como mostra a Fig. 6. Em conjunto, um processo iterativo e incremental é orientado a riscos, ou seja, cada nova versão tem como foco atacar e reduzir os riscos mais significativos para o sucesso do projeto (BOOCH, *et. al.,* 2000).

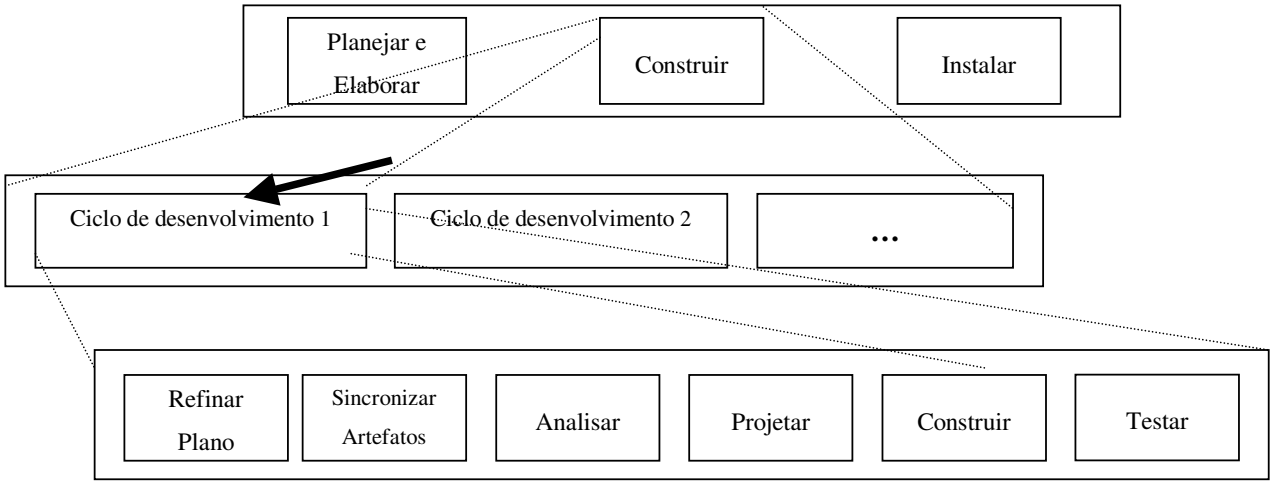

Figura 6: Ciclo de Vida Iterativo (LARMAN, 2000).

3.6. Breve Comparativo dos Modelos

O modelo em cascata tem a desvantagem do alto custo de correção das especificações quando atinge as fases de Teste e Implantação. O modelo de prototipação se baseia na utilização de um protótipo do sistema real. Serve para auxiliar na determinação de requisitos. Um protótipo deve ser de baixo custo e de rápida obtenção, para que possa ser avaliado.

Para isto, uma determinada parte do sistema é desenvolvida com o mínimo de investimento, mas sem perder as características básicas, para ser analisada juntamente com o usuário.

A prototipação permite que se descubram os problemas antes de se comprometer o sistema. Através da prototipação busca-se entender as necessidades do cliente, orientar o desenvolvedor e permitir o aumento do interesse do usuário final com sua participação.

Uma das vantagens em usar prototipação são as iterações constantes. Mas, a ênfase dada é para a codificação e não para a especificação, há um envolvimento contínuo do cliente. Uma grande vantagem é o enfoque de reutilização. As desvantagens são problemas com riscos, prazos e orçamento.

O modelo espiral tenta abranger as melhores características tanto do ciclo de vida clássico como a prototipação, acrescentando uma nova fase (a análise de riscos) que faltam nestes paradigmas. A desvantagem é o alto custo para adoção, envolve muitos estágios intermediários refletindo os problemas na documentação e treinamento. Gasta-se muito tempo nas fases 1, 2 e 4 conforme Fig. 5, podendo em alguns casos, não convergir para uma solução (LARMAN, 2000).

O ciclo de vida iterativo está baseado no aperfeiçoamento sucessivo de um sistema, através de múltiplos ciclos de análise, de projeto e de construção. Estas diferentes possibilidades de responsabilidades pelo desenvolvimento têm impacto na definição de atividades de gerência e controle da qualidade, além de revelar novos papéis de recursos humanos no contexto do desenvolvimento.

3.7. Metodologias para desenvolvimento em IA

Este capítulo aborda um estudo sobre as metodologias existentes com a finalidade de ter subsídios para a escolha do modelo a ser utilizado. É necessário tecer estas considerações, uma vez que, alguns aspectos relevantes ao estudo são fundamentais. O leitor interessado em aprofundar o assunto deve consultar bibliografia adequada, por exemplo: (ANGELE, *et. al.* 1993), (ECK, *et. al.,* 2001), (FENSEL & HARMELEN, 1994), (FININ, 1993), (GAMMACK & YOUNG, 1985), (HART, 1992), (HICKMAN,

et. al., 1989), (KELLNER, 1989), (MUSEN, *et. al.*, 1995), (ROLANDI, 1986), (STYLIANOU, *et. al.*, 1992), (WIELINGA, *et. al.*, 1993).

Dentre as metodologias existentes destacam-se as seguintes: *KADS* (*Knowledge Acquisition and Design Support*), *MIKE* (*Model-Based and Incremental Knowledge Engineering*), *KOD* (*Knowledge Oriented Design*), *Vital, Desire, MKSM* (*Methodology for Knowledge System Management*), *CYGMA* (*Cycle de vie e Gestion des Métiers et des Applications*). Será discutido e comparado três enfoques diferentes de abordagens de metodologias a seguir: *KADS, MIKE* e *PROTÉGÉ.*

A dimensão final desta comparação considera tais metodologias com conceitos diferentes de reuso que é a base das metodologias envolvidas. Apesar destas serem largamente usadas, a maioria das abordagens aqui mencionadas se concentraram apenas em método simbólico, pois na época o simbolismo era a inovação da IA e hoje temos novos paradigmas a serem tratados, sendo esta uma característica importante para os sistemas atuais.

3.7.1. Projeto KADS

No início do projeto o nome da metodologia foi um acrônimo. Hoje o *KADS* é considerado uma metodologia para desenvolvimento de sistemas, influente em todo os mundos por profissionais, acadêmicos e industriais. O *KADS* prescreve fases, atividades e modelos documentados, provê técnicas, métricas de projeto e demonstração de procedimentos de qualidade no desenvolvimento de sistemas (MEDEIROS & BARRETO, 2003).

A história do *KADS* iniciou-se em 1982, com a necessidade de um processo mais estruturado para desenvolver programas do que as aproximações de prototipagens rápidas que prevalecia naquela época. Em parceria com a equipe da *University of Amsterdam*, o Kno*wledge-Based Systems Centre* e a *Polytechnic of the South Bank*, foi submetido uma proposta para o primeiro ciclo do programa do *ESPRIT* (E*uropean Strategic Program in Research in Information Tecnology*), um projeto piloto chamado Projeto 12, sendo lançado em julho de 1983 *(*HICKMAN, *et. al.* 1989*).*

Os resultados principais do projeto no primeiro ano foram limitados, mas resultou em uma análise sistemática de técnicas da elicitação do conhecimento. Em 1985, foi

proposto um projeto muito maior com o nome de *ESPRIT* 1098 conduzido em duas fases com mais cinco parcerias na segunda fase: a "*K.B.S.C*." mais tarde de *Knowledge-Based Systems Centre of Touche Ross Management Consultants*, *STC Tecnology Ltd*, *SCICON Ltd*, *NTE NeuTech Gmbh*, *SCS Gmbh*, e a *Cap Sogeti Innovation* com novo nome *Cap Sesa Innovation*.

O projeto foi dividido em três partes: desenvolvimento da teoria, desenvolvimento das ferramentas e teste experimental da metodologia emergente com os estudos de casos práticos.

Segundo Hickman (1989), no *KADS* são construídos modelos de análise, consistindo em um número específico de fases. O desenvolvimento está dividido em conjunto de fases pré-determinadas na ordem da execução. Estão incluídas sete fases: análise, projeto, refinamento do conhecimento, implementação, instalação, uso e manutenção.

Dentro de cada fase está um conjunto distinto de atividades que precisam ser executados como mostra a Fig. 7. Esta divisão do desenvolvimento em fases e atividades facilita o gerenciamento do projeto, em particular o planejamento e a divisão de tarefas, e assegura que a atenção seja focalizada em distribuir corretamente no tempo certo. A metodologia provê uma coleção de métodos, ferramentas e técnicas, que suportam o desenvolvimento durante várias atividades.

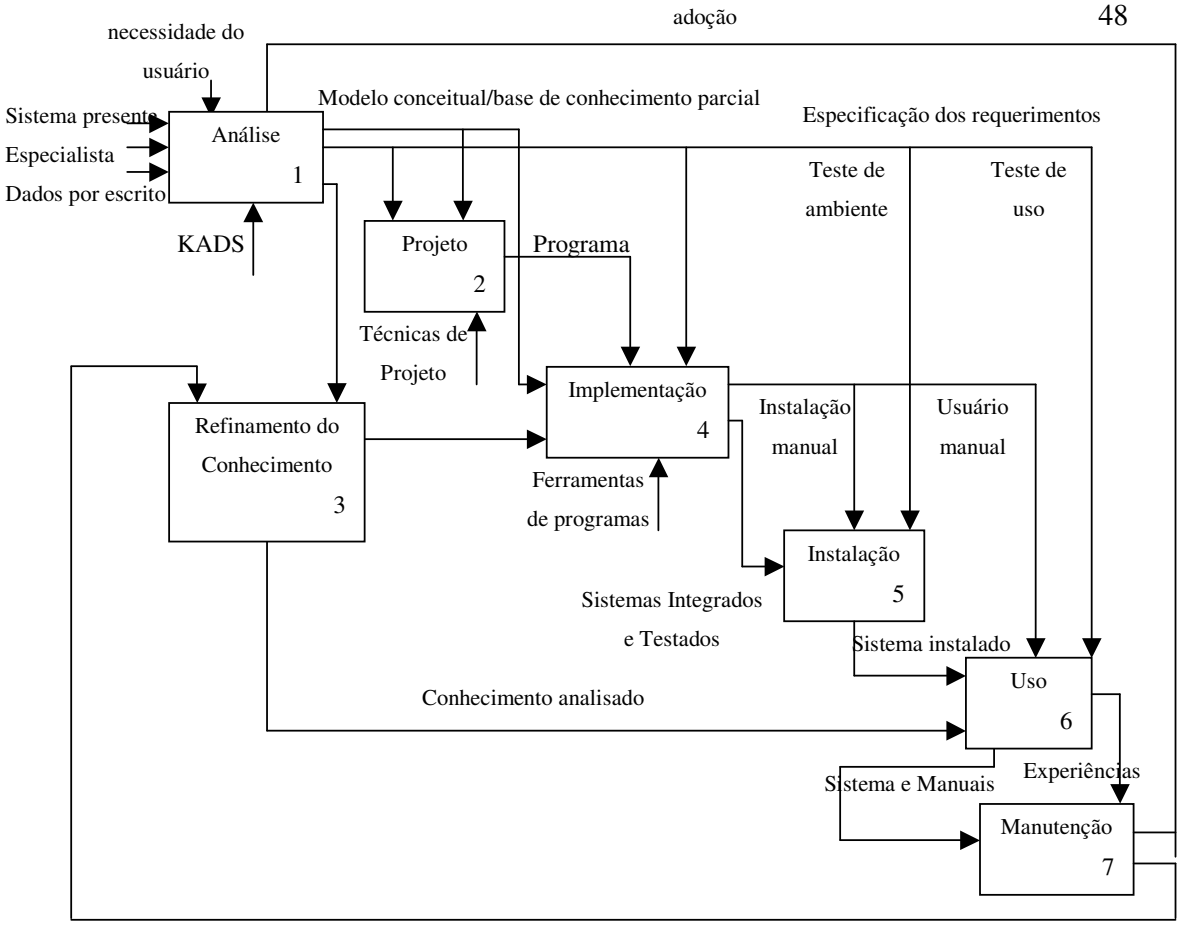

Figura 7: O ciclo de vida do modelo *KADS (*HICKMAN, *et.al.* 1989*).*

Ele usa um modelo de ciclo de vida em cascata modificado, para descrever o processo de desenvolvimento de sistemas, tal como análise, projeto, refinamento do conhecimento, implementação, instalação, uso e manutenção. Um importante aspecto do *KADS* é a importância de fazer em fases a pré-implementação do desenvolvimento: uma análise total dos dados é conduzida antes de algum projeto e implementação.

O *KADS* distingue os seguintes modelos: o modelo organizacional, o modelo de aplicação, o modelo da tarefa, o modelo da cooperação, o modelo da especialidade, o modelo conceitual e o modelo de projeto. A idéia da descrição do nível de conhecimento é incorporada na abordagem do *KADS* através da introdução dos modelos da especialidade, isto é, estes modelos podem ser usados para analisar e especificar o comportamento requerido da resolução dos problemas em um domínio baseado em conhecimento da aplicação do sistema (WIELINGA, *et. al.,* 1993).

Na Fig. 8, o papel que o requerimento externo tem, é, por exemplo, um guia e um foco para a transformação do modelo conceitual para o modelo de projeto. Estes fatores

externos são importantes, pois alteram o conteúdo do modelo conceitual e assim eles habilitam o Engenheiro do Conhecimento para construir um modelo com cuidado acerca dos requerimentos externos na plataforma de desenvolvimento.

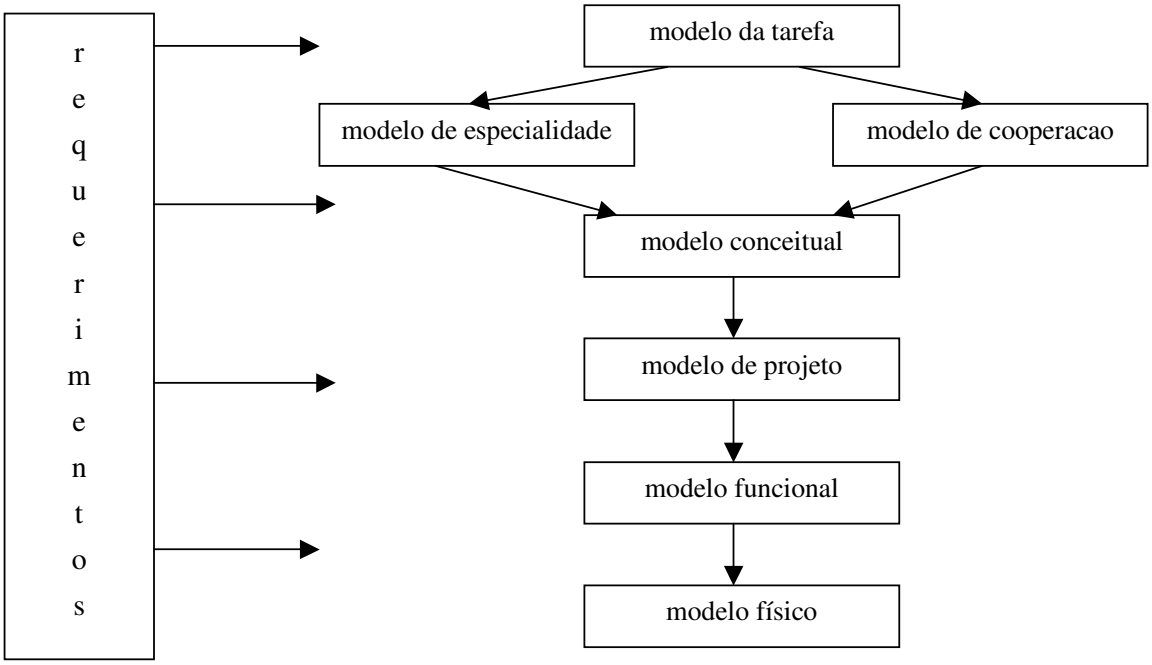

Figura 8: Transformação de modelo em KADS (HICKMAN, *et. al.*, 1989).

O modelo conceitual consiste de duas principais constituições: do modelo de especialidade e do modelo de cooperação. O modelo de projeto foi usado com sucesso em desenvolvimento de sistemas para prover a base de aquisição de conhecimento estruturado e a representação intermediária que precede o estágio de projeto. O modelo físico reflete a visão que não somente está apropriado para a interação do modelo sistema/usuário no nível de abstração, mas que também esta característica para a interação é necessária para ocorrer (HICKMAN, *et. al.,* 1989).

A terminologia e as ferramentas mudaram e alguns dos resultados mais estáveis do projeto se encontram fora do consórcio do *KADS*, sendo chamado de *KADS-I*. O método para modelar o conhecimento foi à base de um método de desenvolvimento de Conhecimento de Engenharia Estruturado (CEE). Apesar de seu sucesso, o projeto *KADS-I,* não alcançou inteiramente seus objetivos originais (WIELINGA, *et. al.,* 1993).

Em 1989, em um projeto entre a Universidade de Amsterdã e da Fundação da Pesquisa de Energia dos Países Baixos (ECN) uma estrutura para a especificação formal

de modelos do nível de conhecimento foi desenvolvida. Em 1990, isto resultou na formação de um novo consórcio chamado agora de *KADS-II*. O objetivo do projeto era desenvolver uma metodologia detalhada para o desenvolvimento de sistemas em IA qualificada para transformar-se em um padrão comercial pelo menos na Europa (WIELINGA, *et. al.,* 1993).

O *KADS-II* evoluiu para o *CommonKADS.* Sendo agora o padrão europeu de fato para incorporar em parte a análise do conhecimento e desenvolvimento de sistema baseado em conhecimento, e foi adotado por muitas companhias na Europa, EUA e Japão (ABEL, 2001), (FALBO & TRAVASSOS, 1995), (FENSEL & POECK, 1994).

3.7.2. Projeto MIKE

MIKE é um dos projetos realizados pelo AIFB (Institute for Applied Computer Science and Formal Description Methods) da University of Karlsruhe. Define um padrão de engenharia para elicitação, interpretação, formalização, e implementação de conhecimento para construir sistemas especialistas (ANGELE, *et. al.* 1993).

O *MIKE* integra técnicas de especificação formal e semiformal no desenvolvimento do processo de forma incremental. As principais características do MIKE são: engenharia de conhecimento como um processo de modelagem, engenharia do conhecimento como um processo incremental, divisão da parte de representação e reuso dos modelos existentes (ANGELE, *et. al.,* 1996).

Uma transição de uma especificação formal e semiformal favorece a realização do projeto com todas as técnicas confiáveis de descrição no mesmo modelo conceitual para descrever os aspectos funcionais e não funcionais do sistema. Assim, o sistema fica documentado em níveis de descrição diferentes, cada um enfocando um aspecto distinto (ANGELE, *et. al.* 1996).

As quatro principais fases do modelo no processo de desenvolvimento do *software* são: análise, projeto, implementação e evolução (CUNHA, 1995). O *MIKE* separa a análise e a implementação durante o projeto de um sistema especialista. O resultado da fase da análise é uma especificação formal da tarefa que é para ser resolvido pelo sistema e do conhecimento que é requerido para resolver a tarefa eficaz e eficientemente.

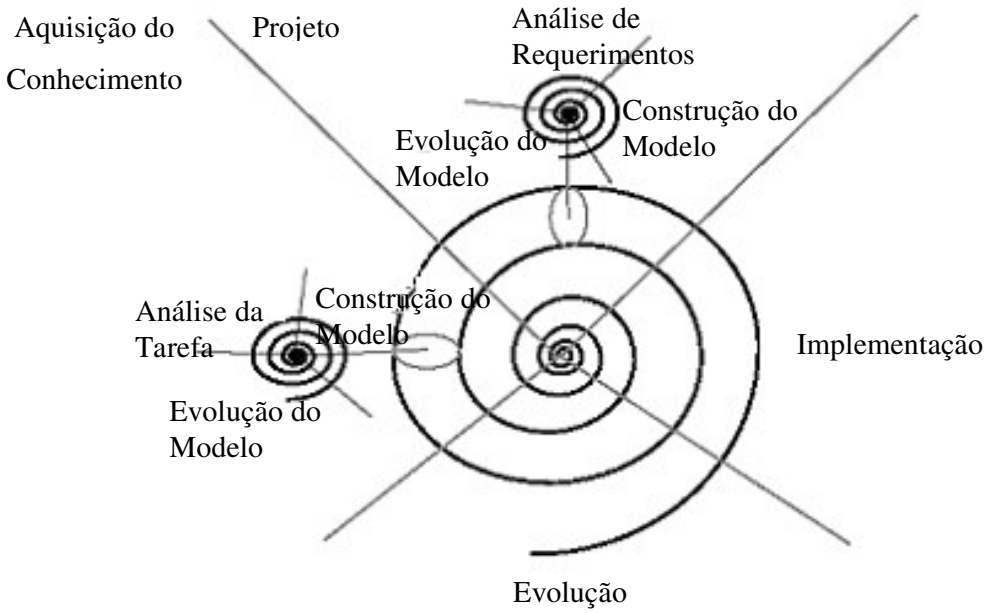

Figura 9: Fase do ciclo de vida em espiral do *MIKE* (ANGELE, *et. al.*, 1993).

O resultado da fase da implementação é um agente computacional que resolve o problema. Conforme Fig. 9, a aquisição de conhecimento das fases, Projeto, Implementação e Evolução são cíclicas até o sistema construído estar realizado conforme desejado.

3.7.3. Projeto PROTÉGÉ

PROTÉGÉ é o resultado do projeto *Knowledge Modeling Group at Stanford Medical Informatics* onde foi proposto um conjunto de ferramentas para modelagem do conhecimento. Era uma aplicação pequena, inicialmente proposta para construir ferramentas de aquisição de conhecimento para alguns programas especializados no planejamento médico (GENNARI, *et. al.,* 1999).

Provê uma plataforma para edição de bases de conhecimento que é facilmente adaptada a qualquer linguagem de modelagem de conhecimento baseada em frame. Usa um modelo de abstração de métodos de resolução de problemas, e permite ao Engenheiro do Conhecimento associar parte do conhecimento de método com legendas específicas do domínio. Baseado nestas associações *PROTÉGÉ* pode ser usado para

gerar ferramentas de modelo de instanciação, que interage com especialistas em terminologia específica do domínio.

O *PROTÉGÉ* fornece a visão do sistema enquanto compreende os modelos do domínio que satisfazem às exigências dos dados de métodos de resolução de problemas reusáveis, o que facilita a modelagem de sistemas inteligentes. A modelagem consiste de:

- Elicitação de novos métodos para resoluções de problemas.
- Desenvolvimento de técnicas para criar, editar, e manter ontologias do domínio.
- Estabelece mecanismos para traçar modelos do domínio às exigências dos dados de métodos de resolução de problemas.

Embora *PROTÉGÉ* tenha sido desenvolvido há 16 anos para suportar a aquisição de conhecimento para sistemas especialistas médicos, tornou-se muito popular para muitas outras finalidades, sendo possível usar para (MUSEN, *et. al.,* 1995):

- Modelagem de classe: Fornece uma relação de usuário gráfica (GUI), classes dos modelos (conceitos do domínio), seus atributos e relacionamentos.
- Edição do exemplo: Destas classes, gera automaticamente os formulários interativos que permitem os especialistas de domínio incorporar exemplos válidos.
- Processar modelo: Tem uma biblioteca que ajuda definir a semântica, fazer perguntas e definir o comportamento lógico.
- Trocar modelo: Os modelos resultantes, classes e exemplos podem ser carregados e conservados em vários formatos, incluindo *XML, UML,* dentre outros.

Suas fases constituem de: identificação, concepção, formalização, implementação, teste e revisão conforme mostrado na tabela 2.

Tabela 2: O modelo clássico do desenvolvimento de sistema especialista em *PROTÉGÉ.*

Estágio	Descrição	Realizado por
1. Identificação	Importantes aspectos caracterizam o problema. Identificando participantes, características dos problemas, recursos e metas.	Especialistas do Domínio, Engenheiro do Conhecimento.
2. Concepção	O conceito e relação da identificação do estágio explícito.	Engenheiro do Conhecimento.
3. Formalização	Conceitos chaves identificados são representados em linguagem formal.	Engenheiro do Conhecimento.
4. Implementação	Estágio de formalização do conhecimento é representado no Shell do sistema especialista. Validação das estruturas de representação.	Engenheiro do Conhecimento.
5. Teste	O sistema completo é testado em casos simples e fraquezas são identificadas.	Especialistas do Domínio, Engenheiro do Conhecimento.
6. Revisão	Projeto e implementação do sistema, no início dos resultados de teste.	Engenheiro do Conhecimento.

Fonte: GENNARI, J.H., MUSEN, M.A., FERGERSON, R.W., GROSSO, W.E., CRUBEZY, M., ERIKSSON, H., NOY, N.F., TU, S.W. The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. International Journal of Human-Computer Studies, 1999.

PROTÉGÉ tem fonte aberta, sendo uma característica muito atrativa aos desenvolvedores. Usa um modelo de abstração de métodos de resolução de problemas, e permite o Engenheiro do Conhecimento associar parte do conhecimento de método com legendas específicas do domínio.

Baseado nestas associações, pode ser usado para gerar ferramentas de modelo de instanciação, que interage com especialistas em terminologia específica do domínio. Fornece a visão do sistema enquanto compreende os modelos do domínio que satisfazem às exigências dos dados de métodos de resolução de problemas reusáveis, o que facilita a modelagem de sistemas inteligentes.

O *PROTÉGÉ–II* tenta vencer as limitações da primeira versão. É um amplo ambiente que embute várias ferramentas, com um mecanismo facilmente configurável e um editor de ontologia. Foi uma significante extensão e generalização do sistema original *PROTÉGÉ.* Provê de uma ontologia de domínio, uma base de conhecimento, e um novo método para resolução de problemas que, com ele, pode conduzir

experimentos no desenvolvimento e no reuso de componentes da base de conhecimento (MUSEN, *et. al.,* 1995), (MUSEN, *et. al.,* 1998).

3.8. Breve comparativo das metodologias

Comparando as abordagens com respeito ao relacionamento de conhecimento do domínio e método de solução de problemas, todos os enfoques são compatíveis, estes pontos principais podem ser identificados:

As três abordagens vêem a diferença de métodos de solução de problemas e conhecimento e enfatizam aspectos diferentes no processo desses sistemas. Uma diferença significativa entre *KADS* e *PROTÉGÉ* é que os modelos de *PROTÉGÉ* são operacionais. Em contraste, o modelo de especialidade em *KADS* é tipicamente uma especificação escrita, que fornece principalmente a orientação para os envolvidos que devem projetar e construir o sistema em etapas separadas.

O *MIKE* tem uma visão do domínio como parte do conhecimento e visa sua descrição declarativa, tenta mecanizar o processamento da engenharia do conhecimento vendo o conhecimento do engenheiro como primordial. Dá ênfase na especificação formal e executável do modelo de especialidade como resultado da fase de aquisição de conhecimento. Na construção o *PROTÉGÉ* do modelo separa e explora explicitamente as ontologias de domínio, aplica orientação a objeto, permite estruturas hierárquicas de modelos complexos de especialidade e enfatiza o reuso. O *KADS* constrói uma coleção de modelos onde cada modelo captura aspectos específicos do sistema.

3.9. Ciclo de Vida de Sistemas em IA

3.9.1. Sistema Especialista

De acordo com Barreto (2001), como qualquer programa, um sistema especialista tem um ciclo de vida. Entretanto em se tratando de tecnologia ainda não perfeitamente absorvida do grande público, se faz mister especificar as fases iniciais de modo mais detalhado, evitando com isto que se inicie um trabalho que não corresponda às expectativas do utilizador final. Assim:

Estudo dos problemas relevantes a serem tratados pelo SE: análise de oportunidade.

Análise Funcional: é necessário saber que funcionalidades são desejadas.

Conceituação: criação do modelo capaz de resolver o problema, incluindo a definição das ferramentas a serem usadas.

Elicitação do Conhecimento: capturar e utilizar o conhecimento de um ser humano em uma aplicação computacional.

Implementação: implementação do sistema em uma linguagem específica.

Teste: utilizando problemas e soluções propostas por especialistas diferentes das usadas para construir a base de conhecimentos e comparando as respostas do SE com o especialista.

Manutenção: atualização da base de conhecimentos, melhoria de interface, dentre outros

Morte: quando o sistema entrar em desuso.

A maioria das etapas do ciclo de vida de um SE pode ser considerada como ciclo de vida de um programa qualquer. Entretanto, a obtenção do conhecimento do especialista pelo Engenheiro do Conhecimento, chamada elicitação do conhecimento, envolve características peculiares (BRASIL, 1994).

3.9.2. Redes Neurais Artificiais

Para o desenvolvimento de uma RNA são levados em consideração muitos fatores. O desenvolvimento se divide em quatro etapas: definição, treinamento, manutenção e utilização. Assim, pode-se observar as etapas para a implementação de uma RNA (TAFNER, *et. al.,* 1995):

Definição da rede: nesta são definidas questões fundamentais, como por exemplo, quais problemas a rede deverá solucionar e quais áreas são importantes para análise. **Treinamento:** esta etapa é de suma importância à validação da rede, pois é nela que a rede recebe os dados para treinamento.

Utilização da RNA: Após um bom treinamento passa a fornecer dados confiáveis, utilizando-se então de estimativas. Pode-se entrar com novos dados para se verificar como certa alteração teria efeito sobre determinada função.

Manutenção: A partir de grandes mudanças de ambiente a rede deve receber certa manutenção para que os pesos das conexões não fiquem desatualizados. Deve estar sempre em prática, caso contrário a rede corre o risco de fornecer dados e previsões incorretas para os quais estava treinada.

Morte: A partir do momento que o sistema não é mais usado.

Diante deste contexto, a princípio poderia ser usada uma das metodologias existentes, entretanto, constatou-se que atuam em níveis diferentes e que algumas fases, e atividades não suprem a necessidade, o que não atingiria o objetivo proposto. Mostra-se assim em linhas gerais as principais desvantagens:

- Cascata
  - Na prática, projetos não seguem o fluxo seqüencial.
  - Acomodações de incertezas no início do projeto é difícil.
  - É necessário esperar até a fase de instalação e liberação para ver como o sistema funciona.
  - Faltam ao modelo as noções de prototipação e desenvolvimento incremental.
  - Torna-se muito dispendioso, caso tenha que ser substituído por uma versão melhorada.
- Prototipação
  - O desenvolvedor pode perceber as reações iniciais do usuário e obter sugestões para mudar ou inovar o protótipo.
  - A iteração pode adequar o protótipo às necessidades do usuário.
  - O protótipo pode ser descartado ou fazer parte do produto final.
  - Ignora alguns requisitos: qualidade, confiabilidade, manutenibilidade, prazos, segurança e riscos.
- Espiral
  - Alto custo para adoção.

- Circunstâncias adversas que podem atrapalhar o processo e a qualidade do produto a ser desenvolvido.
- Não pode ser determinada completamente de antemão.
- O modelo é relativamente novo, requer esperteza. Pode nunca terminar.

- *KADS, MIKE* e *PROTÉGÉ*
  - A metodologia *KADS* tem uma boa descrição do modelo de problema e do modelo de solução, mas a sua descrição do processo de abstração é parcialmente satisfatória, pois é possível notar que se concentra muito na Abordagem Simbólica.

Assim, do desejo de se aplicar algo semelhante na hora de se desenvolver sistemas com qualidade é que se propõe uma nova MEtodologia de Desenvolvimento de Sistemas em IA (MEDSIA). A metodologia ora proposta visou aproveitar os pontos mais significantes e eliminar os pontos negativos das outras abordagens existentes. Combinando estes modelos, criou-se uma metodologia que seja adequada às necessidades dos novos paradigmas que temos hoje. Tudo irá depender do problema a ser resolvido como se mostra a seguir.

4. METODOLOGIA PROPOSTA

Em diversas áreas de desenvolvimento utilizam-se métodos e ferramentas que facilitam o desenvolvimento de programa. Enquanto tais áreas começaram como uma "mera" representação das etapas do processo, elas evoluíram para esquemas de solução de problemas no decorrer do desenvolvimento.

Enquanto os programas de uso geral, com aplicações para diferentes áreas, passaram progressivamente a serem escritos usando metodologias que lhes davam características científicas, programas em inteligência artificial continuavam a ser escritos usando apenas intuição. Muitos autores chegaram mesmo a justificar tal atitude como uma qualidade da IA (RICH, 1983).

Durante muitos anos, pelas mais diversas razões, surgiram conjuntos de regras e técnicas para se desenvolver sistemas, entretanto isso não aconteceu com a IA. Inclusive Rich (1983) declarou que em IA não era interessante criar metodologias para desenvolver sistemas porque "era mais interessante ver que os sistemas demonstrarem ter comportamentos inteligentes do que provar que o programa estava correto".

Embora progressos tenham sido feitos nos últimos anos. Desde o início aplicações convencionais foram privilegiadas. Com o crescimento de sistemas no domínio da IA mais complexos, torna-se cada vez mais importante ter a definição do processo que sistematize o seu ciclo de vida, possibilitando uma maior habilidade de descrever formalmente o comportamento dos modelos. É necessário fazer um acompanhamento de sua execução, guiando os desenvolvedores com uma representação de conhecimento que permita modelar, projetar, avaliar, manutenir, validar, verificar, reusar e acompanhar o processo de desenvolvimento destes programas. É recomendável que não se dispensem esforços na hora da escolha de uma metodologia apropriada e no levantamento de novas soluções alternativas de desenvolvimento.

Em Engenharia de Programas existem estudos de várias metodologias. Analisando a nova direção no final dos anos 80, observou-se que a princípio poderia ser

usada uma das metodologias existentes. Entretanto, constatou-se que atuam em níveis diferentes e que algumas fases e atividades não suprem a necessidade, o que não atingiria o objetivo proposto.

Um fato a constatar é que todas as metodologias propostas e apresentadas em capítulo anterior supõem como ponto de partida que o problema de IA é abordado utilizando técnicas clássicas (hoje de manipulação simbólica). Conceitos mais atuais, se bem que descoberta também antiga, tais como Redes Neurais Artificiais e Computação Evolutiva, devido às suas características intrínsecas, se enquadram mal nos ciclos de vida apontados. Por exemplo, treinar uma RNA corresponderia a quê em um método simbólico?

Um ponto chave a ressaltar é a resposta, a questão óbvia: "que método usar?" Neural? Evolutiva? Simbólico? Algorítmico? Tudo dependerá do problema a ser resolvido.

Frisando o que já foi dito, um problema compreende: enunciado, solução e resposta. Ao um mesmo enunciado podem corresponder várias maneiras de se resolver um problema.

Neste contexto, do desejo de se aplicar algo semelhante na hora de se desenvolver sistemas com qualidade é que se propõe uma nova MEtodologia de Desenvolvimento de Sistemas em IA (MEDSIA). A metodologia ora proposta visa aproveitar os pontos mais significantes e eliminar os pontos negativos de outras abordagens.

Este esforço resulta em uma nova metodologia de desenvolvimento voltado para a IA. Devido à sua natureza altamente iterativa e incremental é similar ao modelo de ciclo de vida iterativo e incremental. Pode-se inclusive desenvolver um protótipo logo na segunda fase, a fim de se avaliar o sistema a ser produzido. Os objetivos de cada etapa também são semelhantes àqueles propostos por outros modelos.

4.1. MEDSIA

Como já dito anteriormente MEDSIA é uma sigla que significa "MEtodologia de Desenvolvimento de Sistema de IA. Foram apresentadas anteriormente as metodologias *KADS, MIKE* e *PROTÉGÉ*, que eram voltadas para desenvolvimento simbólico. Visando generalizar estas metodologias, a MEDSIA compreende as seguintes etapas:

1. Análise de oportunidade
2. Definição do Problema
3. Reconhecimento do Problema e da Ferramenta para resolvê-lo
4. Análise Funcional
5. Implementação
6. Validação
7. Implantação
8. Manutenção
9. Morte

Analisando as 9 (nove) fases apresentadas, tem-se a seguir suas principais características conforme Fig 10:

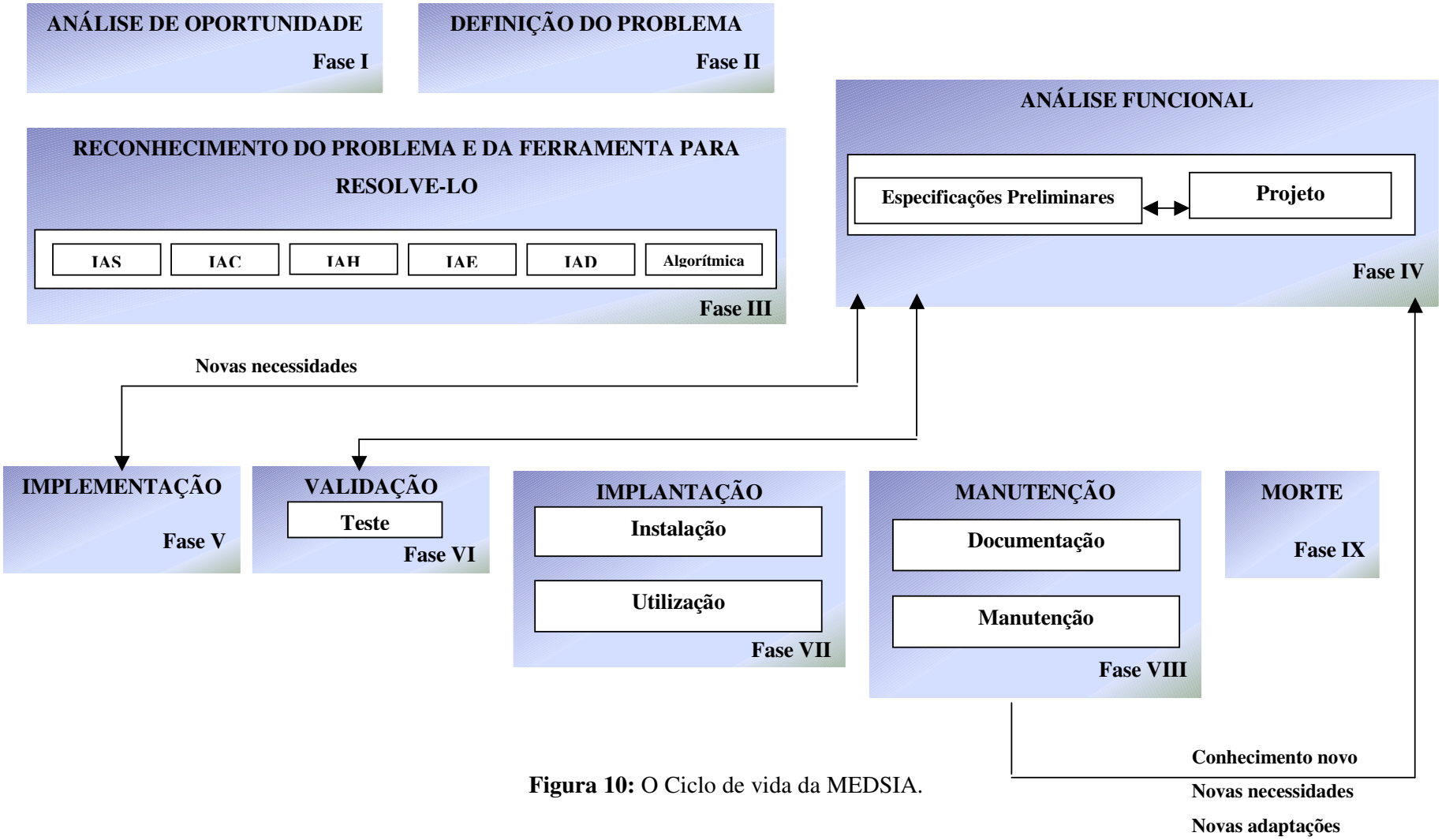

Figura 10: O Ciclo de vida da MEDSIA.

4.2. Fase I - Análise de Oportunidade

Esta primeira fase caracteriza-se de uma análise de mercado. As metodologias supõem como ponto de partida o fato de que o programa a desenvolver atende a necessidade de algum cliente.

Entretanto, muitos são os programas que ficam na prateleira simplesmente por terem sido lançados no mercado sem que este mercado necessite de suas funcionalidades; é como se o programa tivesse sido escrito antes de seu tempo.

Fazer uma análise das opiniões de possíveis usuários de um programa antes de escrevê-lo será aqui chamado Análise de Oportunidade. Se bem que algumas vezes pode-se dizer que esta análise é algo óbvio, isto não é verdade, e a prova é o insucesso de aceitação de tantos sistemas tecnicamente corretos, bonitos e potencialmente úteis.

Muitas vezes o usuário não esteve presente, não participou de alguma forma, sendo que, o usuário deve estar presente desde a idéia até o produto final. Análise de Oportunidade deveria ser sempre evidenciada principalmente para que não se esquecesse que resultados positivos nesta fase são indispensáveis para o sucesso de todo o sistema.

Esta fase pode parecer óbvia, mas temos vários exemplos concretos dentre eles o *Mycin*, em que a fase foi esquecida resultando em uma aplicação de IA que nunca foi utilizada.

Este exemplo é a tese de doutorado do Shortliffe (1974). Em encontro com o orientador deste trabalho em Viena 1990, Shortliffe relatou que nunca viu o seu programa ser utilizado realmente, bem como, nenhum outro programa de diagnóstico médico.

Constatou Shortliffe (1976), que a primeira exceção foi durante a sua visita a um hospital de Tókio, em que havia um programa que era utilizado por todos os médicos e todos os pacientes. Tentou descobrir a razão de tão grande sucesso do programa. Analisando sua estrutura verificou tratar-se de algo semelhante ao *Mycin*. E agora? Porque ninguém usa *Mycin* e todos usavam este programa? A resposta veio da conversa com a secretária no dia de sua partida: ela lhe disse ser óbvio que todos utilizassem o programa, pois se tratava de hospital particular e era o dono do hospital quem fizera a

concepção e programação do sistema. Não uso do programa será igual a pedido de demissão do hospital (SHORTLIFFE, 1991). Neste contexto é necessário saber de antemão:

- Quais são as oportunidades de mercado?
- Qual é o potencial do mercado?
- Os Clientes possuem capacidade e disponibilidade para pagar pelos produtos ou serviços?
- Qual a competição na área afim?

Se fosse analisado o ambiente em que um programa de diagnóstico médico deveria ser usado, seria fácil chegar à conclusão de que não se deveria chegar à etapa seguinte, pois os médicos não precisam e não querem programa de diagnóstico. Com alguma dificuldade aceitariam um programa que ajuda a decisão médica, mas, jamais de diagnóstico, pois a palavra de "diagnóstico" é prerrogativa do médico conforme Lei número 3268, de 30 de setembro de 1957, e implica responsabilidade, que um computador não é capaz de assumir.

4.3. Fase II - Definição do problema

A definição de um problema pode envolver cinco etapas: identificação do problema a ser solucionado, avaliação do problema, tomada de decisão, execução e avaliação da solução (POLYA, 1975).

Todo problema contém um enunciado, um método de resolução e um resultado. Estes três componentes do problema ocorrem em cascata. Com efeito, não se consegue ter a resposta do problema sem ter realizado a resolução do mesmo. Da mesma forma, o número de resolução depende do enunciado e como o mesmo é apresentado (BARRETO, 2001). Veloso (1981) *apud Polya,* sugere que antes de tentar buscar a solução de um problema procure-se responder as seguintes perguntas:

- Quais são os dados?
- Quais são as soluções possíveis?

- O que caracteriza uma solução satisfatória?

Formular um problema é diferente de achar a solução do mesmo. A definição de um problema possibilita testar se um certo elemento é ou não solução, mas não guia na busca deste elemento. Verificar se um elemento é solução é simplesmente verificar se os dados e elemento candidato à solução satisfazem ou não a condição, mas, nem sempre é possível, isto dependerá do modo de definir a função associada ao problema. Este fato é a origem de um dos métodos de busca em IAS, o de gerar e testar (NILSSON, 1982).

Para definir o modelo a ser utilizado na solução do problema deve-se responder o que caracterizará o problema a ser resolvido, por exemplo:

- Qual o problema a ser solucionado?
- O problema é ou não computável?
- O problema é fixo?
- O problema é bem estruturado, bem conhecido?
- Quais resultados deverão ser alcançados?
- Qual a sua caracterização? Quais os aspectos mais importantes?

4.4. Fase III - Reconhecimento do Problema e da ferramenta para resolvê-lo

A IA aborda problemas pouco estruturados, onde não se conhece na maioria dos casos qual o melhor método para resolvê-lo no desenvolvimento de sistemas no domínio da IA. Deverá ser levado em consideração a escolha da oportunidade, bem como a definição do problema, e em seguida, desenvolver um estudo preliminar do problema.

A escolha de ferramenta a ser utilizada no desenvolvimento dependerá da forma e dos dados disponíveis, ou seja, como o conhecimento se apresenta. O problema da escolha recai em uma árvore e se tem que determinar a melhor ferramenta ou abordagem a ser utilizada. É necessário um levantamento ou perguntas possíveis com determinada abordagem, a que responderá qual abordagem a ser utilizada. Assim, alguns exemplos:

- Quais são as suas necessidades?

- Quais são as metas do sistema?
- Quais são os pontos fracos do desenvolvimento do sistema?
- Quais processos poderão ser utilizados?
- Qual o tipo de método que será usado?
- Será Algoritmo Numérico? Manipulação Simbólica? Conexionista? Sistema Evolutivo?

4.5. Fase IV – Análise Funcional

4.5.1. Especificações Preliminares

É o momento de buscar informações pertinentes. Esta fase envolve o estudo e seleção de métodos, escolha de formas de representação, dentre outros. Esta deve ser detalhada, para permitir a identificação de todos os requisitos relevantes a serem atendidos pelo sistema, de maneira a evitar que o sistema deva ser modificado após ou durante a implantação. Para apoiar o levantamento algumas técnicas podem ser usadas como:

- Amostragem: o engenheiro de conhecimento escolhe para entrevista um grupo de indivíduos que tem conhecimento e interesse no novo sistema.
- Investigação: análise de documentos.
- Entrevistas: gravador, anotações (relatório ou ata).
- Questionário.
- Observação.

Nesta fase é recomendável tratar o nível de risco ou "evento futuro" do projeto. Esta é uma atividade muito importante, sendo necessário seis atividades principais: identificar, analisar, planejar, acompanhar, controlar e comunicar. As atividades identificar, analisar e planejar, devem ser tratadas no plano de projeto. O gerenciamento de riscos no projeto se estende ao longo de todo o projeto e três fatores a influenciam:

- Nível de estruturação do problema: quanto maior for o número de possíveis soluções maior será o risco.

- Tamanho do projeto: quanto maior o projeto maior será o risco.
- Nível de tecnologia: se as ferramentas a serem usadas foram desconhecidas, maior será o risco do projeto.

Nesta etapa pretende-se realizar um levantamento de questões chaves:

- objetivo principal e elaborar plano detalhado;
- análise de contexto;
- análise de risco;
- plano de gerenciamento de risco (medidas para tomar e monitoramento de risco);
- alternativas e escolha justificada;
- determinar as necessidades de informação;
- levantamento da situação: o que o sistema deve fazer;
- identificação de processos;
- requisitos, atividades e riscos;
- estruturação e representação dos requisitos do sistema;
- gerenciamento dos requisitos:
  - ferramentas (CASE, linguagens de programação, linguagens de modelagem, SGBD (Sistema Gerenciador de Banco de Dados, dentre outros);
  - recursos humanos, financeiros, prazos, orçamentos, ambiente e infra-estrutura.
- especificação de documentos, questionários, relatórios, formulários, dentre outros;

4.5.2. Projeto

Esta fase envolve a realização do projeto, descrição dos módulos, gerenciamento da base de conhecimento, definir e projetar funcionalidade, soluções e arquitetura do programa, identificar novos riscos e projetar aplicação.

Após todas as etapas definidas como descritas anteriormente inicia-se o projeto permitindo realizar o processo de desenvolvimento do sistema, tais como:

- preparar os planos detalhados (elaborar plano básico de projeto);
- análise de documentos, questionários, dentre outros;
- análise de requisitos funcionais e não-funcionais;
- elaborar especificação do sistema;
- características do sistema;
- identificação dos procedimentos, interfaces, aspectos físicos, identificação da plataforma para implementação;

Esta fase envolve o aprimoramento do conhecimento e definição dos módulos e estrutura. Nesta fase já é possível desenvolver um protótipo para ter uma visão de funcionamento e auxiliar na compreensão do sistema, cuja estrutura representa e processa o conhecimento do problema, possuindo algumas limitações.

O protótipo possibilitará: validar a proposta, confirmar a escolha da técnica de representação e fornecer um veículo para aquisição do conhecimento. Nesta fase pode ser realizado o refinamento do conhecimento, do controle e da interface.

Realizada a atividade de especificação preliminar, serão realizadas análises onde são tomadas decisões importantes, podendo inclusive retornar à atividade de especificação preliminar, quando não forem alcançados resultados satisfatórios.

4.6. Fase V – Implementação

Esta fase envolve o desenvolvimento do programa, tais como:

- construir código fonte;
- realizar teste unitários;
- corrigir defeitos;
- integrar código ao sistema;
- avaliação e adoção de ações preventivas sempre que for necessário.

Esta fase é o estágio de formalização do conhecimento sendo representado na ferramenta escolhida pelo EC (Engenheiro do Conhecimento). São realizadas análises onde são tomadas decisões importantes, podendo inclusive retornar à etapa de Análise Funcional, quando não forem alcançados resultados satisfatórios.

4.7. Fase VI – Validação

4.7.1. Teste

Planejar e realizar testes de validação. É necessário prever todas as situações que o sistema deverá responder. O resultado final dos testes deverá validar o sistema e ter uma aprovação do usuário do futuro sistema, considerando facilidade de uso, clareza das questões e explanações, apresentação dos resultados e utilitários do sistema.

Deverá ser periodicamente testado e avaliado para assegurar que estará convergindo para estabelecer os objetivos propostos. O processo de testes e avaliação apresenta uma característica evolucionária com o desenvolvimento do projeto: apresenta os estágios de testes preliminares, de demonstração, validação informal, refinamento, formais e testes de campo. São realizadas análises onde são tomadas decisões importantes, podendo inclusive retornar à etapa de Análise funcional, quando não forem alcançados resultados satisfatórios.

4.8. Fase VII – Implantação

Nesta fase envolve a identificação de eventuais necessidades de ajustes permitindo verificar se o sistema é funcional e atende aos requisitos propostos.

4.8.1. Instalação

Instalação do sistema. Treinamento do usuário nas atividades de operação. Início de operação do projeto. São realizadas análises onde são tomadas decisões importantes, podendo inclusive retornar à etapa de Implantação, quando não forem alcançados resultados satisfatórios.

4.8.2. Utilização

Nesta etapa é possível vislumbrar novas alternativas e possibilidades de uso.

4.9. Fase VIII – Manutenção

4.9.1. Documentação

Deve ser organizada de forma a facilitar a entrada de novo conhecimento, o acesso e a modificação do conhecimento antigo, o acesso a informações relacionadas.

4.9.2. Manutenção

Será necessário fazer alterações ao longo do tempo para que o sistema esteja adequado às mudanças para que atenda as demandas do usuário. São realizadas análises onde são tomadas decisões importantes, podendo inclusive retornar à etapa de Análise Funcional, quando não forem alcançados resultados satisfatórios.

4.10. Fase IX – Morte

A adaptação ou evolução do sistema irá continuar até o EC observar que não compensa mais a manutenção; então é a hora do sistema entrar em desuso (morte).

Diante do contexto, a MEDSIA é representada por uma estrutura gráfica demonstrada pela Fig. 11, onde a metodologia permitirá observar a seqüência das fases, adaptada para diferentes abordagens da IA, bem como, visualizar e compreender todo o processo de desenvolvimento.

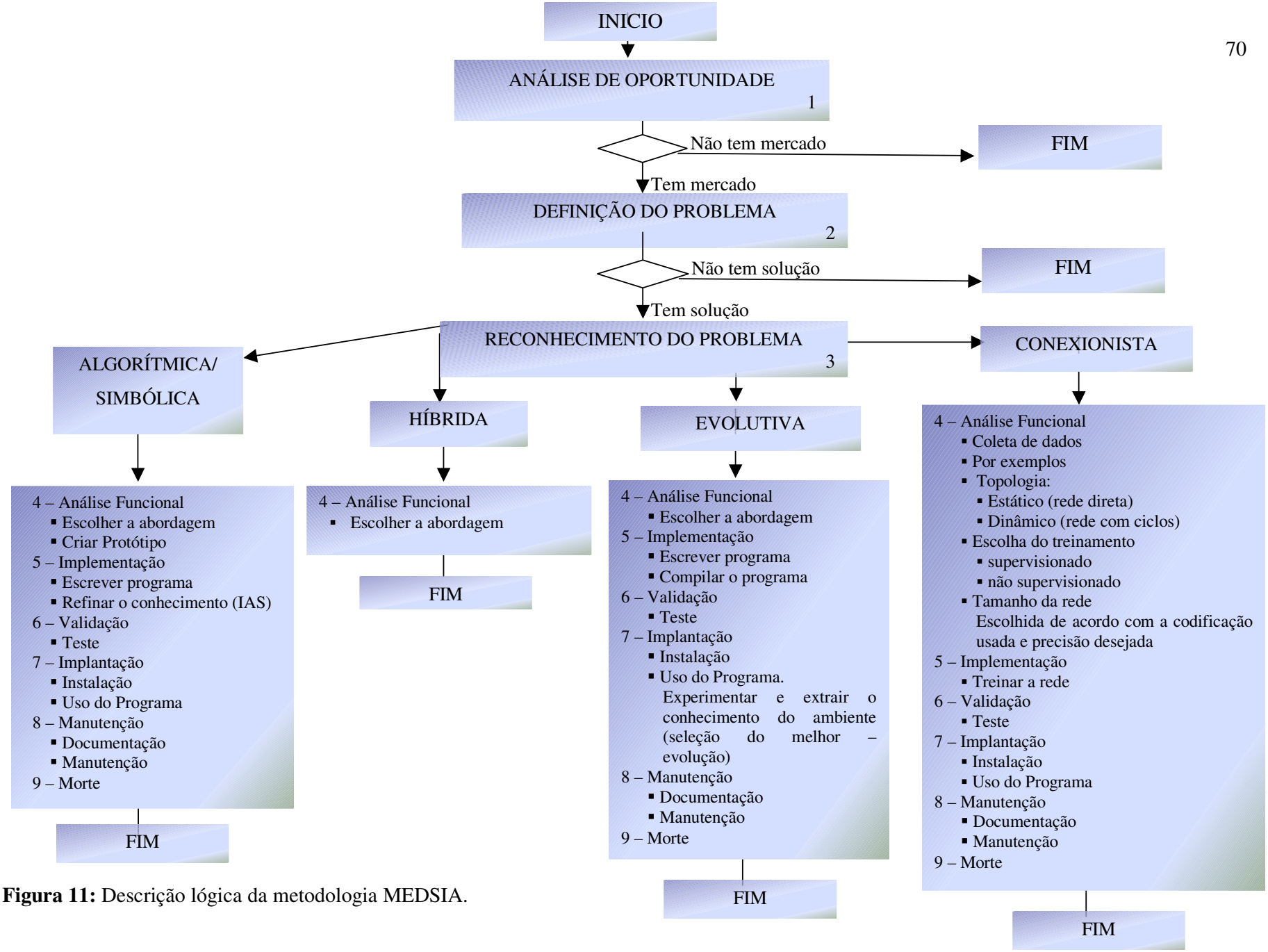

Figura 11: Descrição lógica da metodologia MEDSIA.

4.11. Exemplos clássicos e escolha da abordagem correta

4.11.1. Problemas que poderiam ser resolvidos por um Sistema Algorítmico

- Existem algoritmos conhecidos capazes de resolver o problema?
- O algoritmo é de qual complexidade? Se for NP-Completo melhor tentar outra abordagem.

4.11.2. Problemas que poderiam ser resolvidos por um IAS

- Tem-se uma descrição lógica do problema, ou seja, sabe-se a maneira do raciocínio?
- O problema é bem definido e torna-se fácil saber como ele funciona?
- O problema pode ser decomposto?
- Alguns passos em direção a solução podem ser ignorados
  - Passos ignoráveis: as etapas para a solução podem ser ignoradas. Exemplo: demonstração de teoremas.
  - Passos recuperáveis: as etapas para a solução podem ser desfeitas. Exemplo: quebra-cabeça de 8. O quebra-cabeça de 8 é um tabuleiro quadrado com 9 divisões onde são colocadas oito peças quadradas, numeradas de 1 a 8. A nona divisão permanece descoberta. Cada peça tem um número. A peça adjacente ao espaço em branco pode ser deslocada para aquele espaço. O jogo consiste em uma posição inicial e uma posição-meta. O objetivo é transformar a posição inicial em posição-meta, deslocando as peças até que elas atinjam o local desejado.
  - Passos irrecuperáveis: as etapas para a solução não podem ser desfeitas.
  - Exemplo: xadrez. Pode ser implementado usando-se agentes inteligentes.

- O universo do problema é previsível?
  - Com resultado certo: pode-se usar planejamento para gerar uma seqüência de operadores que certamente levará a uma solução. Exemplo: quebra-cabeça.
  - Com resultado incerto: pode-se usar planejamento para gerar uma seqüência de operadores com boas chances de levar a uma solução. É necessário uma revisão de planos durante a execução e que seja fornecida a realimentação necessária.
  - Exemplo: truco, canastra.
- O domínio é bem limitado?
- Envolve tratamento de incerteza ou o raciocínio é inexato?

4.11.3. Problemas que poderiam ser resolvidos por um IAC

- Tem-se necessidade de resposta rápida; através de problemas resolvidos, por exemplo, por problemas baseados em casos?
- Os métodos utilizados tendem a utilizar funções?
- É sensível a pequenas variações?
- O problema é mal definido, ou seja, falta conhecimento explícito para realizar a tarefa?
- Em termos de robustez é pouco sensível em alguns dados falsos?

Exemplo1: Descobrir onde furar um poço de um petróleo. Têm-se os dados numéricos e topográficos de uma região. A sua constituição de camadas de solo, com números apresentando percentagens. Conhecem-se os dados onde se achou petróleo e onde não se achou.

Exemplo2: Descobrir como investir na bolsa de valores. Neste caso o problema é muito complexo. Tem-se, um histórico da evolução da bolsa, o qual constituem em exemplos que se pode utilizar.

4.11.4. Problemas que poderiam ser resolvidos por um IAE

- Tem-se casos resolvidos, não se sabe exatamente a maneira do raciocínio, mas, sabe-se quando uma solução é melhor que outra.
- Tem-se exemplo de solução do problema bem definido?
- É preciso usar método de otimização, com restrições variáveis e muitas vezes desconhecidas?

4.11.5. Problemas que poderiam ser resolvidos por um IAH

- Problemas difícieis de resolver, sendo necessário usar mais de uma técnica para se chegar na solução.
- Soluções distintas se resolvem com a IAH. Problemas que podem ser decomponíveis em subproblemas. Se um subproblema é adequado à manipulação simbólica não quer dizer que os outros subproblemas vão usar a mesma abordagem. Apesar de parecer estranho, este hibridismo permite romper alguns "dogmas" usuais em IA, como:
  - Sistemas Simbólicos não aprendem.
  - Não se consegue extrair explicação de uma RNA.
  - Toda RNA deve ser treinada.

4.11.6. Problemas que poderiam ser resolvidos por um Sistema Distribuído

- Aprendizagem ou programação declarativa?
- Há necessidade de comunicação entre os processos?
- Como são as ações (determinística, estática, dinâmica)?
- Há autonomia ou flexibilidade?
- Utiliza-se, apesar de ser um comportamento inteligente, o resultado de uma aplicação de um algoritmo conhecido?

- Os problemas a serem solucionados terão que ser através da interoperabilidade de sistemas?

Exemplo1: Conta-se que em algum lugar perto de Hanói havia um mosteiro onde os monges dedicam suas vidas a uma tarefa muito importante. No pátio havia três postes bem altos. Enfiado em um deles 64 discos, cada um com um buraco no centro e cada um com um raio diferente. Quando o mosteiro foi criado, todos os discos estavam em um só poste, e cada disco estava em cima daquele com tamanho maior que o seu. A tarefa dos monges é mover todos os discos para um dos outros postes. Apenas um disco pode ser deslocado de cada vez, e todos os outros discos precisam estar em um dos postes. Alem disso, em nenhum momento durante o processo um disco pode ser colocado sobre um disco menor. É claro que o terceiro poste pode ser usado como local temporário para os discos. Qual a maneira mais rápida para os monges concluírem sua missão?

5. EXEMPLOS USANDO MEDSIA

Com a finalidade de poder avaliar a utilidade da metodologia MEDSIA foram integralmente desenvolvido dois sistemas que potencialmente poderão ser utilizados.

5.1. Fase I - Análise de Oportunidade

5.1.1. Apoio à Decisão Judicial

Hoje a competitividade nos negócios exige que os profissionais estejam preparados. Com a evolução rápida no meio das tecnologias de informação e, para seguir este ritmo, o profissional ativo tem o dever de acompanhar a evolução das técnicas existentes. A informação é considerada como um dos principais ativos das empresas e dos profissionais em suas estratégias. No esforço em dominar o conhecimento tornando capaz de seu uso correto é que se propõe desenvolver um protótipo voltado para alunos do curso de Direito – Área de Família – Investigação de Paternidade. O sistema a ser gerenciado irá prover a integração das informações para agilizar a procura de Leis e Artigos vigentes em nosso país, adequadas a determinadas situações.

Este documento visa ilustrar a MEDSIA (MEtodologia de Desenvolvimento de Sistemas em IA), para a construção de programas. Com a finalidade de expor e demonstrar a metodologia proposta será desenvolvido uma análise evidenciando somente as Fases I, II, III e IV, sendo encontradas nestas as características principais da mesma. Serão analisados os recursos necessários para a realização do projeto, considerando os recursos do ponto de vista técnico, levando-se em consideração: equipamento para processamento, sistema operacional, usuário, recursos materiais, dentre outros. Serão apresentados os resultados parciais da análise, no que tange basicamente aos aspectos de informação. Busca-se, portanto:

- Melhor performance.
- Segurança oferecida pelo sistema.
- Rapidez de operação.
- Facilidade na manutenção.

O sistema a ser construído tem a finalidade de ilustrar a MEDSIA. Trata-se de um programa ainda não existente no mercado e pela necessidade de discentes do curso de Direito em gerenciar as informações referentes às Leis vigentes aplicáveis na área de Direito de Família – Investigação de Paternidade.

O cliente é o Dr. Computastro Urano Satélite. Atuando no Escritório situado a Av. Cratera 2005, Buraco 3024, Edifício Empresarial Nazza, 28° andar, sala 150000, Lua – Sistema Solar.

5.2. Fase II – Definição do Problema

Os discentes do curso de Direito precisam recorrer a Leis e Artigos vigentes. Tais Leis e Artigos são da área de Direito de Família - Investigação de Paternidade. Este levantamento é realizado de forma sistemática, através de pesquisas em livros.

Há muita dificuldade em manipular esta grande quantidade de informações sem o auxílio de uma ferramenta que faça uma análise e defina qual Lei e Artigo deverá ser aplicada para determinada situação. A preocupação é: manter-se sempre atualizado com as Leis e Artigos vigentes e manusear tais informações de uma forma eficaz e eficiente.

5.3. Fase III – Reconhecimento do Problema

Na definição do problema acima descrita levantou-se algumas hipóteses para a escolha da melhor abordagem como a seguir:

- Há necessidade de grande quantidade de conhecimento para resolver o problema? Ou o conhecimento é importante apenas para limitar a busca?
- É possível utilizar conjunto de regras e/ou indução e árvores de decisão ou outros instrumentos que capturem o conhecimento da resolução de problemas?
- Podem ser descritos por um grupo de características que mostrem como podem ser implementados?
- Não há solução algorítmica, mas, existe conhecimento para ser implementado?

No reconhecimento do problema uma solução satisfatória para a dificuldade acima descrita será desenvolver um sistema especialista tutor para apoio a decisão judicial a qual se dará o nome de Sistema Especialista de Apoio a Decisão Judicial - SISEADJU.

Em um primeiro momento se fará uma demonstração desenvolvendo um protótipo para validar a proposta e entender as necessidades do cliente.

5.3.1. Impacto

O sistema a ser implantado visa oferecer a consulta e análise rápida e precisa das informações. Automatização do controle das atualizações das Leis vigentes. Ganho de tempo na manipulação dos dados. Atender as informações agilizando e tornando a busca pelas informações relacionadas: ágil, seguro e eficiente, para que o usuário possa resolver o problema o mais rápido possível.

5.3.2. Metas

As metas do sistema serão:

- um sistema de fácil utilização;
- de acordo com as leis federais vigentes e atualizadas em nosso país;
- integrar as informações em um banco de dados e agilizar a procura de Leis e Artigos de forma que as informações estejam interligadas e reduzam a possibilidade de erros humanos;
- disponibilizar recursos para manipulações auxiliando no processo de armazenamento e na busca de informações ao banco de dados, com garantia de completude, segurança, confiabilidade, agilidade, eficiência destes dados;
- permitir consultas às leis aplicáveis a determinadas situações, e através desta o aluno terá condições para elaborar planos, relatórios e petições aplicando as leis e artigos corretamente.

- gerar dados coerentes e seguros eliminando ou diminuindo em aproximadamente 60% os erros comuns e freqüentes em trabalhos acadêmicos;

5.4. Fase IV - Análise Funcional

5.4.1. Escolha do conteúdo e organização do conhecimento

Caso o cliente seja o Autor, será necessário levantar as seguintes questões:

- O menor é registrado?
  - Se for registrado: fazer um pedido de pensão alimentícia cumulada com alimentos, como também petição de herança caso houver.
  - O menor está registrado, mas o réu nega ter condições financeiras:
    - Se negar sem condições: o Réu pode deixar de exercer, mas não poderá renunciar o direito a alimentos.
  - Senão poderá ser feito um acordo entre as partes para estipular o valor, podendo o equivalente exceder o valor de 30% do salário do réu.
  - Se não for registrado: fazer um pedido de investigação de paternidade cumulada com alimentos, como também petição de herança, caso houver.
    - Caso o suposto pai seja falecido: fazer um pedido de investigação de paternidade. Pode-se pedir exame de DNA dos supostos avós, ou irmãos ou tios.

Caso o cliente seja o réu, será necessário levantar as seguintes questões:

- O réu discorda da paternidade?
  - Se discorda: fazer um pedido de investigação de paternidade.
  - Se não discorda, mas, nega ter condições financeiras:

- Se negar sem condições: o réu pode deixar de exercer, mas não poderá renunciar o direito a alimentos.

- Senão poderá ser feito um acordo entre as partes para estipular o valor, podendo o equivalente exceder o valor de 30% do salário do réu.

5.4.2. Árvore de Decisão Parcial do SISEADJU

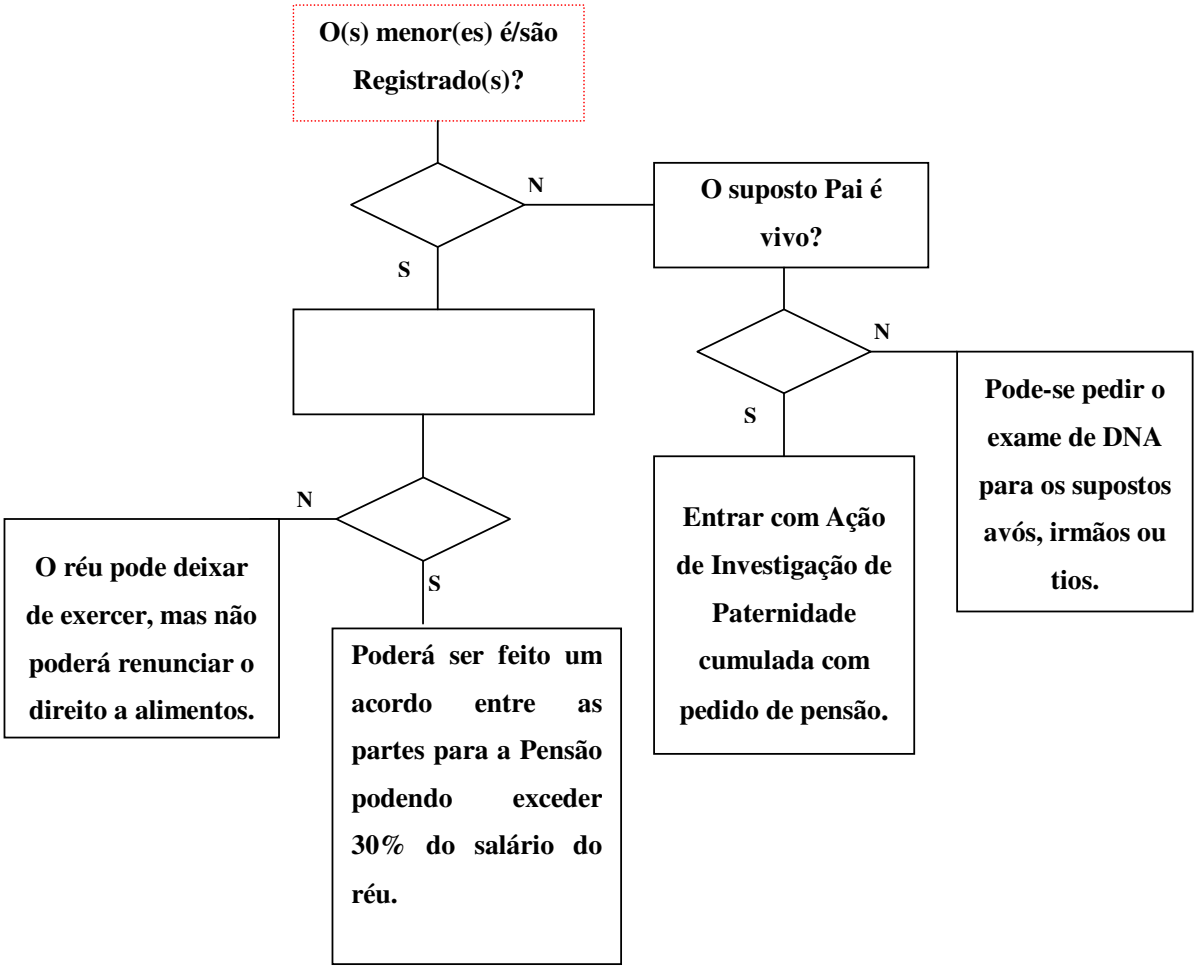

Figura 12: Árvore de Decisão Parcial do SISEADJU.

5.4.3. Requisitos Funcionais

Tabela 3: Requisitos funcionais do SISEADJU.

Funcionalidade	Categoria
Cadastrar Leis e Artigos atualizados.	Evidente
Realizar consulta ao banco de dados.	Evidente
Realizar a operação de busca no banco de dados.	Oculta
Exibir as informações na tela.	Evidente
Realizar a operação de enviar os dados para impressora do usuário.	Oculta
Deve prover um mecanismo persistente de armazenamento.	Escondida

5.4.4. Requisitos Não-Funcionais

Tabela 4: Requisitos não-funcionais do SISEADJU.

Ref.	Funcionalidade
Tempo de resposta	Ao consultar dados da empresa/profissional os dados devem ser visualizados em até 10 segundos.
Tipo de Interface desejada	Usar caixas de diálogos. Maximizar a facilidade do uso de teclado. O usuário não terá dificuldades em identificar as operações bem como executa-las.
Plataformas operacionais	**Configuração Mínima:** Micro computador xx 233 MHz ou equivalente. 64 Mb RAM ou superior. HD 8 GB ou superior. Sistema Operacional. Impressora jato de tinta ou equivalente. **Configuração desejada:** Micro computador xx2.66 MHz. 128 Mb RAM ou superior. HD 40 GB ou superior. Sistema Operacional. Impressora jato de tinta ou equivalente.
Desempenho	O tempo de resposta garantirá uma consulta confiável, rápida e segura.

Serão selecionados alguns elementos da UML (*Unified Modeling Language*) para melhor expressar a modelagem. A ferramenta de modelagem visual escolhida foi o Rational Rose 2000 para fornecer informações importantes antes de gerar os códigos.

5.4.5. Dicionário de Dados

Classe: representa as classes nas quais os artigos podem ser classificados. Atualmente, são duas as classes: pensão alimentícia e investigação de paternidade.

- codigoclasse: código da classe, composto por números.
- descricaoclasse: nome da classe (pensão alimentícia e/ou investigação de paternidade).

Artigo: representa os artigos vigentes em nosso país.

- codigoartigo
- descricaoartigo: nome da lei (Exemplo: Art.2º, Parágrafo 4º).

Lei: representa as Leis vigentes em nosso país.

- codigoartigo
- descricaoartigo: nome da lei (Exemplo: Lei 8.560 de 29.12.1992.).

Usuário: classe abstrata que representa o usuário que irá utilizar o sistema.

- codigousuário
- nomeusuário

Especialista: classe abstrata que representa o especialista que irá utilizar o sistema.

- codigoespecialista
- nomeespecialista
- enderecoespecialista
- codigocidade
- foneespecialista

Engenheiro do conhecimento: classe abstrata que representa o engenheiro que irá utilizar o sistema.

- codigoengenheiro
- nomeengenheiro

- enderecoengenheiro
- codigocidade
- foneengenheiro

Cidade: representa as cidades do nosso país.

- codigocidade
- nomecidade
- ufcidade

5.4.6. Modelo de Classes

A modelagem de classes envolve a identificação de classes, atributos, associações e operações. A seguir, são apresentados os resultados da análise, no que tange basicamente aos aspectos de informação. A Fig.13 apresenta o Diagrama de Classes para um melhor entendimento do futuro sistema.

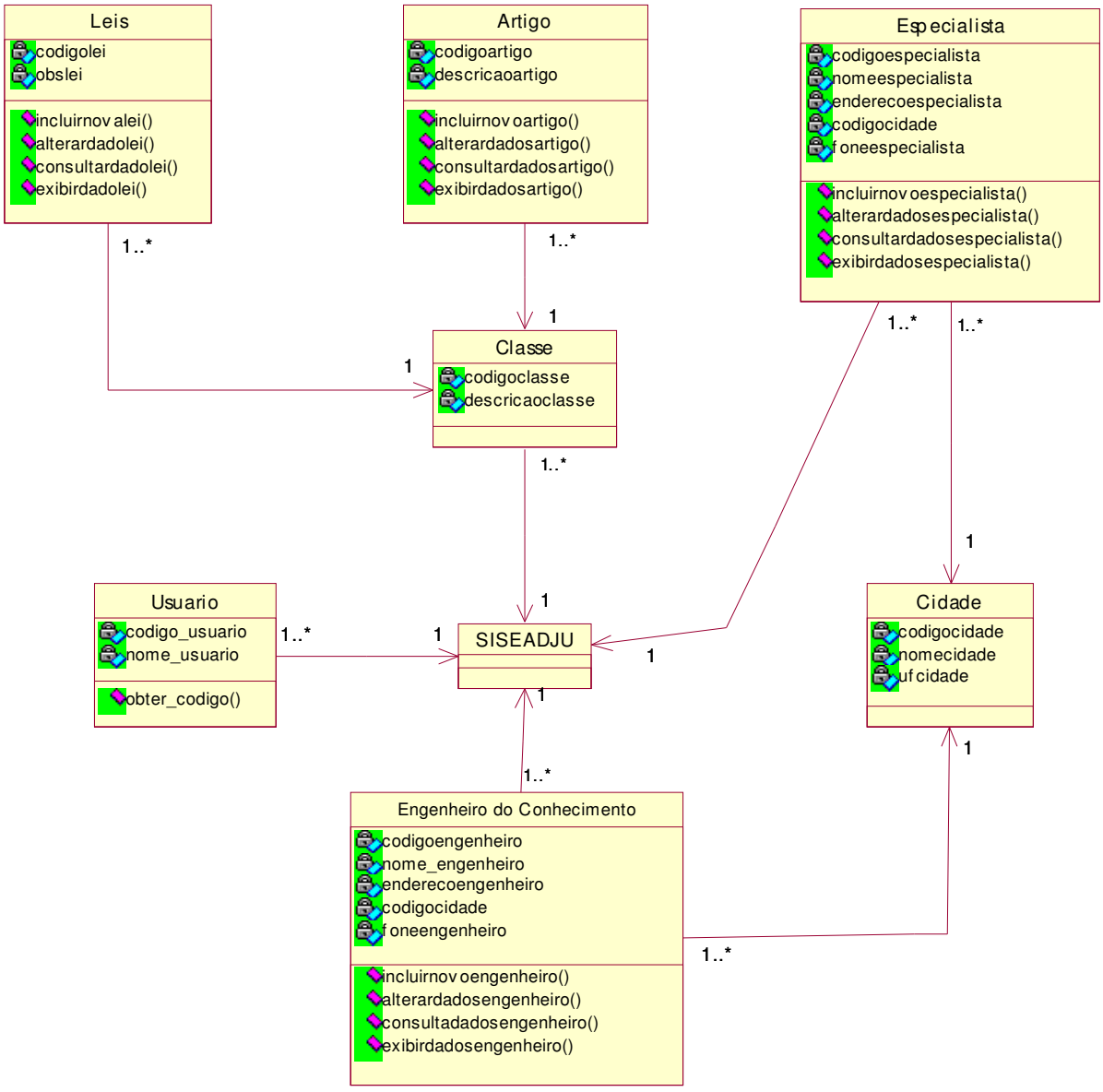

Figura 13: Diagrama de Classes do SISEADJU.

5.4.7. Escolha de formas de Representação de Conhecimento Simbolista

Foram utilizadas técnicas de entrevistas e questionários realizados durante o mês de fev/mar de 2004 para: captar o conhecimento do especialista conforme anexo 1; avaliar o desempenho e manipulação do sistema com a finalidade de verificar sua eficácia que poderá ser visualizado através do anexo 2.

5.4.8. Escolha da Ferramenta de Desenvolvimento

Em um primeiro momento se fará uma demonstração desenvolvendo um protótipo para validar a proposta e entender as necessidades do cliente. Definiu-se uma linguagem declarativa, ou seja, foi utilizado o Prolog. A ferramenta de desenvolvimento escolhida foi o *Strawberry Prolog* (*DOBREV*, 2003), pelo fato da facilidade de utilização, por ser capaz de armazenar e apresentar informações com eficiência, por ser flexível, fácil e intuitivo de usar, o que permitirá uma prototipação para um melhor entendimento de como será o sistema.

5.4.9. Implementação do protótipo

A interface do SISEADJU é amigável baseada em um ambiente "*windows*" proporcionado pela ferramenta *Strawberry Prolog*. A Fig. 14 e 15 se refere a tela principal do SISEADJU.

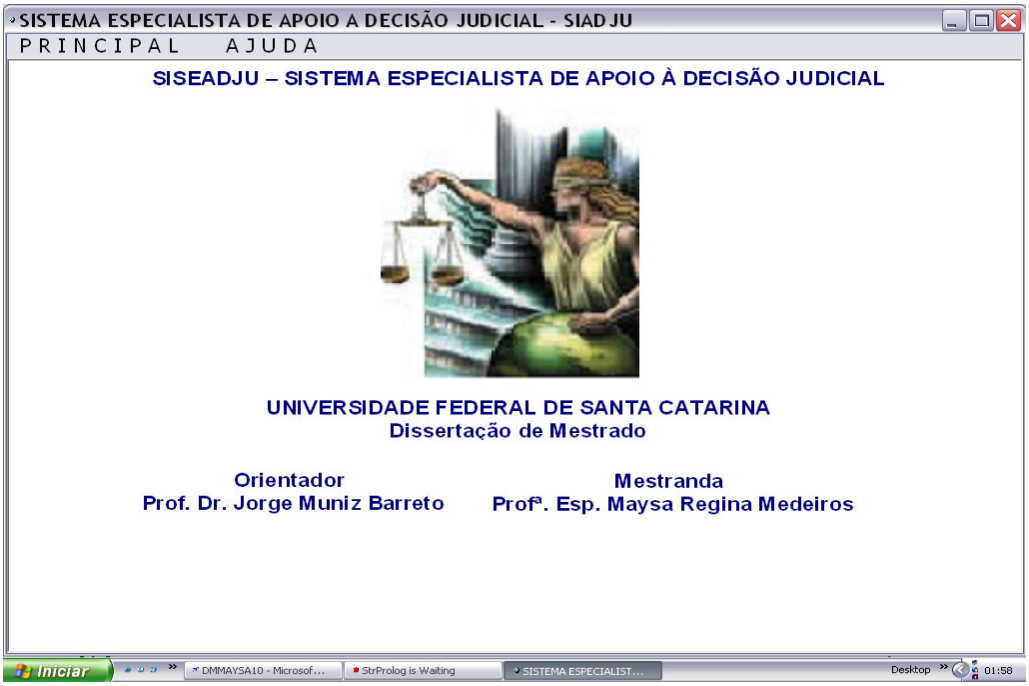

Figura 14: Acessando o SISEADJU.

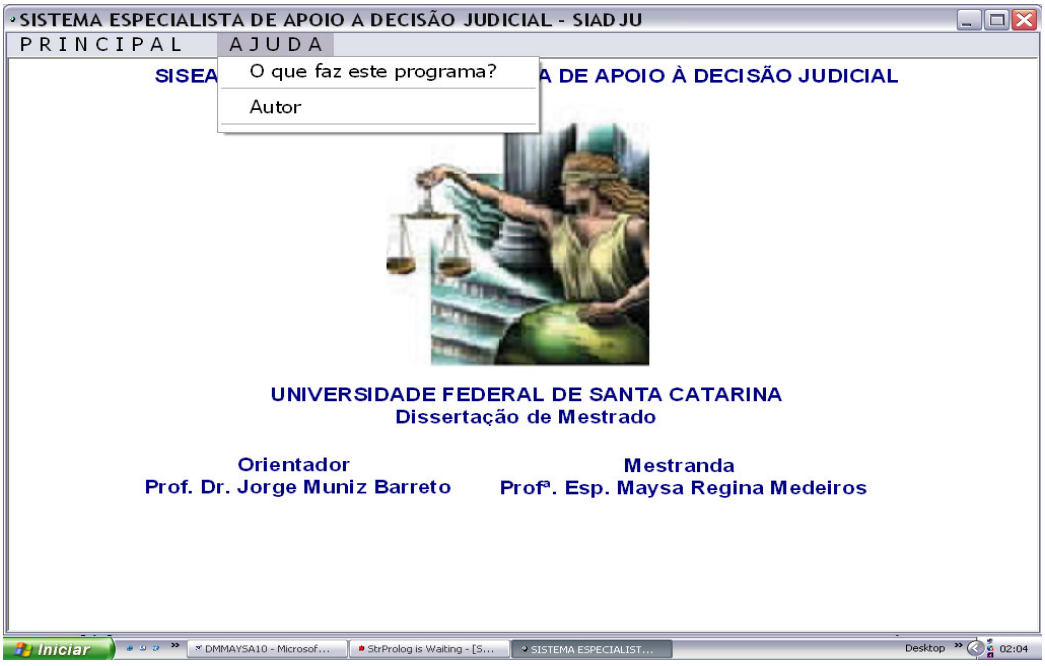

Figura 15: Tela principal do SISEADJU.

As Figuras 16 e 17 mostram o que faz o sistema e o autor que desenvolveu o protótipo.

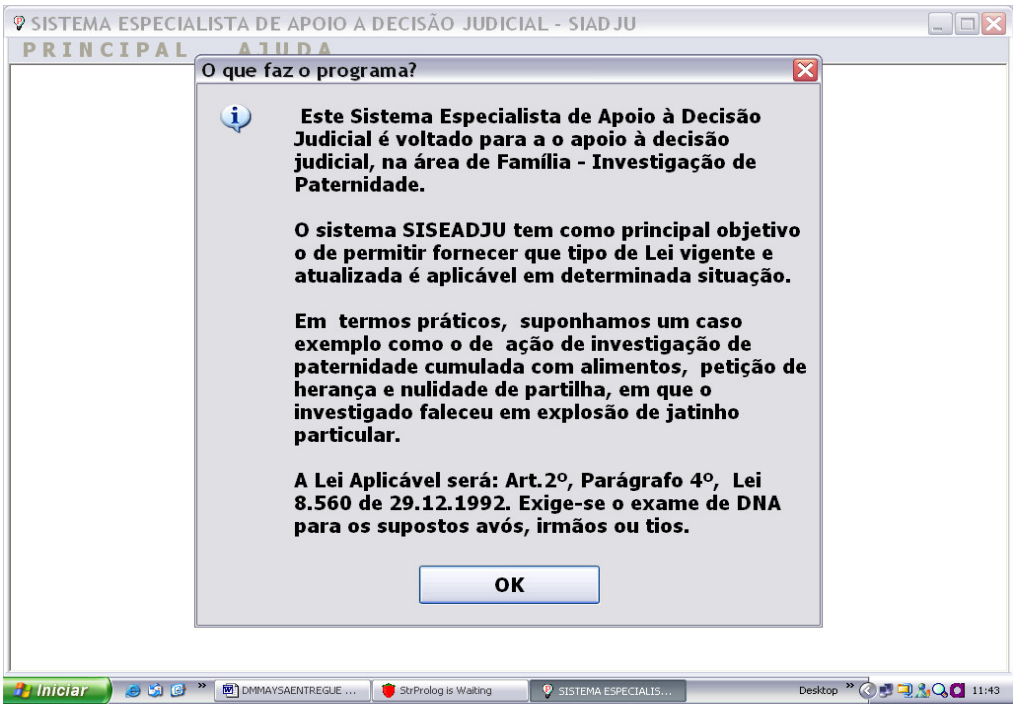

Figura 16: Ajuda do SISEADJU.

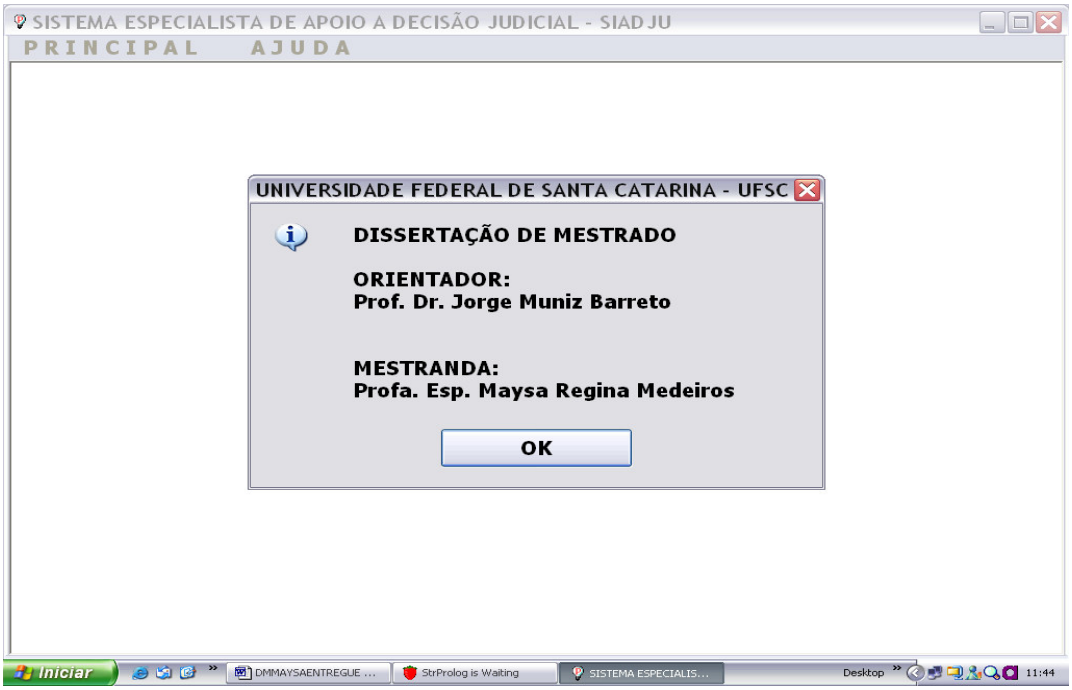

Figura 17: Autor do SISEADJU.

5.4.10. Operacionalidade do protótipo

Quando executado o protótipo, inicialmente encontra-se uma tela com o menu do sistema, onde se pode escolher entre o Autor ou o Réu. A Fig. 18 mostra o menu principal do sistema onde o Autor deseja entrar com um processo de Investigação de Paternidade ou Pensão Alimentícia.

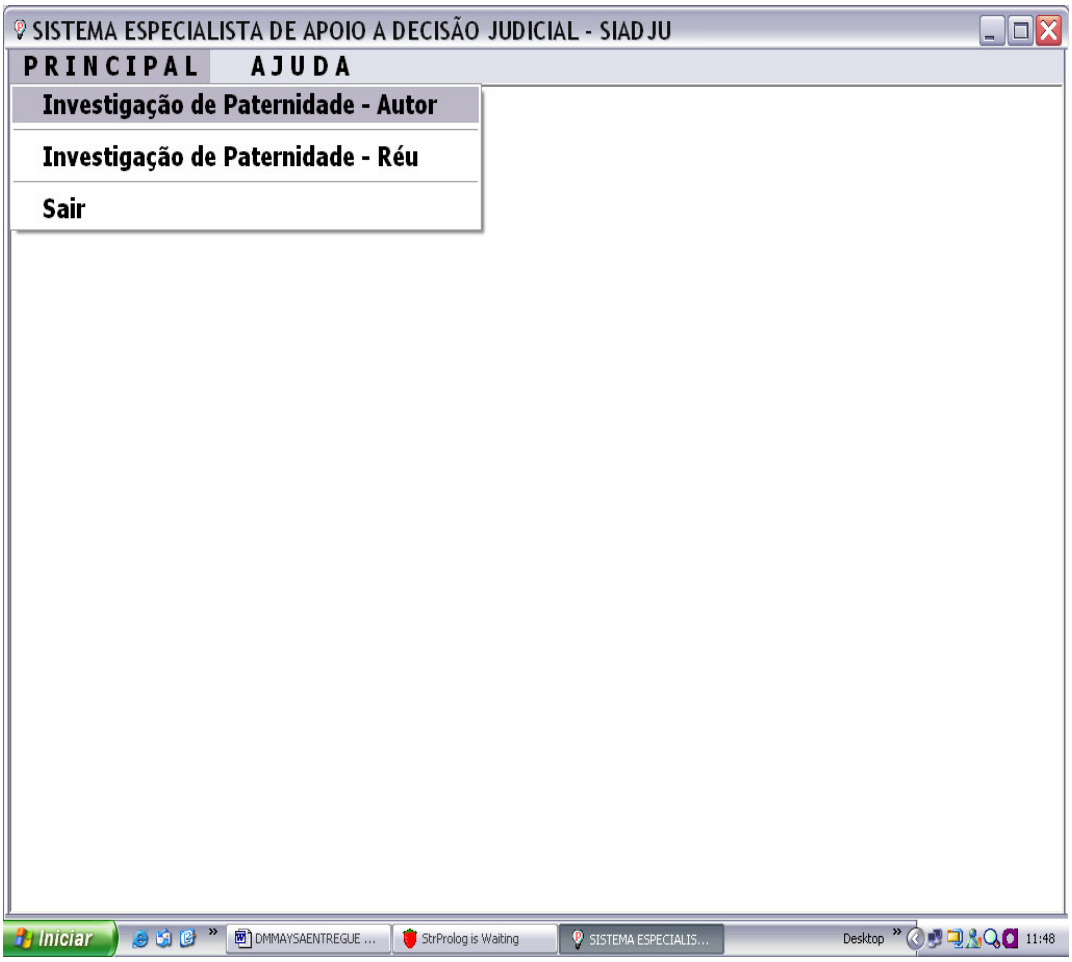

Figura 18: Menu Principal do SISEADJU.

A Fig. 19 e 20 mostra a entrada do sistema através das perguntas a serem realizadas pelo sistema dependendo da resposta (sim/não) o sistema retornará qual será a lei aplicável para determinada situação.

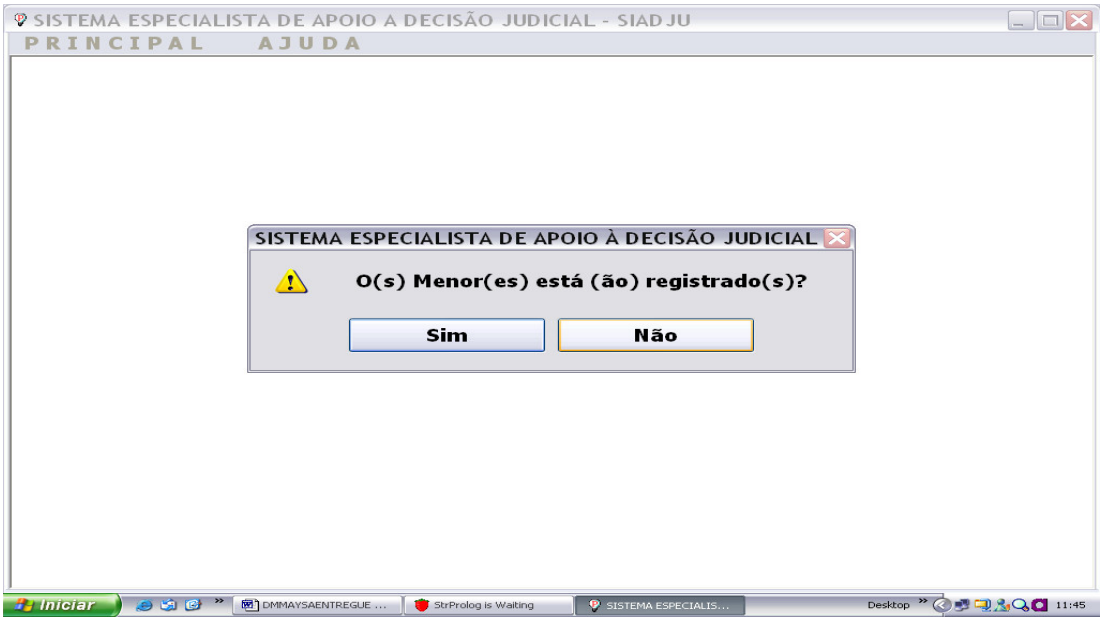

Figura 19: SISEADJU pergunta se o menor está ou não registrado.

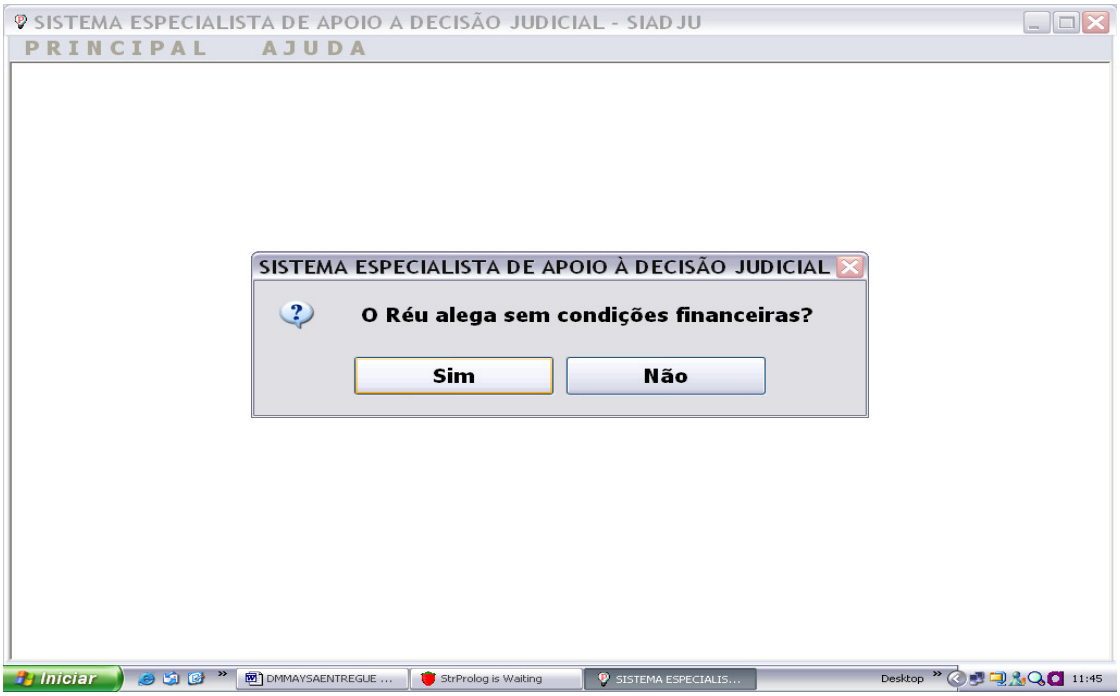

Figura 20: Se o menor está registrado então o réu pode estar alegando falta de condições financeiras.

A Fig. 21 mostra que, o réu alegou falta de condição financeira e o sistema retorna qual a Lei será aplicável.

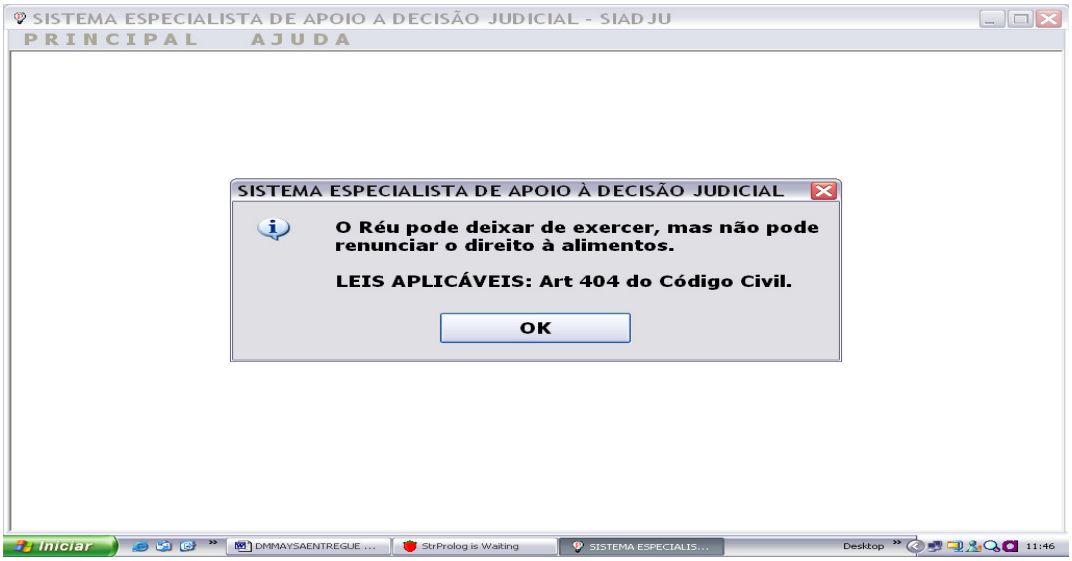

Figura 21: O SISEADJU retorna a Lei aplicável.

A Fig. 22 mostra que o réu não discorda que tem condição financeira e o sistema retorna um apoio a decisão.

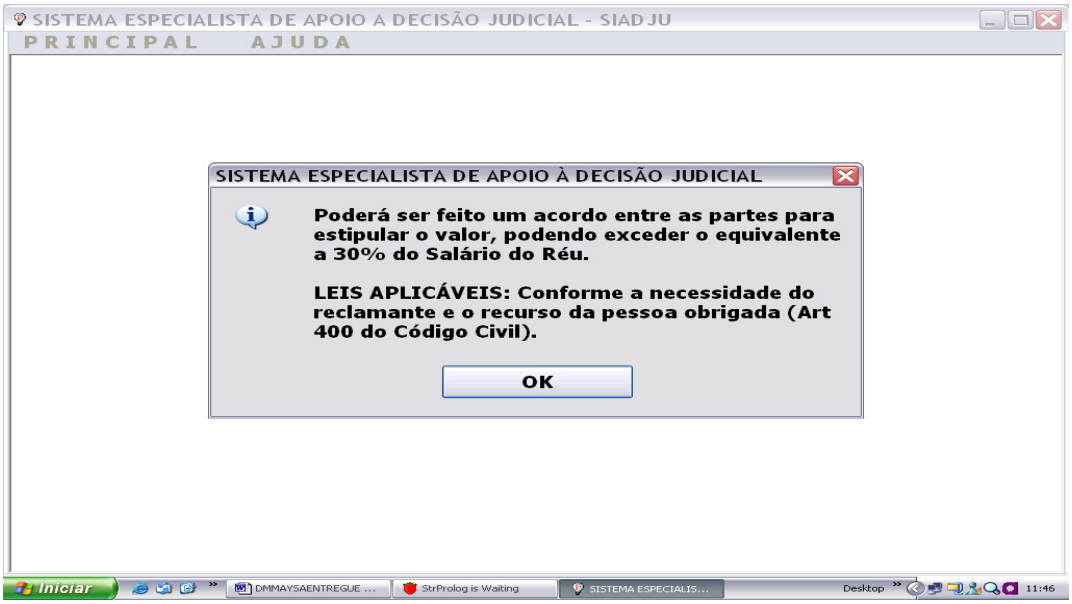

Figura 22: O SISEADJU retorna a Lei aplicável.

As figuras 23 e 24 mostram que, o menor não está registrado, e o se suposto réu está ou não vivo.

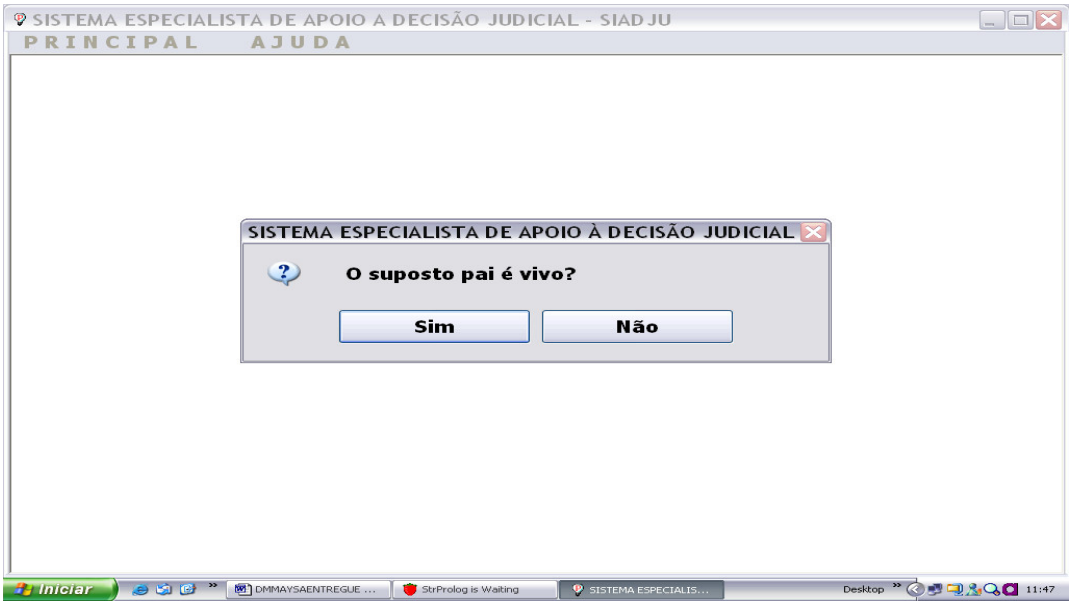

Figura 23: Se o menor não está registrado, o sistema pergunta se o suposto pai é vivo.

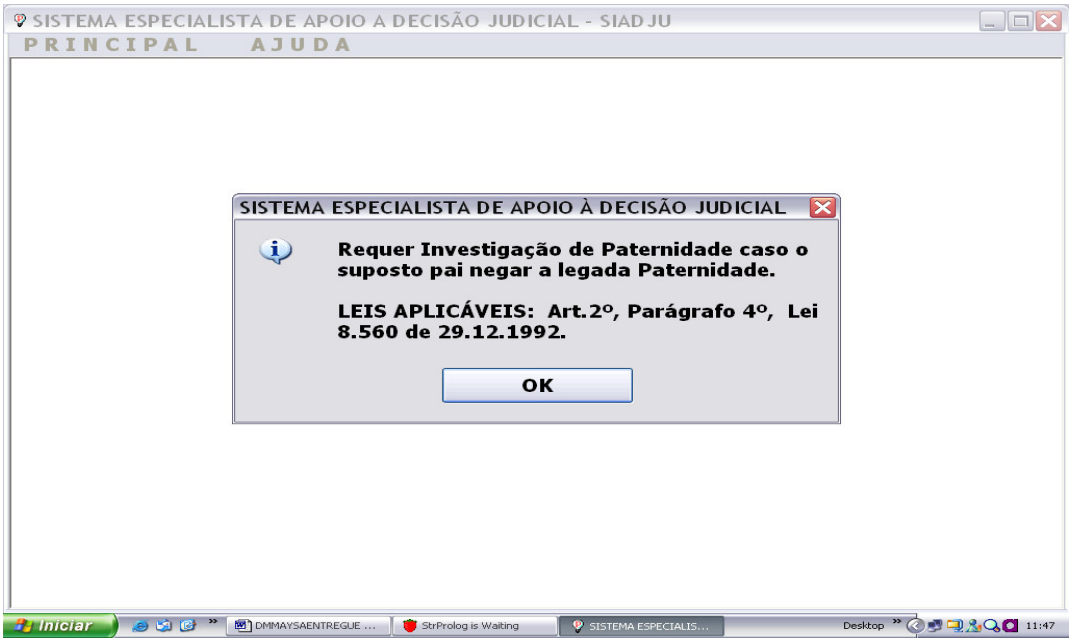

Figura 24: O SISEADJU retorna a Lei aplicável.

A Fig. 25 mostra se o menor não está registrado e o suposto réu é morto, retornando um apoio a decisão.

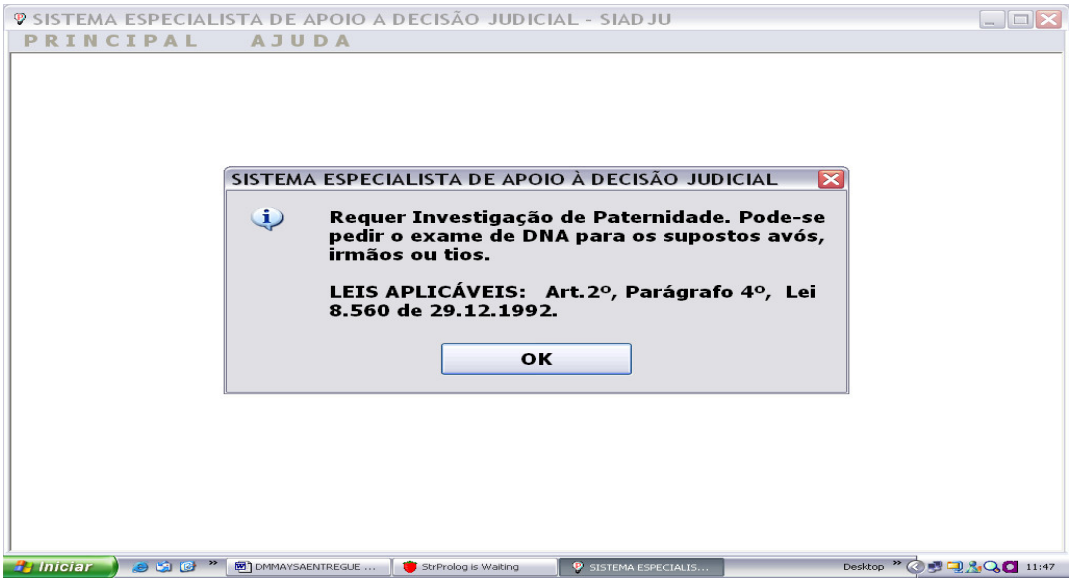

Figura 25: O SISEADJU retorna a Lei aplicável.

A Fig. 26 mostra o menu principal do sistema onde o réu deseja entrar com um processo de Investigação de Paternidade.

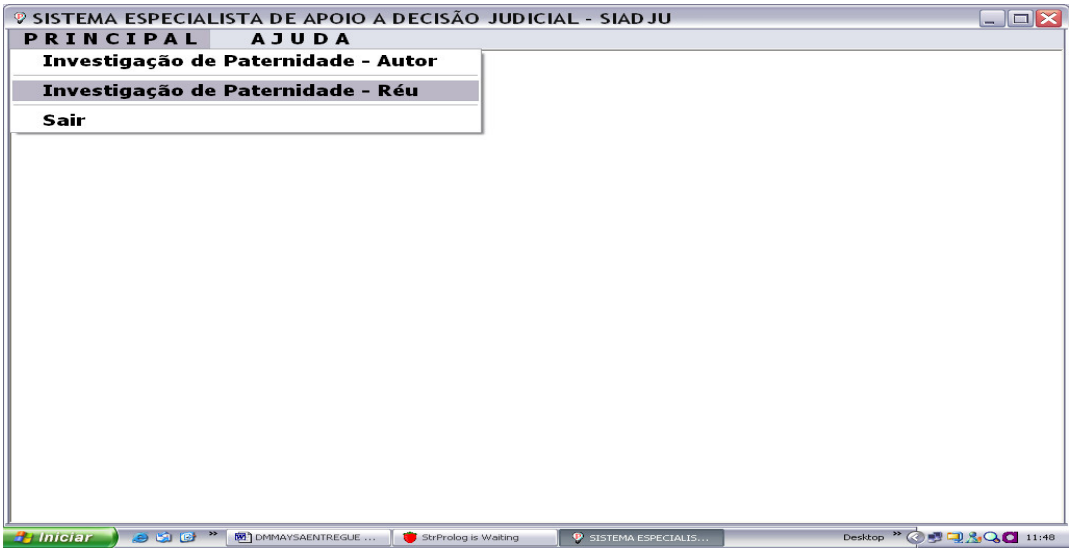

Figura 26: Menu principal do SISEADJU.

As figuras 27 e 28 mostra que, o réu nega ou não a paternidade e o sistema retorna um apoio a decisão.

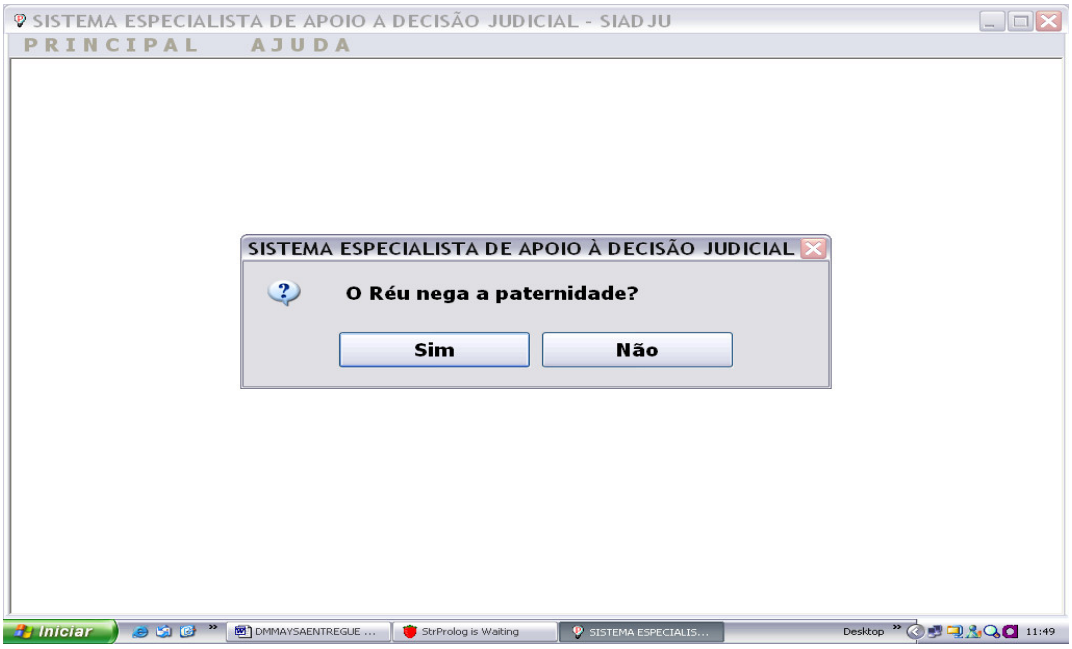

Figura 27: SISEADJU pergunta se o réu nega a paternidade ou não.

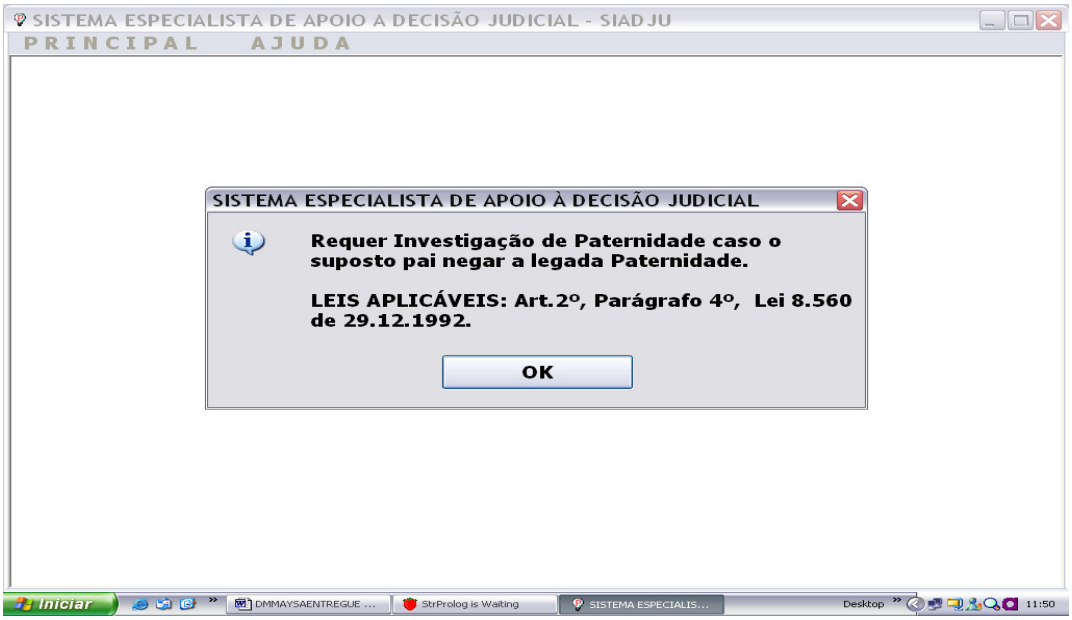

Figura 28: O SISEADJU retorna a Lei aplicável.

A Fig. 29 e 30 mostra que, se o menor está registrado então possivelmente o réu alega ou não condição financeira e o sistema retorna a Lei aplicável.

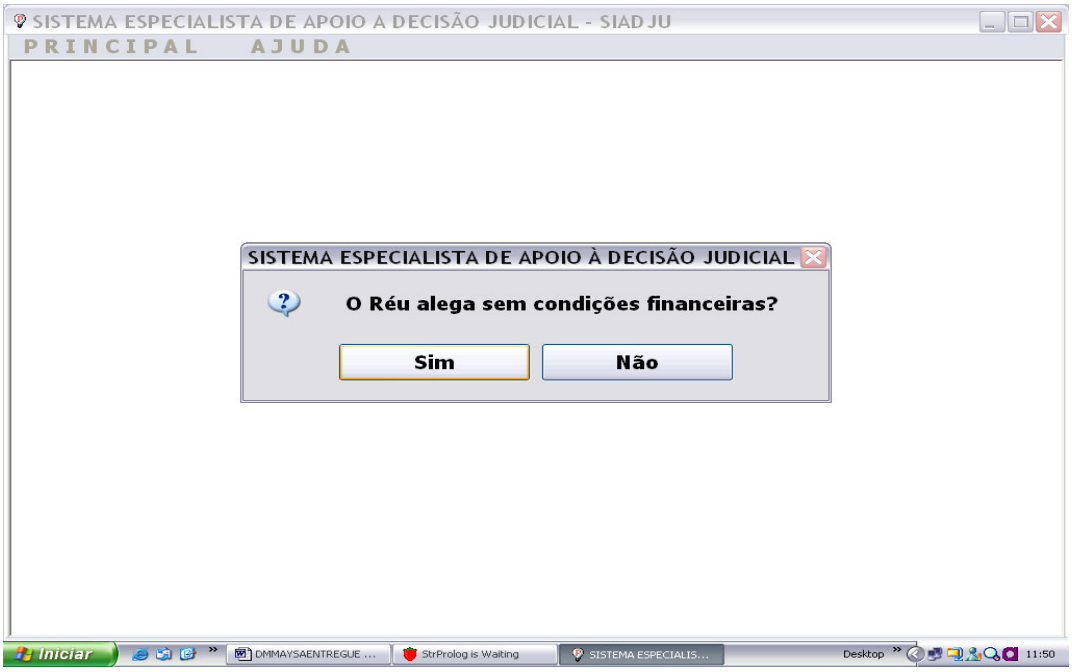

Figura 29: O SISEADJU pergunta se o réu alega ou não falta de condições financeiras.

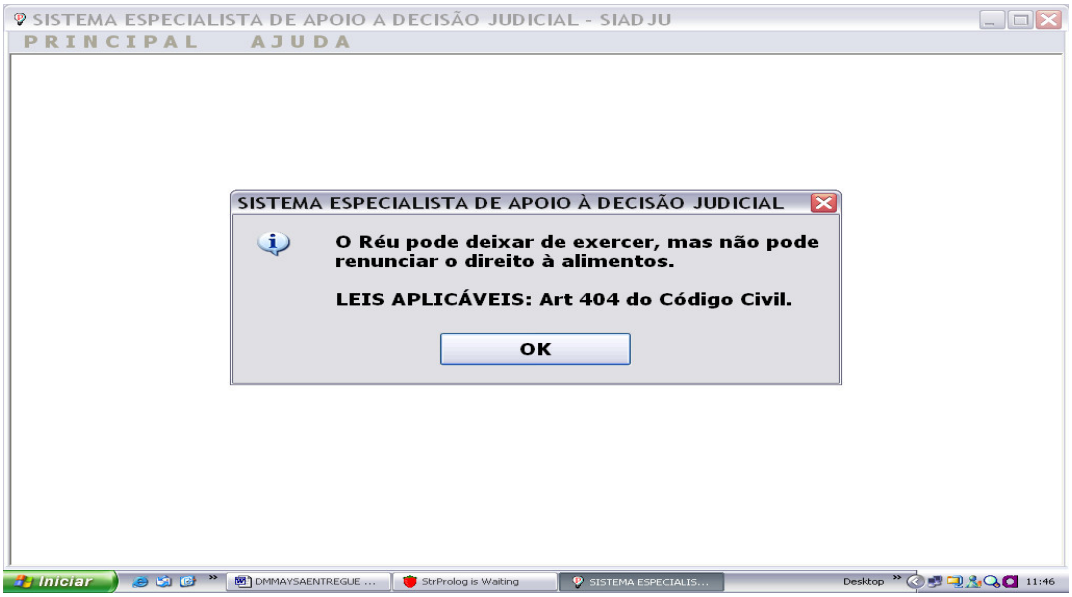

Figura 30: O SISEADJU retorna a Lei aplicável.

A Fig. 31 mostra que o réu não alegou falta de condição financeira para pagar Pensão Alimentícia e o sistema retorna um apoio a decisão.

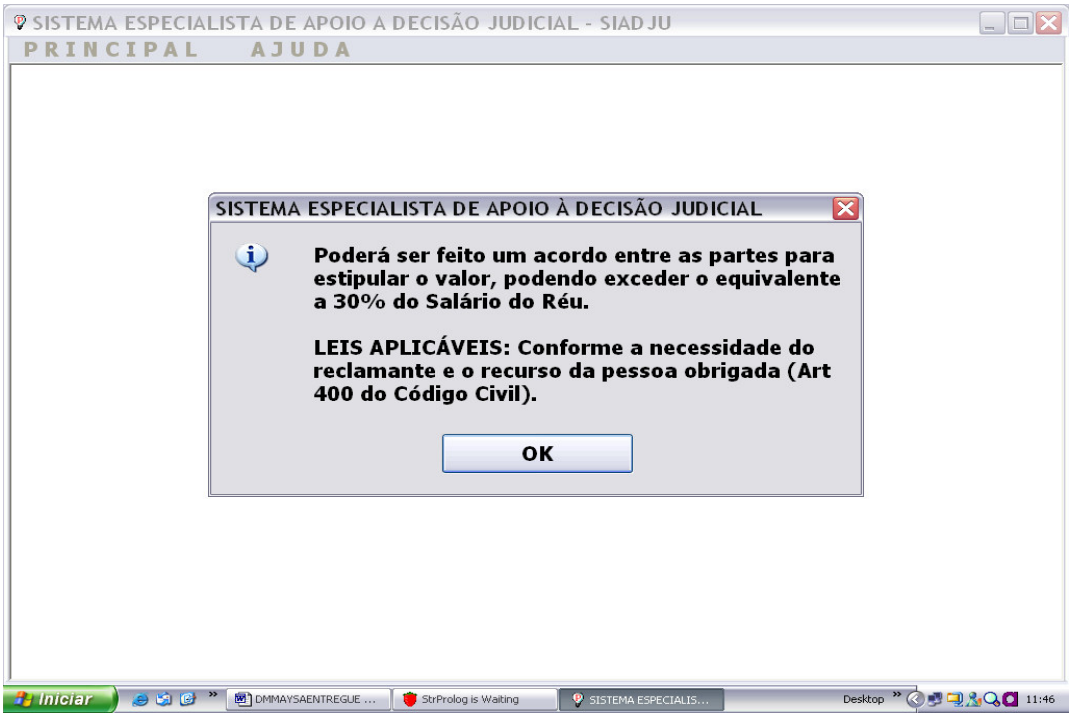

Figura 31: O SISEADJU retorna a Lei aplicável.

5.4.11. Riscos

Diante da análise e do protótipo desenvolvido, a funcionalidade irá atender ao objetivo proposto inicialmente. A tecnologia escolhida foi a ideal e a mais adequada. Entretanto, o processo de desenvolvimento deverá ser flexível e eficiente às novas mudanças que ocorrerão. Será necessário realizar um novo levantamento das ferramentas a serem utilizadas no futuro sistema.

5.5. Fase I - Análise de Oportunidade

5.5.1. Diagnóstico de Reumatologia

Este documento visa ilustrar a MEDSIA para a construção de programas. Com a finalidade de expor e demonstrar a metodologia será desenvolvido um protótipo de um sistema de Diagnóstico em Reumatologia. Será evidenciado somente as Fases I, II, III e IV. Serão apresentados basicamente os resultados parciais da análise, no que tange aos aspectos de informação.

O sistema a ser construído tem a finalidade de ilustrar um exemplo e mostrar a metodologia MEDSIA. Com efeito, o sistema pronto não teria passado da fase de oportunidade, por não conhecer médicos interessados nesse programa.

5.6. Fase II - Definição do Problema

Muitas das doenças em Reumatologia são de evolução crônica e necessitam de tratamento prolongado. Ao contrário do que se diz popularmente não é uma "doença de velho", mas pode acontecer em qualquer idade referenciar (REUMATOLOGIA, 2004).

Os termos "reumatismo" ou "doença reumática", na realidade, nada significam, pois não são diagnósticos. O médico deve procurar identificar qual doença que cada paciente tem. Quando alguém diz que tem artrite significa apenas que tem inflamação da articulação, que pode ser evidenciada por dor, inchaço e calor na junta. A artrite é uma manifestação comum à maioria das doenças reumáticas que comprometem as articulações. A evolução e o diagnóstico são muito variáveis, de doença para doença e de paciente para paciente. Diante disto, o tratamento dependerá do tipo de doença e do paciente em si, do diagnóstico correto, e o tratamento adequado é fundamental (REUMATOLOGIA, 2004).

Portanto, o problema aqui evidenciado constitue-se de diagnóstico em reumatologia, se deve considerar os possíveis tipos de doenças e os sintomas de cada paciente.

5.7. Fase III - Reconhecimento do Problema

Na definição do problema acima descrito levantou-se algumas hipóteses para a escolha da melhor abordagem como a seguir:

- Tem-se uma parte do conhecimento e é possível induzir o restante?
- Tem-se uma amostra de dados?
- Processamento de grandes quantidades de dados pouco estruturados?

No reconhecimento do problema uma solução satisfatória será desenvolver um sistema conexionista para diagnóstico em reumatologia ao qual se dará o nome de Sistema Conexionista para Diagnóstico em Reumatologia - SICDIRE.

5.8. Fase IV – Análise Funcional

5.8.1. Escolha do conteúdo e organização da base de conhecimento conexionista

Será utilizado uma base de exemplos de casos de reumatologia (ver Tabela 9), que servirão para realizar o treinamento e os testes da rede, necessários para as verificações e análises sugeridas no trabalho. Os exemplos foram obtidos de casos clínicos reais da área de reumatologia, vindos do Departamento de Medicina Geral da UNIRIO, através da Professora Lycia Epprecht (DAZZI, 1999), (AZEVEDO, *et. al.*, 2000).

5.8.2. Descrição Lógica de Treinamento do SICDIRE

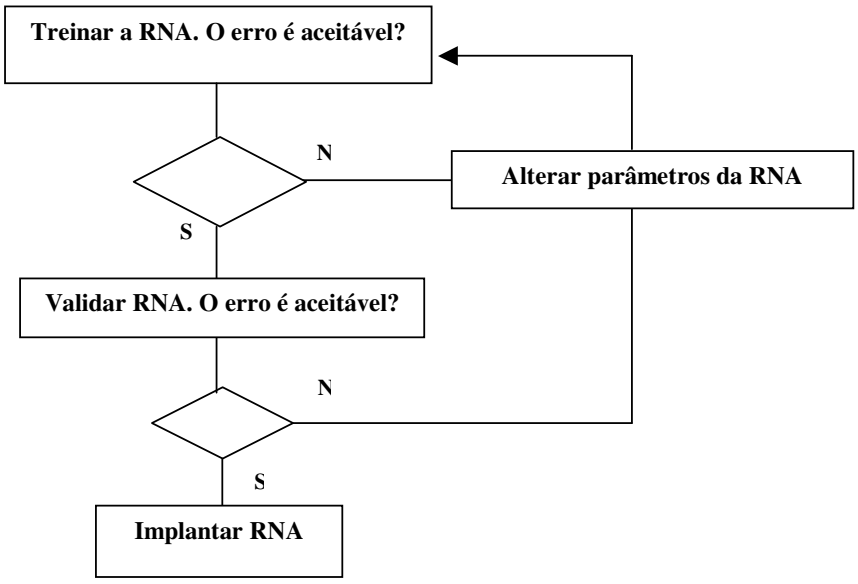

Figura 32: Descrição lógica de treinamento do SICDIRE.

5.8.3. Requisitos Funcionais

Tabela 5: Requisitos funcionais do SICDIRE.

Funcionalidade	Categoria
Cadastrar Sintomas, Exames Clínicos e Diagnósticos.	Evidente
Gerar pesos.	Evidente
Realizar treinamento da RNA.	Evidente
Exibir o resultado do treinamento na tela.	Evidente
Deve prover um mecanismo persistente de armazenamento.	Escondida

5.8.4. Requisitos Não-Funcionais

Tabela 6: Requisitos não-funcionais do SICDIRE.

Ref.	Funcionalidade
Tempo de resposta	Ao realizar treinamento dados devem ser visualizados em até 10 segundos.
Tipo de Interface desejada	Usar caixa de diálogos. Maximizar a facilidade do uso de teclado.
Plataformas operacionais	**Configuração Mínima:** Micro computador xx 233 MHz ou equivalente. 64 Mb RAM ou superior. HD 8 GB ou superior. Sistema Operacional. Impressora jato de tinta ou equivalente. **Configuração desejada:** Micro computador xx2.66 MHz. 128 Mb RAM ou superior. HD 40 GB ou superior. Sistema Operacional. Impressora jato de tinta ou equivalente.
Desempenho	O tempo de resposta garantirá uma consulta confiável, rápida e segura.

Serão selecionados alguns elementos da UML (*Unified Modeling Language*) para melhor expressar a modelagem. A ferramenta de modelagem visual escolhida foi o Rational Rose 2000 para fornecer informações importantes antes de gerar os códigos.

5.8.5. Diagrama de Contexto

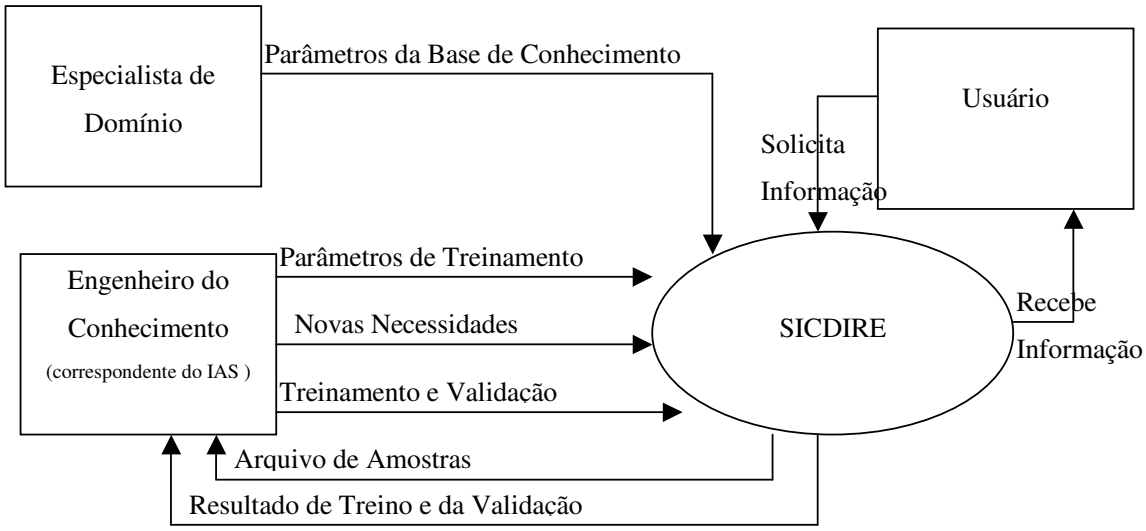

Figura 33: Diagrama de Contexto do SICDIRE

5.8.6. Dicionário de Dados

Diagnóstico: representa diagnósticos

- codigodiagnosticos: código do diagnóstico, composto por números.
- obsdiagnosticos:

Sintoma: representa os sintomas conhecidos de pacientes.

- codigosintoma
- obssintoma

Exame Clínico: representa os exames realizados por pacientes

- codigoexame
- obsexame

Paciente: classe abstrata que representa o paciente.

- codigopaciente
- nomepaciente

- enderecopaciente
- codigocidade
- fonepaciente

Especialista: classe abstrata que representa o especialista que irá utilizar o sistema.

- codigoespecialista
- nomeespecialista
- enderecoespecialista
- codigocidade
- foneespecialista

Engenheiro do conhecimento: classe abstrata que representa o engenheiro que irá utilizar o sistema.

- codigoengenheiro
- nomeengenheiro
- enderecoengenheiro
- codigocidade
- foneengenheiro

Cidade: representa as cidades do nosso país.

- codigocidade
- nomecidade
- ufcidade

5.8.7. Modelo de Classes

A modelagem de classes envolve a identificação de classes, atributos, associações e operações. A seguir, são apresentados os resultados da análise. A Fig. 34 apresenta o Diagrama de Classes para um melhor entendimento do sistema.

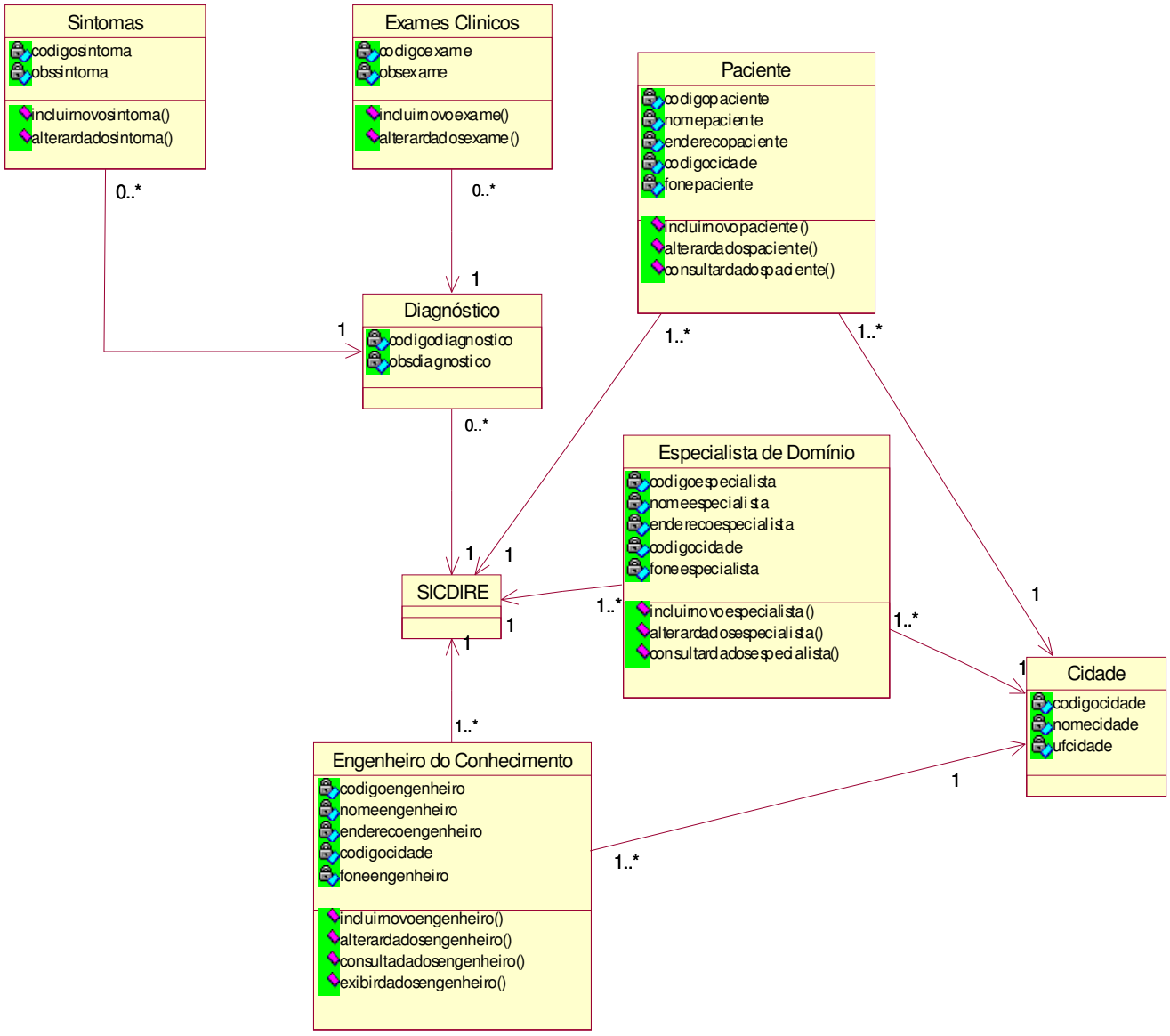

Figura 34: Diagrama de Classes do SIDIRE.

5.8.8. Descrição das patologias investigadas

Apenas didaticamente, serão classificadas aqui as doenças reumáticas, de acordo com os mecanismos de lesão ou localização preferencial da doença conforme se mostra a seguir.

5.8.9. Doenças difusas do tecido conjuntivo

Doenças que cursam com inflamação do tecido conjuntivo e que estão relacionadas aos distúrbios do sistema imunológico, que passam a reagir contra uma célula, tecido ou outro antígeno do próprio organismo (REUMATOLOGIA, 2004).

A Artrite Reumatóide é uma doença autoimune na qual as juntas, geralmente aquelas das mãos e pés, tornam-se, simetricamente, inflamadas, resultando em inchaço, dor e eventual destruição do interior da junta (REUMATOLOGIA, 2004).

Pode produzir vários sintomas como febre baixa, inflamação dos vasos sangüíneos (vasculite), podendo causar úlceras de perna ou danos nos nervos, pericardite e outros sintomas. Sua causa ainda é desconhecida, porém diversos fatores diferentes, como a predisposição genética, podem influenciar a reação autoimune. Nesta doença, o sistema imune ataca o tecido que reveste e amortece as juntas (REUMATOLOGIA, 2004).

5.8.10. Espondilartropatias

Doença inflamatória da coluna vertebral podendo ou não causar artrite em articulações periféricas e inflamação em outros órgãos como o olho(REUMATOLOGIA, 2004).

A Espondilite Anquilosante envolve a inflamação de uma ou mais vértebras. É uma doença crônica inflamatória que afeta as juntas entre as vértebras da espinha e as juntas entre a espinha e a pélvis. Eventualmente, faz com que as vértebras afetadas se fundam ou cresçam juntas(REUMATOLOGIA, 2004).

5.8.11. Artropatias Microcristalinas

Doenças articulares causadas por microcristais.

A Artrite de Gota é um distúrbio caracterizado por ataques repentinos e intermitentes, muito dolorosos, de artrite, causado pelo depósito de cristais de urato de sódio. Os cristais se acumulam nas juntas devido a níveis de ácido úrico anormalmente altos nas mesmas, advindos de níveis elevados do mesmo ácido no sangue. A inflamação das juntas pode se tornar crônica e causar deformações após ataques repetidos (REUMATOLOGIA, 2004).

5.8.12. Levantamento dos Casos Clínicos

Os conjuntos de exemplos foram divididos em três classes: 3 Patologias, 14 Sintomas e 3 Exames Laboratoriais, como se mostra a seguir.

5.8.13. Diagnósticos

Artrite Reumatóide = AR

Artrite de Gota = AG

Espondilite Anquilosante = EA

Nenhuma Patologia

5.8.14. Sintomas

Dor Lombar = DL

Rigidez na coluna = RC

Deformação na coluna = DC

Mobilidade = M

Dor ao toque no Sacroiliaco = DTS

Artrite = AR

Rigidez Matinal = RM

Bursite = B

Tophi = T

Sinovite = S

Artralgia = ART

Nódulos Reumatóides = NR

HLA-B27 = HLA

Deformação nas Juntas = DJ

5.8.15. Exames Clínicos

Inflamação Laboratorial = IL

Evidências Radiológicas = ER

Tomografia Computadorizada = TC

5.8.16. Descrição dos Dados Utilizados

Serão organizados os dados de forma que um paciente apresenta um sintoma com uma certa gravidade, o que poderá ter um diagnóstico com mais de uma doença. Foram coletados os dados relativos ao problema e a sua separação em um conjunto de treinamento e um conjunto de testes. Os dados de treinamento serão utilizados para o treinamento da rede e os dados de testes para verificar a performance da rede. Os exemplos serão divididos em 2 conjuntos, um para o treinamento da rede, com 30 casos cuidadosamente analisados e outro com 20 restantes para teste.

Tabela 7: Base de Exemplos do SICDIRE.

Caso	DL	RC	DC	M	DTS	IL	ER	TC	Diagnóstico	AR	RM	B	T	S	ART	NR	HLA-B27	DJ
1	sim	sim	sim	limitado	sim	ausente	moderado	moderado	Espondilite Anquilosante	não	não	não	não	não	não	não	neg	não
2	não	não	não	normal	não	ausente	moderado	não	Artrite de Gota	sim	não	não	não	não	sim	não	não	não
3	não	não	não	limitado	não	moderado	moderado	não	Artrite de Gota	sim	sim	sim	sim	sim	sim	não	não	não
4	não	não	não	limitado	não	ausente	moderado	não	Artrite de Gota	não	não	não	sim	não	sim	não	não	não
5	não	não	não	normal	não	moderado	importante	moderado	Artrite Reumatóide	não	não	não	não	não	sim	não	não	não
6	não	não	não	limitado	não	muito importante	muito importante	importante	Artrite de Gota	sim	não	não	sim	sim	sim	não	não	não
7	não	não	não	limitado	não	moderado	muito importante	não	Artrite Reumatóide	sim	não	não	não	sim	sim	não	não	não
8	sim	não	sim	normal	sim	moderado	importante	importante	Espondilite Anquilosante	não	não	não	não	não	não	não	pos	não
9	sim	sim	não	limitado	sim	importante	muito importante	importante	Artrite Reumatóide	sim	sim	não	não	sim	sim	não	não	não
10	não	não	não	limitado	não	muito importante	ausente	moderado	Artrite Reumatóide	sim	sim	não	não	sim	sim	sim	não	não
11	sim	sim	sim	limitado	sim	moderado	importante	não	Espondilite Anquilosante	não	não	não	não	não	não	não	pos	não
12	não	não	não	limitado	não	leve	moderado	não	Artrite Reumatóide	sim	não	não	não	sim	sim	não	não	não

Tabela 8: Base de Exemplos do SICDIRE (continuação).

Caso	DL	RC	DC	M	DTS	IL	ER	TC	Diagnóstico	AR	RM	B	T	S	ART	NR	HLA-B27	DJ
13	não	não	não	limitado	não	muito importante	ausente	não	Artrite de Gota	sim	não	não	não	sim	sim	não	não	não
14	não	não	não	limitado	não	importante	ausente	leve	Artrite Reumatóide	sim	sim	não	não	sim	sim	não	neg	não
15	não	não	não	normal	não	não	não	não	Nenhum	não	não	não	não	não	não	não	não	não
16	sim	não	não	normal	não	leve	muito importante	não	Espondilite Anquilosante	não	não	não	não	não	não	não	neg	não
17	não	não	não	limitado	não	moderado	moderado	não	Artrite de Gota	sim	não	não	sim	não	sim	não	não	não
18	não	não	não	normal	não	leve	importante	não	Artrite Reumatóide	sim	sim	não	não	sim	sim	não	não	não
19	sim	sim	sim	limitado	não	moderado	muito importante	moderado	Espondilite Anquilosante	sim	não	não	não	não	sim	não	pos	sim
20	sim	não	não	normal	não	moderado	ausente	não	Espondilite Anquilosante	sim	não	não	não	sim	sim	não	pos	não
21	não	não	não	limitado	não	importante	ausente	moderado	Artrite Reumatóide	sim	sim	não	não	sim	sim	não	não	sim
22	não	não	não	normal	não	leve	importante	não	Artrite de Gota	sim	não	não	sim	sim	sim	não	não	não
23	sim	não	não	normal	não	ausente	moderado	moderado	Espondilite Anquilosante	não	não	não	não	não	sim	não	pos	não
24	sim	sim	não	normal	sim	ausente	moderado	leve	Espondilite Anquilosante	não	sim	não	não	não	não	não	pos	não
25	sim	não	sim	normal	não	ausente	importante	não	Espondilite Anquilosante	não	não	não	não	não	não	não	pos	não

Tabela 9: Base de Exemplos do SICDIRE (continuação).

Caso	DL	RC	DC	M	DTS	IL	ER	TC	Diagnóstico	AR	RM	B	T	S	ART	NR	HLA-B27	DJ
26	não	não	não	normal	não	importante	ausente	não	Artrite Reumatóide	não	não	sim	não	não	sim	sim	não	não
27	sim	sim	não	normal	não	leve	importante	importante	Espondilite Anquilosante	não	não	não	não	não	não	não	pos	não
28	sim	sim	não	normal	não	ausente	ausente	não	Artrite Reumatóide	não	sim	não	não	não	sim	não	não	não
29	não	não	não	limitado	não	moderado	moderado	moderado	Artrite de Gota	sim	não	não	não	não	sim	não	não	não
30	não	não	não	limitado	não	importante	importante	moderado	Artrite de Gota	sim	não	sim	sim	sim	sim	não	não	não
31	sim	sim	sim	normal	não	ausente	importante	não	Espondilite Anquilosante	não	não	não	não	não	não	não	pos	não
32	não	não	não	normal	não	importante	importante	não	Artrite de Gota	sim	sim	sim	sim	não	sim	não	não	não
33	sim	não	não	normal	sim	moderado	importante	não	Espondilite Anquilosante	não	sim	não	não	não	não	não	pos	não
34	não	não	não	limitado	não	leve	importante	não	Artrite Reumatóide	sim	sim	não	não	sim	sim	não	não	não
35	não	não	não	limitado	não	leve	moderado	não	Artrite Reumatóide	sim	sim	não	não	sim	sim	não	não	não
36	sim	sim	não	normal	sim	ausente	importante	importante	Espondilite Anquilosante	não	sim	não	não	não	sim	não	pos	não
37	não	não	não	normal	não	importante	moderado	não	Artrite de Gota	sim	não	sim	sim	sim	sim	não	não	não
38	não	não	não	limitado	não	moderado	ausente	não	Artrite de Gota	não	não	sim	sim	não	não	não	não	não
39	não	não	não	limitado	não	moderado	importante	não	Artrite Reumatóide	sim	não	não	não	sim	sim	não	não	não

Tabela 10: Base de Exemplos do SICDIRE (continuação).

Caso	DL	RC	DC	M	DTS	IL	ER	TC	Diagnóstico	AR	RM	B	T	S	ART	NR	HLA-B27	DJ
40	não	não	não	normal	não	importante	importante	não	Artrite Reumatóide	sim	não	não	não	sim	sim	não	neg	não
41	não	não	não	limitado	não	importante	moderado	moderado	Artrite Reumatóide	sim	não	não	não	sim	sim	não	não	não
42	sim	não	não	normal	não	moderado	importante	não	Espondilite Anquilosante	não	não	não	não	não	não	não	pos	não
43	não	não	não	normal	não	importante	importante	não	Artrite Reumatóide	não	sim	não	não	não	sim	não	não	não
44	não	não	não	limitado	não	ausente	ausente	não	Artrite Reumatóide	sim	não	não	não	sim	sim	não	neg	não
45	sim	não	não	normal	não	leve	importante	não	Artrite de Gota	sim	não	não	não	sim	sim	não	não	não
46	sim	sim	não	limitado	sim	leve	importante	não	Espondilite Anquilosante	não	sim	não	não	não	não	não	neg	não
47	não	não	não	limitado	não	importante	moderado	não	Artrite Reumatóide	sim	sim	não	não	sim	sim	não	não	não
48	não	não	não	normal	não	moderado	importante	não	Artrite Reumatóide	não	sim	não	não	não	sim	não	não	não
49	sim	sim	não	limitado	não	importante	moderado	não	Espondilite Anquilosante	não	não	não	não	não	não	não	pos	não
50	sim	não	não	normal	não	ausente	moderado	moderado	Artrite de Gota	não	não	não	não	não	sim	não	não	não

5.8.17. Escolha da Ferramenta de desenvolvimento

Definiu-se o simulador *QwikNet* para verificar o comportamento da rede neural com aprendizado retropropagação (JENSEN, 2004).

O simulador *QwikNet* possibilita gravação e restauração dos pesos e previsões obtidas pela rede neural durante uma simulação, gravação interativa dos resultados e amostragem da margem de erro máximo e margem de erro total. Oferece gráficos para o acompanhamento e resultado do aprendizado da rede. As características do simulador são:

- Learning Rate: A taxa de aprendizado controla a taxa à qual a rede aprende. Normalmente, quanto mais alta a taxa de aprendizagem, mais rápido a rede aprende. Se a taxa de aprendizado é muito alta, a rede pode ficar instável.
- Momentum: Este parâmetro controla a influência da última mudança de peso na atualização de peso atual. Também conhecida com introdução da inércia ou momento. Geralmente para garantir o sucesso do aprendizado devemos usar um momento baixo.
- Weights: Os campos Mínimo e Máximo permitem ao usuário fixar os mais baixos e os mais altos valores para cada peso. Durante o treinamento, todos os pesos são usados conforme estes parâmetros.
- Hidden Layers: A rede pode conter até 5 camadas escondidas, cada uma com qualquer número de neurônios. Cada camada escondida pode ter qualquer função de ativação: Linear, Logística, Tangente Hiperbólica, ou Gaussiano. A escolha da função de ativação depende da natureza dos dados de treinamento. Logística é a escolha mais comum, mas a Tangente Hiperbólica podem resultar em treinamentos mais rápidos para muitos problemas.

O QwikNet possibilita:

- gravação e restauração dos pesos e previsões obtidas pela rede neural durante uma simulação;
- gravação interativa dos resultados;

- gráficos para o acompanhamento e resultado do aprendizado da rede.

5.8.18. Parâmetros utilizados na rede

Será utilizado como parâmetro uma rede direta implementada no simulador QwikNet2.23. Constituiu-se para usar a rede neural com 20 registros da base para testes e 30 registros para o treinamento. Para ilustrar, a Fig. 35 exibe o gráfico da rede. Este é definido com suas camadas de entrada, que é representada por quadrados, a camada intermediária e de saída representada por círculos e os triângulos representam o *bias* (desvio) para cada neurônio.

A topologia da rede será definida por: camada de entrada com 17 neurônios de entrada que corresponderá aos sintomas e exames clínicos; camada interna com 10 neurônios, para extrair as características da rede; camada de saída com 4 neurônios; três neurônios que corresponde ao tipo de doença, e um neurônio que não apresenta nenhuma doença.

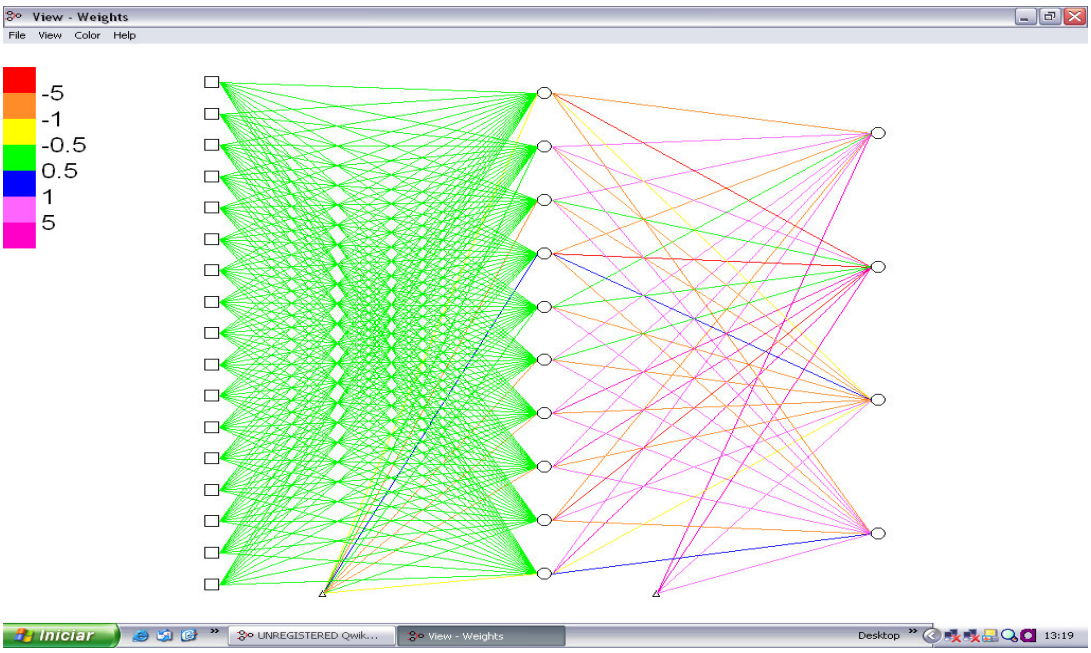

Figura 35: Topologia da rede do SICDIRE com seus pesos definidos.

Na Fig. 36 se mostra o gráfico em que foi obtida no treinamento da rede. Neste gráfico notou-se que o acerto da rede neural estabilizou depois de um certo número de épocas.

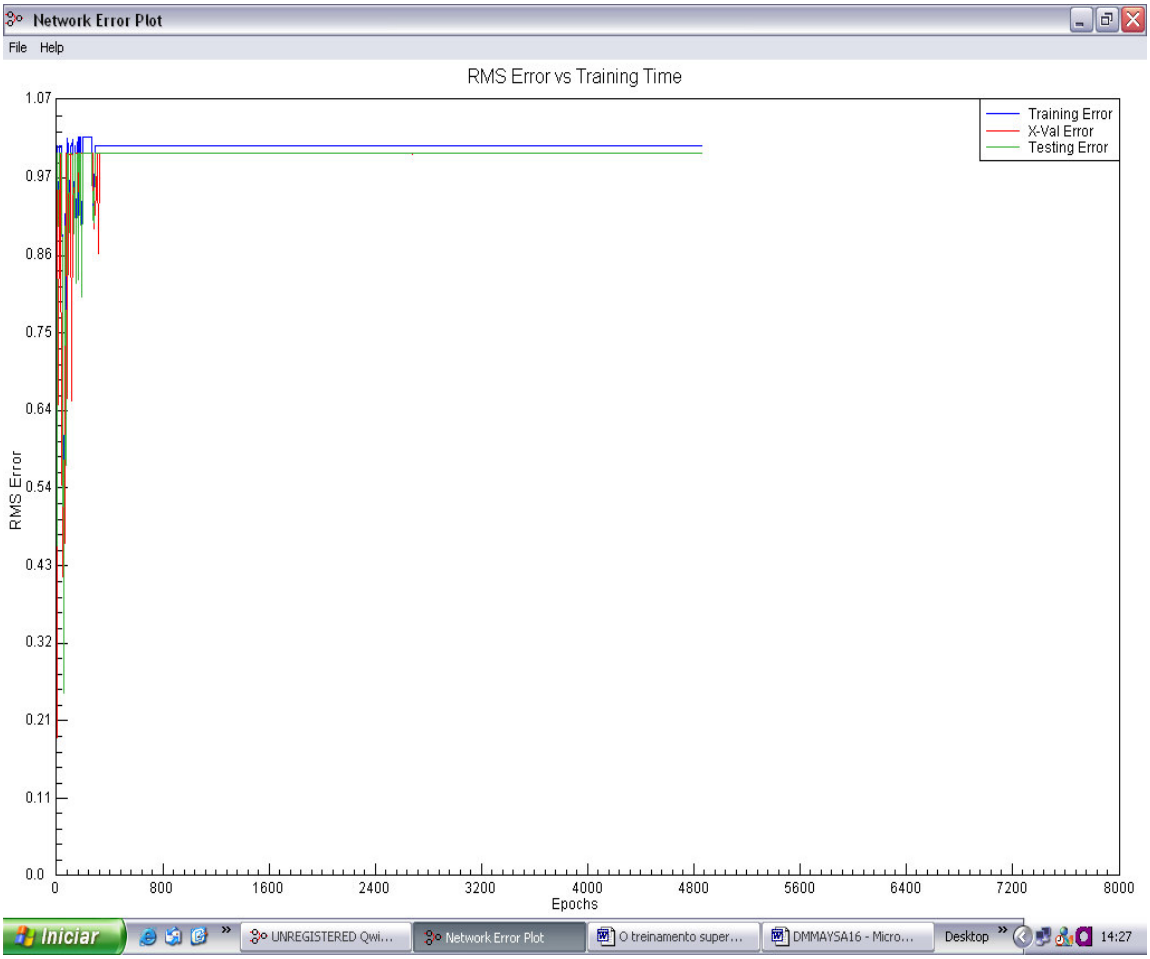

Figura 36: Gráfico do erro, obtido no treinamento do SICDIRE.

Não foi analisado detalhadamente o resultado de outros gráficos, já que, aqui não se está interessado em ter um sistema funcionando, mas sim, demonstrar a metodologia proposta.

6. CONSIDERAÇÕES FINAIS

6.1. Resultados Obtidos

Neste trabalho mostrou-se que, no desenvolvimento de sistemas, deve ser dispendido algum esforço antes da implementação. Normalmente os desenvolvedores começam o ciclo de vida do *software* pela etapa de programação, que deveria ser umas das "últimas" etapas. Dessa forma o desenvolvimento fica resumido à programação comprometendo a qualidade do sistema.

Embora a maioria dos profissionais de informática envolvida com desenvolvimento conheça uma ou mais metodologias, raramente as etapas dessas abordagens são satisfatoriamente concluídas, ficando todo o esforço nas etapas de implementação, o qual deveria ser minimizado com o cumprimento de etapas iniciais de um modelo de ciclo de vida.

Pellegrini (1995) diz: "Fazendo uma análise da equipe de desenvolvimento e dos usuários pode-se dizer que a maioria dos que estão envolvidos na construção de sistemas são acadêmicos e estudantes de mestrado e doutorado. Se o sistema chega a ser implementado, geralmente é com a intenção de avaliar o desempenho do modelo teórico, e quando isto ocorre, normalmente é negligenciado a aceitação do usuário, o desempenho do sistema, a documentação e sua manutenção".

Se todas as etapas fossem cumpridas os problemas seriam drasticamente reduzidos minimizando as conseqüências decorrentes da não utilização ou utilização inadequada de uma metodologia de desenvolvimento de *software*.

As causas mais comuns para o fracasso dos projetos são: falta de planejamento, informações do mercado e dos usuários, requisitos incompletos, dentre outros. O resultado conduz a tomada de decisões eficazes, reduz surpresas desagradáveis, menos improviso e custos menores.

As dificuldades encontradas na elaboração deste trabalho refletem as dificuldades que se encontra ao se aplicar metodologias de Engenharia de *Software*. No contexto, foi um desenvolvimento detalhado, com valorização nas etapas da Análise de

Oportunidade, Definição do Problema, Reconhecimento do Problema e da Ferramenta para resolvê-lo, Análise Funcional, Implementação, Validação, Implantação, Manutenção e Morte.

Este projeto foi desenvolvido a partir dos resultados de um estudo teórico, objetivando compreender a sistemática da engenharia de *software* no desenvolvimento de sistemas. A partir da análise, permitiu-se definir algumas características que demonstram uma relação entre a base teórica e a condição empírica observada, indicando quais características servem de parâmetro inicial para o desenvolvimento da metodologia.

Assim, MEDSIA é uma forma sistematizada, organizada e lógica de gerenciar adequadamente um desenvolvimento de sistemas voltado para a IA. Esta nova metodologia coloca a disposição a possibilidade de usufruir, mesmo que de forma experimental os conceitos debatidos e avaliar no futuro seus impactos.

Com a finalidade de expor a metodologia e para o total entendimento, ela foi mostrada através de dois exemplos: uma ilustração de um Sistema Especialista para Apoio à Decisão Judicial e um Sistema Conexionista para Diagnóstico de Reumatologia. Devido à natureza iterativa, partes da implementação foram sendo feitas construindo um protótipo, permitindo assim um ciclo de implementação mais rápido e eficiente.

Chegou-se somente até a fase de Análise Funcional, pelo fato do tópico ser muito extenso para ser explanado num todo. Com isto, foi possível demonstrar que desenvolver um projeto amparado em métodos e técnicas bem documentadas é uma tarefa árdua, mas tal esforço permite avaliar o projeto antes mesmo da execução e definir se haverá possibilidade de continuar ou não.

6.2. Conclusão

Disciplinar o uso e o desenvolvimento de sistemas é uma necessidade iminente. É preciso definir metodologias (conjunto de ferramentas, processos e métodos) que se adequem a esse novo panorama, tendo como objetivo primordial o desenvolvimento de programas de qualidade.

A Engenharia de *Software* é muito mais que gerenciar custos, riscos e documentação: é a aplicação prática do conhecimento científico do projeto. Pensando neste contexto, a escolha de uma metodologia de desenvolvimento de programa detalha o ciclo de vida com a finalidade de otimizar a execução das atividades nele especificadas e como resultado se obtém produtividade, qualidade, eficiência e eficácia.

Considerando os parâmetros citados nos capítulos anteriores, se chegou à conclusão que nenhum dos modelos atingiram eficientemente o objetivo do trabalho, havendo, portanto, necessidade de se desenvolver uma metodologia que seja eficiente no desenvolvimento de sistemas voltados para a IA.

A metodologia ora proposta visou aproveitar os pontos mais significantes e eliminar os pontos negativos das outras abordagens existentes. É similar ao modelo de ciclo de vida iterativa e incremental. Pode-se desenvolver um protótipo logo na quarta fase a fim de se avaliar o sistema a ser produzido. Permite-se assim, descobrir problemas potenciais antes de se comprometer com um sistema completo. Os objetivos de cada etapa também são semelhantes àqueles propostos por outros modelos.

A maior característica da metodologia se encontra nas Fases I, II e III, onde se faz uma análise de mercado, tem-se uma definição do problema, o reconhecimento do problema e da ferramenta para resolvê-lo, antes mesmo do planejamento do projeto. Caso seja inviável, já será possível tomar decisões e avaliar se vale a pena continuar o projeto. E é esta a principal diferença de outras metodologias existentes. Nas fases I e II é recomendável tratar o nível de risco ou “evento futuro” do projeto. Esta é uma atividade muito importante, sendo necessário o gerenciamento de riscos ao longo de todo o projeto, sempre que for necessário.

Poderá ser utilizada em qualquer projeto, independente da plataforma, pois ao reconhecer o problema, se poderá escolher qual a abordagem mais apropriada, sendo, portanto reutilizável alguma atividade independente do problema a ser solucionado.

O objetivo final proposto por este trabalho foi atingido, com a criação de uma metodologia e uma demonstração documentada, chegando-se ao desenvolvimento de um protótipo. Apresentou-se uma visão típica de práticas utilizadas no desenvolvimento de sistemas, oferecidas como ponto de partida para discussão e experimentação de um processo conveniente às necessidades do projeto.

A MEDSIA permite visualizar e compreender o processo de desenvolvimento, permitindo assim tomadas de decisões, ser flexível para resolver problemas satisfatoriamente, as questões serem identificadas em tempo, ter definição dos pontos essenciais, planejar e acompanhar o progresso do projeto, estabelecer os eventos importantes, aplicar gerência de riscos, gerar e revisar alternativas, facilitando a orientação e revisão de todo o projeto podendo, antecipar riscos e situações. Com isso, é possível evitar surpresas durante a execução do projeto. Observar o momento em que o projeto se desenvolve evidencia o real significado de se desenvolver sistemas com qualidade e documentado.

Assim, a MEDSIA tem por objetivo maior criar uma certa disciplina no desenvolvimento de sistema, de maneira que os envolvidos possam decidir de forma justificada quais as atividades e tarefas irão desenvolver, sem perder a flexibilidade de empregar as técnicas mais apropriadas.

Não é o objetivo ensinar a desenvolver cada uma das atividades. O objetivo é criar uma nova atitude de disciplina ao se desenvolver um *software* com qualidade. Esta metodologia não visa substituir outras metodologias de desenvolvimento, mas servir de complementos a estas. Tampouco, prescreve um processo padronizado ou novo.

Este trabalho enquadra-se nesta perspectiva e visa contribuir na definição de uma metodologia, a qual proporcionará oferecer um conteúdo adaptado para diferentes problemas da IA. O fator mais importante é a mudança de visão quanto à aceitação de uma nova forma de se produzir programas de uma maneira eficiente, natural e intuitiva.

6.3. Trabalhos Futuros

O objetivo deste trabalho é realmente ambicioso principalmente considerando que a primeira metodologia conhecida de desenvolvimento em Inteligência Artificial, *KADS,* foi resultado do trabalho de uma equipe, trabalhando por vários anos. Aqui, trata-se de uma dissertação de Mestrado, única, primeira, de uma série de outros trabalhos na mesma direção.

Felizmente chega-se ao final podendo mostrar que a extensão das metodologias para desenvolvimento usando Abordagem Simbólica da IA pode ser concretizada em exemplo para se propor uma nova metodologia. Os exemplos escolhidos para ilustrar a

MEDSIA foram simples, adequados para não turvar a clareza da metodologia com detalhes de um exemplo sofisticado.

Reconhece-se, no entanto ser importante mais reflexão, se possível partindo de onde chegou esta dissertação e por isso, se sugere algumas melhorias:

- Estabelecer critérios para qual das diferentes abordagens da Computação Evolucionária deveram ser utilizadas. As principais são: Algoritmos Evolutivos, Programação Evolutiva e Algoritmos Genéticos.
- Implementar um ambiente de desenvolvimento adequado ao suporte do modelo.
- Oferecer suporte inteligente às atividades do processo de desenvolvimento.

Cabe ressaltar que este trabalho não pretende esgotar o assunto e que, por ser uma metodologia nova que vem se desenvolvendo, constitui-se em um tema fértil para discussões acadêmicas. Almeja-se que esta metodologia e dissertação venham a acrescentar suportes científicos para estudos futuros, com uso acadêmico e profissional. Em conclusão espera-se ter dado uma contribuição à construção de programas em IA, de modo menos empírico que os sistemas dos primeiros dias.

GLOSSÁRIO

Serão apresentados os significados para os termos menos comuns usados durante a elaboração do projeto.

Atividades: ações requeridas para criar e entregar o projeto.

Base de Conhecimento: modelos que representam mecanismos de representação de conhecimento.

Baseline: documento associado às fases do ciclo de vida do sistema. É a especificação do *software* obtida após o detalhamento dos requisitos, considerado a linha de base do desenvolvimento.

Build: versão que atende determinados requisitos.

Elicitação do Conhecimento: capturar e utilizar o conhecimento de um ser humano em uma aplicação computacional.

Epistemologia: ramo da filosofia que estuda a origem, a estrutura, os métodos e a validade do conhecimento. Aparece ligada a IA devido aos seus estudos na área de conhecimento.

Fases: componentes do ciclo de vida ou passos que indicam o progresso do projeto.

Frames: estruturas compostas de dados e informações sobre os objetos ou argumentos (componentes), sua hierarquia e seus atributos. Permite agregar maior quantidade de informações a determinados símbolos.

Heurísticas: algoritmos que não buscam diretamente a otimização pura, mas geram soluções aceitáveis (boas soluções). São utilizadas por serem computacionalmente mais eficientes e fáceis de serem implementadas. Englobam estratégias, procedimentos, métodos de aproximação tentativa/erro, sempre na procura da solução mais aceitável.

Interpolação: é uma técnica para estimar valores de funções em pontos intermediários de intervalos, a partir de valores da função calculados nos extremos desses intervalos.

Milestones: eventos importantes no projeto (momentos de decisão).

Motor de Inferência: é uma estratégia de busca sobre determinada base de conhecimento baseada em regras de produção. A busca mais comum chama-se *modus ponens* devido à fácil compreensão e da forma de raciocínio facilmente representável pela composição das regras.

Overtraining: caso o treinamento de uma RNA for prolongado demais, gera um problema que pode levar a uma super especialização da rede, principalmente quando se dispõe de poucos dados e uma perda da capacidade da rede de responder bem a dados nunca apresentados, ou seja, perda da capacidade de generalização.

Redes Semânticas: estrutura de representação de conhecimento simbólico. Notação gráfica de arco (atributos) – nó (objetos) representam eventos ou ações que contém embutidas informações de raciocínio sobre as relações e objetos.

Regras: forma mais comum de representação de conhecimento, em função da facilidade de construção e aquisição de conhecimento. São usadas para representar relacionamentos.

Release: versão completa do produto.

Otimização: é uma técnica algorítmica usada na busca pela melhor solução (solução ótima) para o problema. Deve ser utilizada quando não existe uma solução simples e diretamente calculável para o problema.

REFERÊNCIAS BIBLIOGRÁFICAS

ABEL, M. **Estudo da especialidade em petrografia sedimentar e sua importância para a engenharia de conhecimento.** Tese de Doutorado em Ciência da Computação, Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Ciência da Computação, UFRGS, Porto Alegre, 2001.

ALEKSANDER, I., MORTON, H. **An Introduction to Neural Computing.** London, Chapman & Hall, 1990.

ALVARES, L. O., SICHMAN, J. S. **Introdução aos Sistemas Multiagentes.** In: MEDEIROS, C. M. B. Jornada de Atualização em Informática. Brasília: SBC, 1997.

ANGELE, J., FENSEL, D., LANDES, D. S., NEUBERT, S., STUDER, R. **Model-Based and Incremental Knowledge Engineering: The MIKE Approach.** In: J. Cuena (ed.), *Knowledge Oriented Software Design, IFIP Transactions A-27*, North Holland, Amsterdam, 1993.

ANGELE, J., FENSEL, D., STUDER, R. **Domain and Task Modeling in MIKE.** In: Domain knowledge for interactive system design. ed.: A.G. Sutcliffe. London 1996. S. 149-163. Universität Karlsruhe; Institut für Angewandte Informatik und Formale Beschreibungsverfahren. 1996.

AZEVEDO, F. M., BRASIL, L. M., LIMÃO, R. C. **Redes Neurais com Aplicações em Controle e em Sistemas Especialistas.** Florianópolis, Bookstore, 2000.

BARRETO, J. M. **Inteligência Artificial no limiar do século XXI.** 3. ed. – Florianópolis, ρρρ edições, 2001.

BITTENCOURT, G. **Inteligência Artificial: ferramentas e teorias.** Sociedade Brasileira de Computação: 10ª Escola de Computação, Campinas, 1996.

BOEHM, B. W. **A spiral model of software development and enhancement.** ACM Software Engineering Notes, 1986.

BOOCH. G., RUMBAUGH, J. JACOBSON, I. **UML, guia do usuário.** trad. Paulo Freitas da Silva, Rio de Janeiro, Campus, 2000.

BRASIL, L. M. **Aquisição do Conhecimento aplicada ao diagnóstico de epilepsia.** Dissertação de Mestrado em Engenharia Elétrica, Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia Elétrica, UFSC, Florianópolis, 1994.

____, L. M. **Uma Proposta de Arquitetura para Sistema Especialista Híbrido e a Correspondente Metodologia de Elicitação/Representação do Conhecimento.** Tese de Doutorado em Engenharia Elétrica, Universidade Federal de Santa, Catarina Programa de Pós-Graduação em Engenharia Elétrica, UFSC, Florianópolis, 1996.

BRUSSO, D. B., MULLER, L., VARGAS, V. **Modelo Espiral.** Universidade de Passo Fundo, Instituto de Ciências Exatas e Geociências, Ciência da Computação, Passo Fundo, 2001.

CASTELFRANCHI, C. **Social Power: A Point Missed in Multi-Agent, DAI and HCI.** In: Decentralized Artificial Intelligence. North-Holland, Elsevier Science Publishers, 1990.

CHARNIAC, E., MCDERMOTT, D. **Introduction to Artificial Intelligence.** Massachusetts, Addison-Wesley, 1985.

CHORAFAS, D. M. **Sistemas Especialistas – Aplicações Comerciais.** São Paulo, 1988.

CUNHA, F. S**. Um Sistema Especialista para a Previdência Privada.** Dissertação de Mestrado em Engenharia de Produção, Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia de Produção, UFSC, Florianópolis, 1995.

DAZZY, R. L. S. **Sistemas Especialistas Conexionistas Implementado por Redes Diretas e Bidirecionais.** Dissertação de Mestrado em Ciência da Computação, Universidade de Santa Catarina, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 1999.

DEMARCO, T. **Mad about measurement.** In: Why does software cost so much? New York, Dorset House, 1995.

DOBREV, D. D.: **Strawberry Prolog.** Capturado em 05 de novembro de 2003. On line – Disponível na Internet.<http://www.dobrev.com>

DUARTE, K. C., FALBO, R. A. **Uma Ontologia de Qualidade de Software.** In: Anais do VII Workshop de Qualidade de Software, XIV Simpósio Brasileiro de Engenharia de Software, João Pessoa, 2000.

FALBO, R.A., TRAVASSOS, G.H. **Um Estudo sobre Ambientes de Desenvolvimento de Software com Suporte Baseado em Conhecimento.** In: Anais da XXI Conferência Latino-Americana de Informática (CLEI'95), Canela, 1995.

____, R. A., MENEZES, C. S. **Integração de Conhecimento em um Ambiente de Desenvolvimento de Software.** Tese de Doutorado em Engenharia, Universidade Federal do Rio de Janeiro, Programa de Pós-Graduação em Ciência em Engenharia de Sistemas e Computação, UFRJ, Rio de Janeiro, 1998.

____, R. A., MENEZES, C. S., ROCHA, A. R. C. **Um Servidor de Conhecimento de Processo de Software.** In: Anais da X Conferência Internacional de Tecnologia de Software - X CITS, Curitiba, 1999.

FALQUETO, J., L, W.C., BORGES, P. S., BARRETO, J. M. **The measurement of artificial intelligence: na IQ form machines?** In: Proceedings of the International Conference on Modeling, Identification and Control - IASTED, Innsbruck, Austria, 2001.

FEIGENBAUM, E. A., MCCORDUCK, P. **The Fifth Generation.** Addison-Wesley, 1983.

FENSEL, D., HARMELEN van F. **A Comparison of Languages which Operationalize and Formalize KADS Models of Expertise.** The Knowledge Engineering Review, Vol. 9, pp. 105-146, 1994.

____, D., POECK, K. **A Comparison of Two Approaches to Model-based Knowledge Acquisition.** In: A future for knowledge acquisition. EKAW '94. Ed.: L. Steels. Berlin 1994. S. 46-62. (Lecture notes in computer science. 867. Lecture notes in artificial intelligence.) Universität Karlsruhe; Institut für Angewandte Informatik und Formale Beschreibungsverfahren. 1994.

FERNANDEZ, C. A. I. **Definición de una Metodología para el Desarrollo de Sistemas Multi-Agente.** Tese de Doctorado. Departamento de Ingeniera de Sistemas Temáticos. Universidad Politécnica de Madrid, 1998.

FININ, T. **Knowledge Query and Manipulation Language.** Laboratory for Advanced Information, Technology Computer Science and Electrical Engineering, University of Maryland Baltimore Country, 1993. Capturado em 30 de set de 2003. On line – Disponível na Internet. http://www.cs.umbc.edu/kqml.

FOGEL, D. B. **Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.** IEEE Press, 1995.

FRANCESCHI, A.S.M., BARRETO, J.M. **Desenvolvimento de Agentes Autônomos para Gerência de Redes de Computadores.** Tese de Doutorado (CPEEL, UFSC), Florianópolis, 1999.

FRIEDERICH, S, GARDANO, M. **Expert Systems Design and Development.** John Wiley and Sons, 1989.

FURLAN, J. D. **Modelagem de Objetos através da UML.** São Paulo: Makron Books, 1998.

GALLANT, S. **Connectionist expert systems.** Communications of the ACM 31,2, 1988.

GAMMACK, J. G.; YOUNG, R. M. **Phisicological Techniques for Eliciting Expert Knowledge.** In Research and Development in Expert System, M.A. Bramer. Cambridge University Press, 1985.

GANASCIA. J. G. **Inteligência Artificial.** Ed. Ática, 1993.

GENNARI, J.H., MUSEN, M.A., FERGERSON, R.W., GROSSO, W.E., CRUBEZY, M., ERIKSSON, H., NOY, N.F., TU, S.W. **The Evolution of Protégé: An Environment for Knowledge-Based Systems Development.** International Journal of Human-Computer Studies, 1999.

GIANNESINI, F. KANOUI, H. PASERO, R., CANEGHEM van, M. **Prolog.** Paris, InterEditions, 1985.

HARMON, P., KING, D. **Expert Systems: artificial intelligence in business.** New York, John Wiley & Sons, Inc., 1985.

HART, A. **Knowledge acquisition for expert system.** 2. ed. MacGraw-Hill, 1992.

HAYKIN, S. **Redes Neurais: princípios e prática.** trad. Paulo Martins Engel. 2. ed. Porto Alegre, Bookman, 2001.

HICKMAN, F. R., KILLIN, J. L. LAND, L., MULHALL, T., PORTER, D., TAYLOR, R. M. **Analysis for knowledge-Based System: A Practical Guide to the KADS Methodology.** London: Ellis Harwood, 1989.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems.** In: Arbor, University of Michigan Press, 1975.

JACOBSON, I. **Object-Oriented Software Engineering.** Addison-Wesley, 1992.

JENSEN, C. **QwikNet Professional Neural Network Software.** Capturado em 21 de fevereiro de 2004. On line – Disponível na Internet.<http://qwiknet.home.comcast.net>

JOHNSON, S. **Emergence.** New York, Scribner, 2001.

KELLER, R. **Tecnologia de Sistemas Especialistas.** São Paulo, Makron Books, 1991.

KELLNER, M. I. **Software process modeling experience.** In: Proceedings of ICSE, IEEECSP. Pittsburgh, 1989.

LANGTON, C.G. **Artificial life: an overview.** Massachusetts, MIT PRESS, 1995.

LARMAN, C. **Utilizando UML e Padrões: uma introdução à análise e ao projeto orientado a objetos.** trad. Luiz A. Meirelles Salgado. Porto Alegre, Bookman, 2000.

LEVINE, R., I., DRANG, D. E., BARRY, E. **Inteligência Artificial e Sistemas Especialistas.** São Paulo, MacGraw-Hill, 1988.

MEDEIROS, M. R., BARRETO, J. M. **Estudo Comparativo sobre Metodologias de Desenvolvimento de Sistemas Especialistas.** In: IT CONFERENCE SUCESU – MT 2003 - II SEMINÁRIO DE INFORMÁTICA. Cuiabá – MT: UNIC, Outubro. 2003.

____, M. R., BARRETO, J. M. **Metodologias para IA: Sonho ou Tecnologia?.** In: IT CONFERENCE SUCESU – MT 2003 - VI Escola Regional da Sociedade Brasileira de Computação. Cuiabá – MT: SUCESU/SBC/UNIRONDON, Outubro. 2003. ISBN 858844269-8.

MCGRAW, K. L., BRIGGS, K. .H. **Knowledge Acquisition: Principles and Guidelines.** Prentice Hall, 1989.

MINSKY, M. L., PAPERT, S. A. **Perceptrons: an introduction to computational geometry.** The MIT Press, Massachusetts, 1969.

MUSEN, M. A., GENNARI, J. H., ERIKSSON, H. T., PUERTA, A. R. **PROTÉGÉ-II: Computer support for development of intelligent systems from libraries of components**. Proceedings of Medinfo'95, pp. 766–770, Vancouver, BC, 1995.

____, M. A., GENNARI, J. H., WONG, W. W. **A Rational Reconstruction of INTERNIST-I using PROTÉGÉ-II.** Section on Medical Informatics, Stanford University School of Medicine, Stanford, CA 94305-5479, U.S.A, 1998.

NILSSON, N. J. **Principles of Artificial Intelligence.** Springer-Verlag, Berlin, 1982.

OLIVEIRA, F. M. **Inteligência Artificial Distribuída.** In: IV Escola Regional de Informática. Londrina, 1996.

OSHIRO, A., K., NOVELLI, D. P., LUCENA, P. **Aquisição de Conhecimento.** Instituto de Ciências Matemáticas de São Carlos, 2001.

PALAZZO, L. A. M. **Algoritmos para Computação Evolutiva.** Relatório Técnico – Pelotas: Grupo de Pesquisa em Inteligência Artificial, Universidade Católica de Pelotas, 1997.

PELLEGRINI, G. S. **Proposta de uma Metodologia de Avaliação para Sistemas Especialistas na Área Médica.** Dissertação de Mestrado em Engenharia Elétrica, Universidade Federal de Santa Catarina, Programa de Pós-Graduação em Engenharia Elétrica, UFSC, Florianópolis, 1995.

PETERS, J. F., PEDRYCZ, W. **Engenharia de Software: Teoria e Prática.** trad. Ana Patrícia Garcia. Rio de Janeiro, Campus, 2001.

POLYA, G. **A arte de resolver problemas.** Intersciência, trad. De "How to solve it: A New Aspect Mathematic Method", Princeton University Press, Rio de Janeiro, 1975.

____, G. **How to solve it – a new aspect of mathematical method.** Nova York, Doubleday Anchor, 1975.

PRESSMAN, R. S. **Engenharia de Software.** 2. ed., São Paulo, Makron Books, 1995.

____, R.S. **Engenharia de Software.** 5. ed., São Paulo, McGraw-Hill, 2002.

RABUSKE, R. A. **Inteligência Artificial.** Florianópolis, Universidade Federal de Santa Catarina, 1995.

REITMAN, W. **Integrated Design Teams: Knowledge Engineering for Large Scale Commercial Expert System Development.** System, Man, and Cybernetics, Vol 19, 1989.

REUMATOLOGIA, Sociedade Brasileira de. **Doenças Reumáticas.** Capturado em 16 de fevereiro de 2004. On line – Disponível na Internet.<http://www.reumatologia.com.br/principais_doencas.htm>

RICH, E. **Artificial Intelligence.** New York, McGraw-Hill, 1983.

____, E., KNIGHT, K. **Inteligência Artificial.** 2. ed., São Paulo: Makron Books 1993.

ROCHA, A. R., MALDONADO, J. C., WEBER K. C. **Qualidade de Software: Teoria e Prática.** São Paulo, Prentice Hall, 2001.

ROLANDI, W. G. **Knowledge Engineering in Practice.** Al Expert, 1986.

ROYCE, W. W. **Managing the development of large software systems.** In: Proceedings of IEEE WESCON, 1970.

RUSSEL, S., NORVIG, P. **Artificial Intelligence: a modern approach.** New Jersey, Prentice-Hall, 1995.

SHERING, L., SHAPIRO, E. **The Art of Prolog.** MIT Press, 1986.

SHORTLIFFE, E. H. **MYCIN: a rule-based computer program for advising physicians regarding ant microbial therapy selection.** Tese de Doutorado, Stanford University, Califórnia, 1974.

____, E. H. **Computer based medical consultation: MYCIN.** New York, American Elsevier, 1976.

____, E. H. **Knowledge-based systems in medicine.** In: Medical Informatics Europe 91, Adlasnic, Berlin, Springer Verlag, 1991.

SILVA, D. R., POSSEBON, E., ALMEIDA, M. A. F. **SEMAÇA – Sistema Especialista para auxílio no diagnóstico de doenças da maçã e macieiras.** III Simpósio de Informática Planalto Médio, UPF, Passo Fundo, 2002.

SIMONS, G. T. **Introdução a Inteligência Artificial.** São Paulo, Classe, 1988.

SOMMERVILLE, I. **Engenharia de Software.** trad. André M. de A. Ribeiro. São Paulo, Addison Wesley, 2003.

STYLIANOU, A. C., MADEY, G. R., SMITH, R. D. **Selection criteria for expert system shells: A socio-technical framework.** Communications of the ACM, 35, 32-48, 1992.

TAFNER, M., XEREZ, M., RODRIGUES, I. F. W. **Redes Neurais Artificiais: Introdução e princípios de neurocomputação.** Blumenau, EKO, 1995.

VELOSO, P. A. S., VELOSO, S. R. M. **Problem decomposition and reduction.** In: Progress in Cybernetics and System Research, F. P. R. Trappl, J. Klir, ed. vol. VIII. Hemisphere, Washington DC, 1981, p. 199-203.

WATERMAN, D. **A Guide to Expert System.** Addison-Wesley, 1986.

WIELINGA, B. J., SCHREIBER A. T., BREUKER J. A. **KADS: A modelling approach to knowledge engineering. Knowledge Acquisition**, vol. 4 (1), pp. 127-161. Special issue The KADS approach to knowledge engineering, 1993.

WINSTON, P. H. **Inteligência Artificial.** Livros Técnicos e Científicos. Rio de Janeiro, 1981.

YOURDON, E. **Administrando o ciclo de vida do sistema.** Rio de Janeiro, Campus, 1989.

ANEXOS

Anexo 1 – Formulário de Aquisição de Conhecimento

Sessão: 007/2004

I) Domínios:

Engenheiro do Conhecimento:	
Especialista do Domínio:	

II) Metas:

Revisar o conhecimento estruturado.

III) Levantamento de Questões:

6.3.1. IV) Resumo da Sessão

Ver árvore de decisão do SISEADJU.

V) Assinaturas:

Engenheiro do Conhecimento:	Especialista do Domínio:

VI) Datas

Data/Hora Início:	Data/Hora Conclusão:

Anexo 2 - Questionário de Avaliação do Protótipo

INSTRUÇÕES

Este questionário foi elaborado com a finalidade de avaliar o desempenho do sistema testado e que será anexado ao projeto realizado. Preencha todos os campos e se necessário comente sua resposta. O questionário deve ser entregue até o dia 12/03/2004.

Nome:_____

1)- Houve alguma dificuldade ao utilizar o sistema? Especifique.

☐ Sim ☐ Não

R: _____

_____

2)- Você percebeu diferença entre o processo manual anteriormente utilizado e o sistema? Caso sua resposta seja Não, por favor, explique.

☐ Sim ☐ Não

R: _____

_____

3)- Houve algum benefício na utilização deste sistema?

☐ Sim ☐ Não

Quais?

○ Rapidez ○ Praticidade ○ Organização

4)- A interface está intuitiva e fácil de usar:

☐ Sim ☐ Não

O que poderia ser acrescentado?

R: _____

_____

5)- Quais foram as principais dificuldades ao usar o sistema?

R: _____

_____

6)- Quais as sugestões que você gostaria de fazer? O que poderia ser melhorado?

R: _____

_____

Muito Obrigado por sua colaboração.