UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

COMPARATIVO DE FERRAMENTAS PARA SISTEMAS ESPECIALISTAS

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

ALINE RASSWEILLER DE SOUZA

BLUMENAU, JUNHO/2001

2001/1-02

COMPARATIVO DE FERRAMENTAS PARA SISTEMAS ESPECIALISTAS

ALINE RASSWEILLER DE SOUZA

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Roberto Heinzle — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Roberto Heinzle

_____

Prof. Oscar Dalfovo

_____

Profª. Marilda Maria de Souza

DEDICATÓRIA

Com todo o meu amor, dedico este trabalho
aos meus pais Noir e Marly pelo amor, carinho e apoio,
e aos meus amigos, que me deram incentivo.

AGRADECIMENTOS

Ao meu professor e orientador Roberto Heinzle, pela orientação na elaboração deste trabalho. A todos os professores do DSC – Departamento de Sistemas e Computação, pelas sugestões, comentários e apoio que foram importantes e certamente só enriqueceram meu trabalho.

Aos meus amigos; em especial ao Eduardo Kohler, pela amizade, incentivo e companheirismo. A minha irmã Fabiana Rassweiller de Souza, que participou comigo nesta conquista. Pelo apoio e compreensão, dando-me sempre entusiasmo para superar os obstáculos.

E principalmente a Deus pela iluminação e força recebida.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

RESUMO

O presente trabalho apresenta os aspectos relacionados às ferramentas para sistemas especialistas. Inicialmente são mostrados os principais conceitos, um histórico dos sistemas especialistas bem como as funções e características dos elementos que os compõe. Em complemento, são apresentadas as ferramentas pesquisadas e realizadas aplicações experimentais a fim de realizar uma análise dos recursos, facilidades e eficiência de algumas ferramentas para sistemas especialistas. Para verificar a funcionalidade das ferramentas procedeu-se um comparativo.

ABSTRACT

This exposed work presents some aspects which are related with expert system tools. Initialy, It will be described main concepts about expert system, including its historic, functions and features about elements of its composition. In addition, It will be presented some tools, which are used to realize some experiments to analyze their features, facilities and efficiency. To verify the functionality of the expert systems tools, It was realized a comparative between them.

1 INTRODUÇÃO

Com o objetivo de construir uma máquina inteligente com capacidade de reduzir problemas, surgiram os sistemas especialistas. Uma de suas principais características é solucionar problemas através do uso de um modelo computacional de raciocínio que simula um especialista humano, chegando as mesmas conclusões que este chegaria, caso se defrontasse com um problema comparável (Levine, 1988).

Como um sistema especialista é um programa computacional que tem uma finalidade específica, ele depende do estabelecimento e ordenação de procedimentos em todas as suas etapas de implementação. Segundo Heckmann (1998), formalizado o problema e identificado o objetivo, deve-se estudar quais as possíveis representações e ferramentas que poderão servir de base para a implementação.

Além das linguagens de programação dirigidas ao desenvolvimento de aplicações de Inteligência Artificial, merecem destaque as "*shells*", nome dado a uma família de ferramentas que objetivam apoiar e simplificar o processo de construção de sistemas especialistas. Elas representam a tendência atual no processo de desenvolvimento destes sistemas, incorporam os mais variados recursos e são capazes de suportar vários dos mecanismos de raciocínio (Heinzle, 1995). Com essas ferramentas, a representação do conhecimento e o raciocínio tornaram-se mais flexíveis, suportando regras, sistemas de manutenção da verdade e um série de outros mecanismos de raciocínio. Assim sendo, a utilização dessas ferramentas proporciona uma economia de tempo para os desenvolvedores e também um melhor aproveitamento por parte do usuário final, isso deve-se a diversos fatores tecnológicos e econômico-sociais, dentre os quais tem-se: a dificuldade de acesso a especialistas humanos em determinadas regiões, o armazenamento e formalização do conhecimento de vários especialistas humanos, ferramenta de apoio à tomada de decisões por parte do especialista, treinamento de profissionais e imparcialidade na tomada de decisões.

Esse aspecto do desempenho e potencialidade das ferramentas para sistemas especialistas será a fonte de pesquisa do referido trabalho. Fundamentado em pesquisas e ensaios, procurar-se-á estudar as facilidades, os recursos e a qualidade das *ferramentas Expert SINTA*, *Arity Prolog* e *SPIRIT*, através da implementação de aplicações experimentais.

1.1 ORIGEM DO TRABALHO

A Inteligência Artificial (IA), conceito relativo a programas de computador que simulam o comportamento humano, tem se consolidado ao longo dos últimos anos, como uma das mais poderosas ferramentas para obtenção de ganhos significativos de produtividade nas empresas. Os resultados práticos de sua aplicação, nos mais diversos nichos de negócio, vêm crescendo tanto em valor como em importância à medida que se aproximam dos setores gerenciais de tomada de decisão nas empresas.

Existem muitas idéias controvertidas e conceitos mal formulados a respeito do que é a Inteligência Artificial. Um conceito, que de algum modo é aceito universalmente, é o de que a Inteligência Artificial compreende uma ciência para desenvolvimento de técnicas e algoritmos destinados à resolução de problemas complexos.

O uso das técnicas de Inteligência Artificial no desenvolvimento de aplicações, resulta em sistemas especialistas, que são nada mais do que programas de computador que usam conhecimento e procedimentos de inferência para a resolução de problemas que são de uma complexidade e dificuldade superiores à capacidade humana em calculá-los (LIA, 1995).

Desta forma, um dos mitos de que esta classe de aplicações busca substituir a atuação humana perde totalmente o sentido. Todo o processo de busca e tomada de decisão deve ser orientado a partir do conhecimento dos "especialistas humanos", que serão os principais condutores do sistema. Os programas são, portanto, ferramentas que embutiram dentro de seu código algum conhecimento ou método para resolver uma questão, e que permitem aos "especialistas humanos" uma visão mais aprofundada ou específica que seria impossível sem esse mecanismo de cálculo avançado.

Falar em ferramentas computacionais é fazer referência a uma gama incontável de recursos, desenvolvidos desde que o computador se tornou uma máquina programável. Sendo assim, designar qual a melhor ferramenta para construção de sistemas especialistas, não é tão simples, pois antes de tudo, eles são sistemas que podem ser programados em qualquer linguagem, bastando para isso o programador ter habilidade suficiente para fazê-lo. Então, boas ferramentas são as que atendem a um espectro bastante amplo de problemas, juntando a isto uma capacidade de lidar com características importantes dos problemas (Rabuske, 1995).

1.2 PROBLEMA

A construção de sistemas especialistas envolve investimentos em profissionais de diversas áreas, implicando em grandes riscos durante a fase de produção. Uma conhecida estratégia para minimizar o problema é o investimento em ferramentas capazes de prototipar, avaliar e implementar o projeto de um sistema. Procura-se, assim, diminuir a necessidade de recursos, entre eles o tempo envolvido. Porém, é difícil conciliar alta produtividade e versatilidade em ferramentas computacionais.

Diante da existência de diversas ferramentas para sistemas especialistas, a dificuldade remanescente para o projetista está em selecionar as ferramentas adequadas para que, com eficácia, o mesmo consiga projetar seus sistemas. Isto ocorre devido à variedade de opções, pela falta de informações que os projetistas normalmente possuem acerca deste assunto e pela ausência de referências para proceder esta escolha. Apesar do constante crescimento e da variedade de recursos disponíveis para ajudar o projetista a escolher a ferramenta que melhor se adapte, existem poucas fontes disponíveis com informações objetivas sobre as características destes recursos. Deve-se ressaltar que, este fato ocorre não por falta de interesse destas pessoas, mas na verdade, pela divulgação pouco clara das características destas ferramentas.

Este trabalho visa a melhoria do processo de construção de sistemas especialistas. Para isto, foram utilizadas ferramentas consagradas na Inteligência Artificial, que serão utilizadas para fazer o comparativo, não com o objetivo de definir qual a melhor ferramenta, mas sim de investigar suas potencialidades.

1.3 JUSTIFICATIVAS

O estudo das ferramentas para sistemas especialistas irá permitir de certa forma que os projetistas observem a existência dos recursos que podem ajudar a melhorar o resultado final de seu trabalho. As ferramentas para sistemas especialistas têm influência sobre o modo de trabalho da equipe de projeto, sobre o tempo necessário para gerar uma boa solução, sobre o tempo de aprendizado para o uso da ferramenta, sobre os custos do projeto e sobre a

necessidade ou não de uma estrutura auxiliar. Portanto, uma escolha mal fundamentada poderá trazer muitos prejuízos.

Com os projetistas munidos das informações presentes neste trabalho, eles terão condições de especificar qual a sua necessidade e selecionar qual(ais) a(s) ferramenta(s) mais adequada(s) para a sua aplicação. Além disso, o trabalho apresenta uma visão mais abrangente dos recursos disponíveis. Os responsáveis pelo desenvolvimento de ferramentas, como, por exemplo, os institutos de pesquisa, têm também material necessário para coleta de informações para que sejam desenvolvidas novas ferramentas.

Outro fator importante que este trabalho também abrange, para os pesquisadores, são os elementos de avaliação de suas ferramentas. Esta avaliação verifica se os sistemas estão indo de encontro aos seus objetivos. Fornece, também, a possibilidade de identificação de pontos fracos em suas ferramentas e de que maneira as ferramentas podem auxiliar, de forma mais consistente, o processo de construção de sistemas especialistas.

1.4 OBJETIVO DO TRABALHO

Analisando as ferramentas para sistemas especialistas: *Expert SINTA*, *Arity* Prolog e *SPIRIT*, pretende-se desenvolver um comparativo dos recursos, facilidades, eficiência e formas de representar o conhecimento dessas ferramentas.

Os objetivos específicos do trabalho são:

a) expor os principais aspectos relacionados à criação e utilização de sistemas especialistas;
b) verificar as ferramentas de apoio para a construção desses sistemas;
c) aquilatar os aspectos relevantes para estas ferramentas;
d) avaliar a qualidade através de aspectos da engenharia de software e outros aspectos relevantes para estas ferramentas.

1.5 ESTRUTURA DO TRABALHO

O trabalho encontra-se estruturado da seguinte maneira:

No primeiro capítulo, é apresentada uma visão geral do trabalho, sua importância, objetivos, justificativas e sua organização.

No segundo capítulo são abordados: o conceito, histórico, estrutura e características dos sistemas especialistas.

No terceiro capítulo enfatizam-se as ferramentas para sistemas especialistas, especificamente as ferramentas *Expert SINTA*, *Arity* Prolog e *SPIRIT* que serão utilizadas posteriormente para que seja feito o comparativo.

O capítulo quatro aborda os aspectos da Engenharia de Software que serão utilizados para avaliar a qualidade das ferramentas.

No capítulo cinco, são apresentadas as considerações para a realização do comparativo.

O capítulo seis apresenta os resultados dos testes aplicados com a finalidade de verificar o alcance dos objetivos. Sendo este o foco principal deste trabalho, onde será realizado o comparativo com os resultados obtidos durante o trabalho.

No sétimo capítulo encontram-se relacionadas às considerações finais e sugestões para estudos futuros.

2 SISTEMAS ESPECIALISTAS

Um sistema que utiliza técnicas de Inteligência Artificial (IA) criado para resolver problemas em um determinado domínio e cujo conhecimento utilizado é fornecido por pessoas que são especialistas naquele domínio é denominado sistema especialista (Batezini, 2001).

Um sistema convencional é baseado em um algoritmo, emite um resultado final, normalmente correto e processa um volume de dados de maneira repetitiva, enquanto que um sistema especialista é baseado em uma busca heurística e trabalha com problemas para os quais não existe uma solução convencional organizada de forma algorítmica disponível.

Um sistema especialista é inicialmente projetado e desenvolvido para atender a uma aplicação determinada e limitada do conhecimento humano. É capaz de emitir uma decisão, tal qual um especialista humano emitiria, apoiando-se em conhecimento justificado, a partir de uma base de conhecimentos.

Alguns autores apresentam definições formais de sistemas especialistas, Crippa (1999) afirma: “Um sistema especialista é um programa inteligente de computador que se utiliza de métodos inferenciais para a resolução de problemas técnicos e altamente especializados. Por utilizar-se da Inteligência Artificial, um ramo da computação que estuda a capacidade de uma máquina raciocinar e aprender como um ser humano, os sistemas especialistas interagem com seu usuário numa linguagem natural de perguntas e respostas, sugerindo e auxiliando na solução de problemas complexos”.

Para Kandel (1992), “Os sistemas especialistas podem ser caracterizados como sistemas que reproduzem o conhecimento de um especialista adquirido ao longo dos anos de trabalho”.

Rich (1993) escreve: “Os sistemas especialistas solucionam problemas que normalmente são solucionados por especialistas humanos. Para solucionar tais problemas, os sistemas especialistas precisam acessar uma substancial base de conhecimentos do domínio da aplicação, que precisa ser criada do modo mais eficiente possível”.

Desta forma, os sistemas especialistas são classificados como programas computacionais inteligentes, que podem numa área específica, ajudar o trabalho de um especialista. Estes sistemas utilizam conhecimentos de áreas e estratégias de solução de problemas transmitidos

por um especialista. Assim é possível que os conhecimentos especializados e as descobertas dos melhores especialistas de uma área sejam concentrados e disponibilizados para uso e benefício geral.

2.1 HISTÓRICO

Os sistemas especialistas têm sua origem no fim da segunda grande guerra. Nessa época, grupos independentes de cientistas ingleses e norte-americanos trabalhavam numa máquina eletrônica que se pudesse conduzir por um programa armazenado de instruções e fosse feita para executar cálculos numéricos complexos, o que hoje se chamaria computador. Simultaneamente, psicólogos interessados na resolução de problemas pelo homem, buscavam desenvolver programas de computador que simulassem o comportamento humano. Esses indivíduos, interessados tanto no processamento simbólico quanto na resolução de problemas pelo homem, formaram uma subdivisão da informática que se chama Inteligência Artificial (IA) (Crippa, 1999).

Na década de setenta, houve uma revolução na era computacional, culminando com a criação dos sistemas especialistas. O objetivo dos pesquisadores de Inteligência Artificial era desenvolver programas de computador que pudessem em alguns sentidos “pensar”, isto é: resolver problemas de uma maneira que seriam considerados inteligentes, se fossem feitos pelo homem. Os sistemas especialistas são frutos de anos de pesquisa (Bronzino, 1995).

No início dos estudos sobre sistemas especialistas, o primeiro sistema a ser considerado sistema especialista foi o DENDRAL, em 1965, porém sua aplicabilidade ficou restrita ao meio acadêmico. Em 1976, desenvolveu-se o sistema especialista mais conhecido, o MYCIN. Sua importância deve-se ao fato de ser o pioneiro no uso de fatores de certeza. Apesar de sua popularidade, o MYCIN também ficou restrito aos meios acadêmicos. Somente em 1982, com o desenvolvimento do XCON, os sistemas especialistas saíram das universidades e ganharam o interesse das indústrias. Atualmente, o setor manufatureiro e empresarial faz uso destes sistemas. Nos anos 90 um novo avanço para os sistemas especialistas, ao invés de utilizar a lógica clássica, muitos sistemas passaram a utilizar a lógica difusa, em busca de soluções mais precisas, rápidas e confiáveis.

Segundo Liebowitz (1994), os sistemas especialistas têm se difundido por todo o mundo, obtendo sucesso, como nos seguintes países:

Em Singapura, desde 1980, os sistemas especialistas são utilizados em setores bancários, área financeira, na manufatura, dentre outros. Pode-se citar alguns dos importantes sistemas especialistas desenvolvidos: (i) *Audit Expert System*, no setor de contabilidade, (ii) *Credit Evoluation*, no setor bancário, (iii) *Intelligent Fuzzy Logic Tutor*, no setor de educação.

No Japão, tem-se desenvolvido sistemas especialistas para diagnósticos, planejamento, escalonamento e para indústrias pesadas. Aplicações de sistemas especialistas associados à lógica difusa estão se multiplicando, principalmente na área de eletrodomésticos. Na Alemanha, os sistemas especialistas são utilizados principalmente para indústrias pesadas e o uso de sistemas especialistas associados à lógica difusa está crescendo rapidamente. Nos Estados Unidos existem várias tendências no uso dos sistemas especialistas, como por exemplo: (i) um movimento contínuo em direção à integração e aos sistemas híbridos; (ii) ênfase para o problema de “solução de negócios”; (iii) crescimento da tendência de sistemas de informação “ativos”, ampla base de conhecimento, compartilhamento deste conhecimento e sistemas inteligentes híbridos; (iv) necessidade de fornecer suporte de alto nível para pesquisa em Inteligência Artificial; (v) uso de metodologias estruturadas para desenvolvimento de sistemas especialistas.

O uso de sistemas especialistas no Brasil encontra-se em crescimento contínuo em diversas áreas. Dentre os sistemas desenvolvidos pode-se citar: (i) “Análise de crédito bancário”; (ii) “Análise de hepatopatias crônicas”; (iii) “Análise química qualitativa de minerais”; há também uma tendência a desenvolver-se sistemas especialistas híbridos.

2.2 CARACTERÍSTICAS

Os sistemas especialistas são também conhecidos como sistemas baseados em conhecimento. O processo de construção de um sistema especialista é geralmente chamado de engenharia de conhecimento, a qual tipicamente envolve uma forma especial de interação entre o construtor do sistema, chamado de engenheiro do conhecimento, e um ou mais especialistas. O engenheiro do conhecimento “extrai” do especialista humano, seus

procedimentos, estratégias e regras para a solução do problema, e constrói seu conhecimento dentro do sistema especialista (Waterman, 1986).

Enquanto os sistemas convencionais, nos seus mais diversos tipos, baseiam-se em dados, fórmulas e informações, em um sistema especialista, além destes, acrescenta-se um conjunto de regras e heurísticas, que concentram conhecimento especializado. Quanto aos dados, fórmulas e informações, em princípio não deixam margem à dúvida: são **fatos**. As **regras** indicam procedimentos a partir de condições ou premissas que, combinadas ou não, determinam a busca da solução ou a própria decisão final. Como heurísticas entendem-se aqueles critérios ou regras particulares que o especialista humano adotou, sendo resultado da sua vivência ou experiência singular na atividade. Como têm a visão específica de um ou mais especialistas, podem ser contestadas por outros profissionais que, também especialistas, têm outra vivência prática.

Um sistema especialista é inicialmente projetado e desenvolvido para atender a uma aplicação determinada e limitada do conhecimento humano. É capaz de emitir uma decisão, tal qual um especialista humano emitiria, apoiando-se em conhecimento justificado, a partir de uma base de conhecimentos.

Para tomar uma decisão sobre um determinado assunto, um especialista o faz a partir de fatos que encontra e de hipóteses que formula, no período de sua formação e no decorrer de sua vida profissional, sobre esses fatos e hipóteses. E o faz de acordo com a sua experiência, isto é, com o seu conhecimento acumulado sobre o assunto e, com os fatos e hipóteses, emite a decisão (Batezini, 2001).

Durante o processo de raciocínio este especialista, vai verificando qual a importância dos fatos que encontra comparando-os com as informações já contidas no seu conhecimento acumulado sobre esses fatos e hipóteses. Neste processo, vai formulando novas hipóteses e verificando novos fatos; e esses novos fatos vão influenciar no processo de raciocínio. Este raciocínio é sempre baseado no conhecimento prévio acumulado. Um especialista com esse processo de raciocínio pode não chegar a uma decisão se os fatos de que dispõe para aplicar o seu conhecimento prévio não forem suficientes.

Um sistema especialista deve, além de inferir conclusões, ter capacidade de aprender novos conhecimentos e, deste modo, melhorar o seu desempenho de raciocínio, e a qualidade de suas decisões.

Um sistema especialista é uma fonte prática capaz de lidar com problemas complexos, resolver questões que requerem um alto nível de juízo e perícia humana, e comunicar-se com seu usuário através de um diálogo eficaz. Os sistemas especialistas fazem perguntas, fornecem conclusões e as justificam.

Desta forma os sistemas especialistas são caracterizados por:

a) utilizar lógica simbólica ao invés de cálculos numéricos;

b) incorporar uma base de conhecimento;

c) fazer inferências e deduções a partir de uma informação disponível;

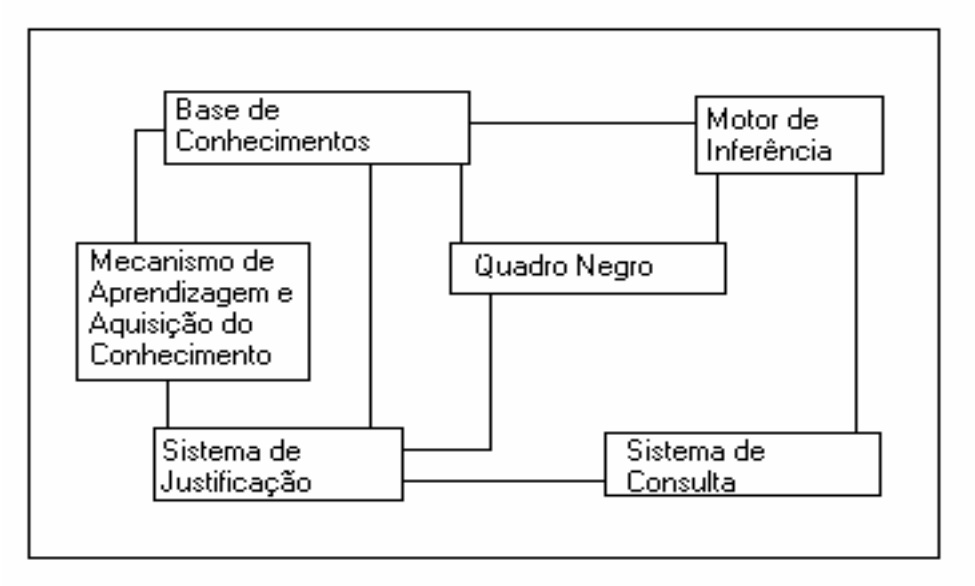
d) ter capacidade para explicar suas conclusões.

2.3 ESTRUTURA DE UM SISTEMA ESPECIALISTA

Conforme Heinzle (1995), a composição de um sistema especialista depende de fatores como a generalidade pretendida, os objetivos do mesmo, a representação interna do conhecimento e as ferramentas usadas na implementação. Entretanto, existem várias arquiteturas de sistemas especialistas sendo usadas. Dentre elas a mais simples de compreender e a mais difundida compõem-se de alguns elementos básicos: base de conhecimentos, máquina ou motor de inferência, quadro negro, sistema de justificação, mecanismo de aprendizagem e aquisição de conhecimento e sistema de consulta.

Esse modelo geral de arquitetura apresentada por grande número de autores é mostrado na Figura 1:

Figura 1 – Estrutura de um sistema especialista



Fonte: (Heinzle, 1995).

2.3.1 BASE DE CONHECIMENTO

Base de conhecimento é um elemento permanente, mas específico de um sistema especialista. É onde estão armazenadas as informações de um sistema especialista, ou seja, os fatos e as regras. As informações armazenadas de um determinado domínio fazem do sistema um especialista neste domínio, portanto o sucesso de um sistema especialista depende, em grande parte, da forma de como o conhecimento é representado e dos mecanismos para a exploração deste conhecimento (Favero, 1999).

2.3.2 MÁQUINA OU MOTOR DE INFERÊNCIA

A máquina de inferência é o mecanismo que procura as respostas na base de conhecimento. Busca as regras necessárias a serem avaliadas e ordena-as de uma maneira lógica. A partir daí, direciona o processo de inferência. Funciona como um supervisor, que dirige a operação sobre o conhecimento contido no sistema especialista. Uma máquina de inferência toma decisões e julgamentos baseados em dados simbólicos contidos na base de conhecimento. As funções básicas da máquina de inferência são inferência e controle. Depois de iniciado o sistema, a máquina de inferência busca na base de conhecimento, fatos e regras, e compara estes fatos com a informação fornecida pelo usuário. A operação da máquina de

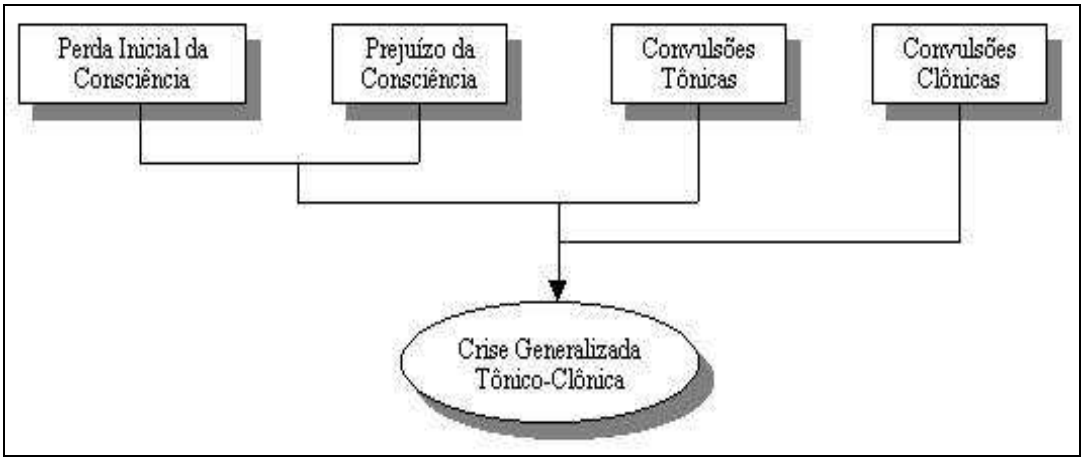
inferência é baseada em algoritmos que definem a busca específica e a unificação de regras. Basicamente, a máquina de inferência compara a entrada fornecida pelo usuário com as regras contidas na base de conhecimento buscando “combinações”.

2.3.2.1 MÉTODO DE RACIOCÍNIO

Existem basicamente dois modos de raciocínio aplicáveis a regras de produção propostos por Weiss (1988): encadeamento progressivo ou encadeamento para frente (do inglês, “*forward chaining*”), e encadeamento regressivo ou encadeamento para trás (do inglês, “*backward chaining*”).

No encadeamento progressivo (Figura 2), também chamado encadeamento dirigido por dados, a parte esquerda da regra é comparada com a descrição da situação atual, contida na memória de trabalho. As regras que satisfazem a esta descrição têm sua parte direita executada, o que, em geral, significa a introdução de novos fatos na memória de trabalho.

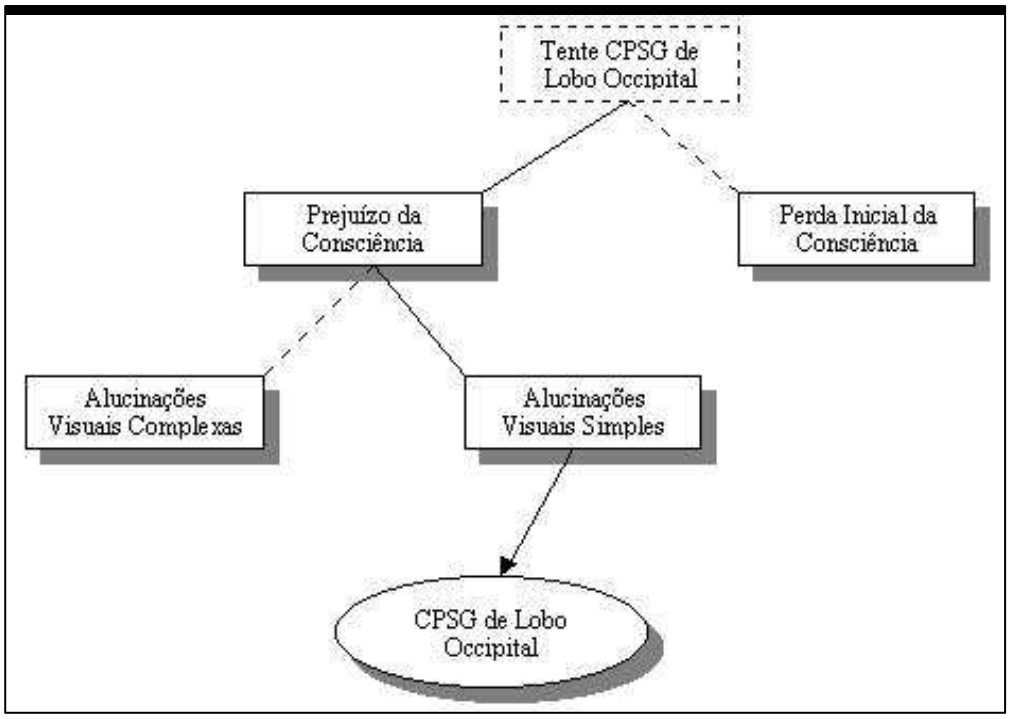
Figura 2- Encadeamento Progressivo.



No encadeamento regressivo (Figura 3), também chamado encadeamento dirigido por objetivos, o comportamento do sistema é controlado por uma lista de objetivos. Um objetivo pode ser satisfeito diretamente por um elemento da memória de trabalho, ou podem existir regras que permitam inferir algum dos objetivos correntes, isto é, que contenham uma descrição deste objetivo em suas partes direitas. As regras que satisfazem esta condição têm as instâncias correspondentes às suas partes esquerdas adicionadas à lista de objetivos

correntes. Caso uma dessas regras tenha todas as suas condições satisfeitas diretamente pela memória de trabalho, o objetivo em sua parte direita é também adicionado à memória de trabalho. Um objetivo que não possa ser satisfeito diretamente pela memória de trabalho, nem inferido através de uma regra, é abandonado. Quando o objetivo inicial é satisfeito, ou não há mais objetivos, o processamento termina.

Figura 3 - Encadeamento Regressivo



O tipo de encadeamento normalmente é definido de acordo com o tipo de problema a ser resolvido. Problemas de planejamento, projeto e classificação tipicamente utilizam encadeamento progressivo, enquanto problemas de diagnóstico, onde existem apenas algumas saídas possíveis mas um grande número de estados iniciais, utilizam encadeamento regressivo.

2.3.2.2 TRATAMENTO DA INCERTEZA

Sabe-se que o conhecimento humano não é determinístico[1]. Não há especialista que sempre se encontre em condições de afirmar determinada conclusão com certeza absoluta.

[1] recebem apenas uma instanciação, valores sim ou não, verdadeiro ou falso, ligado ou desligado, etc.

Graus de confiança são freqüentemente atribuídos às suas respostas, principalmente quando existe mais de uma. Este, sem dúvida, é um dos mais fortes pontos críticos na elaboração de uma representação computacional do saber humano. Observa-se a dificuldade em representar a confiabilidade das informações:

a) especialistas humanos não se sentem confortáveis em pensar em termos de probabilidade. Suas estimativas não precisam corresponder àquelas definidas matematicamente;

b) tratamentos rigorosamente matemáticos de probabilidade utilizam informações nem sempre disponíveis ou simplificações que não são claramente justificáveis em aplicações práticas.

O tratamento de incerteza é uma ativa área de pesquisa em sistemas especialistas, pois os domínios adequados à implementação de sistemas especialistas se caracterizam exatamente por não serem modelados por nenhuma teoria geral, o que implica descrições incompletas, inexatas ou incertas.

Na verdade, existem duas correntes de pensamento: aquela que utiliza fórmulas estatísticas rigorosas, com teoria das probabilidades, e aquela que utiliza uma abordagem da teoria das possibilidades sobre os fatores de certeza, ou seja, mais generalizada e sem uma base matemática forte (LIA, 1995).

2.3.3 QUADRO-NEGRO

O quadro-negro, ou rascunho, como também é chamado, é uma área de trabalho que o sistema utiliza durante o processo de inferência. Nesta área são armazenadas informações de apoio e suporte ao funcionamento do sistema quando o mesmo está raciocinando. Este lugar na memória é destinado para fazer avaliações das regras que são recuperadas da base de conhecimento para se chegar a uma solução. As informações são gravadas e apagadas em um processo de inferência até se chegar à solução desejada. Conforme Rabuske (1995), embora todos os sistemas especialistas usem o quadro negro, nem todos o explicitam como componente do sistema.

2.3.4 SISTEMA DE JUSTIFICAÇÃO

A justificação é um requisito importante dos sistemas especialistas. Tem a função de esclarecer o usuário a respeito de uma conclusão apresentada pelo sistema ou ainda explicar porque uma pergunta está sendo feita. Em muitos domínios nos quais os sistemas operam, as pessoas não aceitam resultados se esses não estiverem devidamente justificados (Heinzle, 1995).

2.3.5 MECANISMO DE APRENDIZAGEM E AQUISIÇÃO DE CONHECIMENTO

Conforme Levine (1988) a fase de aquisição é a que apresenta maior dificuldade a construção de sistemas especialistas. Esta dificuldade provém do fato de inexistir uma linguagem comum de entendimento entre as partes evolvidas do projeto.

O conhecimento deve ser adquirido junto ao especialista da área em que for desenvolvido o sistema, podendo ser através de entrevistas, questionários ou documentação já existente em arquivos. Este conhecimento deverá ser formalizado para ser armazenado na base de conhecimento do sistema (Luchtenberg, 2000).

Um dos aspectos que mais diferenciam os sistemas especialistas dos sistemas tradicionais é a sua capacidade de aprender. À medida que vai sendo utilizado, o sistema especialista deve ampliar, reformular ou atualizar o seu conhecimento. Em geral, esta possibilidade é facultada pelos recursos de um editor de textos, embutido ou não no sistema, com a capacidade de aceitar a atualização da base de conhecimentos (Salvato, 1997).

O mecanismo de aprendizagem é um módulo que, rudimentar na maioria dos sistemas especialistas, consta apenas de recursos que permitem fazer acréscimos e alterações na base de conhecimentos. Existe, porém, a possibilidade de tornar esse recurso mais potente, fazendo com que adquira uma capacidade maior, depurando a base de conhecimentos, reordenando prioridades, estabelecendo mecanismos de controle para expansão da árvore de busca, executando outras ações que melhorem o desempenho do sistema e a qualidade da resposta.

2.3.6 SISTEMA DE CONSULTA

Os usuários de sistema especialistas interagem de forma intensa com o sistema, pois além de receberem dele as conclusões alcançadas também participam ativamente do processo de inferência e da construção da base de conhecimentos. Esses sistemas devem, portanto, oferecer bons recursos de comunicação que permitam, até o usuário sem conhecimentos computacionais, tirar proveito dos mesmos (Heinzle, 1995).

Aspectos internos do sistema, como a terminologia computacional, por exemplo, devem ser evitados, e aspectos técnicos relativos a implementação devem ser transparentes ao usuário. A maioria dos sistemas existente usa técnicas simples de iteração, quase sempre utilizado perguntas já pré-formatadas e repostas tipo múltipla escolha. Outra técnica é a definição de uma gramática sintética simples com um vocabulário restrito e limitado, própria para utilização o sistema.

2.4 AQUISIÇÃO DO CONHECIMENTO

Pode-se dizer que aquisição do 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. A etapa de aquisição do conhecimento é reconhecidamente o “gargalo” do processo de desenvolvimento de um sistema especialista. Esta dificuldade está relacionada com a dificuldade de transmissão do conhecimento por parte do especialista; ou porque o conhecimento não é bem definido, ou porque é difícil expressar este conhecimento em palavras. Da mesma forma, fatores não previstos podem afetar o processo de aquisição do conhecimento especialista-engenheiro (Teive, 1997).

Supõe-se que o conhecimento por si só já é o suficiente para a resolução de problemas. Essa é a idéia por trás do conhecimento declarativo: não há preocupações quanto ao seu uso, somente quanto à sua posse e especificação, e ela já garantirá o alcance dos objetivos desejados através de proposições falsas ou verdadeiras.

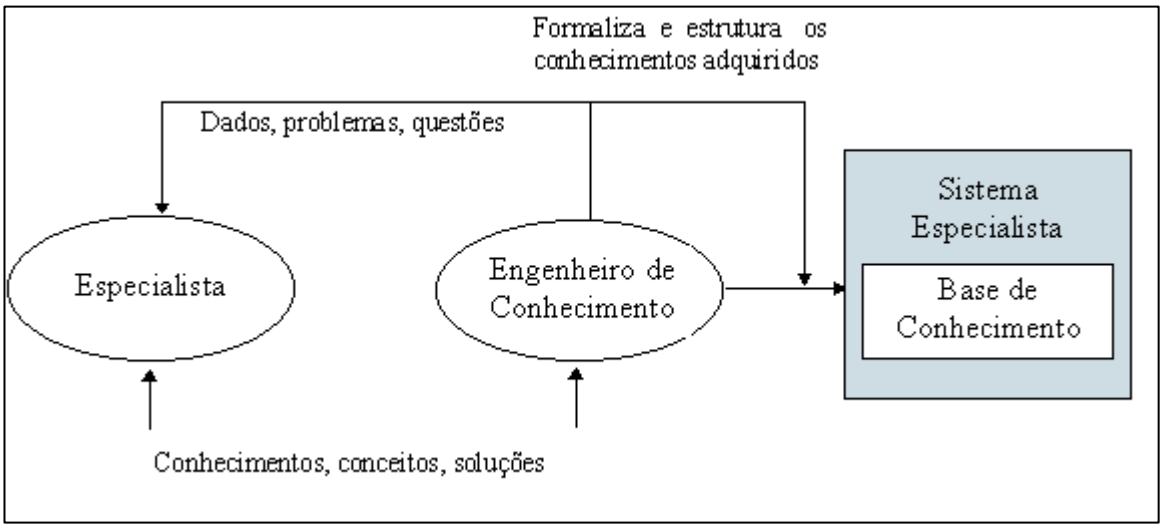
Porém, a realidade é bem mais problemática. Um computador não é capaz de decidir qual a próxima informação que ele utilizará para o desenvolvimento de uma atividade. Falta às máquinas um modo menos metódico e linear de ação. É necessário que se especifique uma estratégia de uso do seu “saber”. Esse tipo de conhecimento que depende de uma regra de extração chamamos de conhecimento procedural (LIA, 1995).

Como foi já foi colocado neste capítulo, a eficácia e o bom desempenho de um sistema especialista é dependente do conhecimento contido em sua base. Desta forma, o processo de aquisição do conhecimento tem um papel de suma importância dentro das etapas de desenvolvimento de um sistema especialista propriamente dito, pois caso esta etapa seja feita sem os cuidados necessários, todo o projeto pode tornar-se comprometido.

A fonte de conhecimento a que se referencia, dependendo do problema de interesse a ser resolvido pelo sistema especialista, podem ser livros, manuais e o especialista da área de interesse, o qual se constitui a fonte de conhecimento mais importante para o desenvolvimento do sistema especialista.

A Figura 4 mostra o processo de aquisição do conhecimento:

Figura 4 – Processo de aquisição do conhecimento



Fonte: (Teive, 1997).

2.5 REPRESENTAÇÃO DO CONHECIMENTO

Representação do conhecimento são métodos utilizados para modelar os conhecimentos de especialistas em algum campo, de forma eficiente e colocá-los pronto para serem acessados pelo usuário de um sistema inteligente e pela máquina de inferência. Ou seja, representação do conhecimento é uma combinação de estruturas de dados e procedimentos interpretativos, que, se usados corretamente em um programa, terão uma conduta inteligente (Luchtenberg, 2000).

Problemas relacionados com a representação de conhecimento não surgiram com o advento dos computadores digitais. Há séculos têm sido estudados pelos filósofos, cujos trabalhos jamais podem ser ignorados pelos cientistas da computação. Para os pesquisadores em Inteligência Artificial, estes problemas continuam sendo bastante significativos.

As interpretações de “representação de conhecimento” variam bastante, mas a questão central, ainda segundo Chaiben (1999) apud Brachman (1990), é a seguinte: “Como transmitir o conhecimento do mundo para um robô ou outro sistema computacional, dando-lhe uma capacidade adequada de raciocínio, de modo que este conhecimento possa ser utilizado para permitir ao sistema uma adaptação e exploração do seu ambiente?”. Com relação a isto, surgem algumas indagações básicas importantes:

a) de que forma o conhecimento pode ser expresso?

b) como encontrar a linguagem adequada para a representação deste conhecimento?

c) como formar uma base de conhecimento suficientemente detalhada e que represente a compreensão do domínio?

d) como realizar inferências automáticas, dando acesso tanto ao conhecimento implícito na base de conhecimento quanto àquele armazenado explicitamente (declarativo)?

e) como o sistema deve proceder na presença de informações incompletas, incorretas ou de senso comum?

A representação do conhecimento está sempre relacionada com as formas de expressão da informação. Diferentes sistemas de representação podem ser mais adequados para diferentes problemas, embora ainda existam muitas pesquisas no sentido de desenvolver sistemas e linguagens de representação de propósito geral. Geralmente, a questão central é: como representar o conhecimento de modo formal sem considerar como ele será utilizado?

Embora a existência de variadas formas de representação do conhecimento facilite a modelagem de sistemas diversamente caracterizados, existe também o inconveniente de exigir vasto e diversificado ferramental. Por isso hoje, grande parte do esforço em Inteligência Artificial tem se concentrado em buscar ou aperfeiçoar formalismo para a representação do conhecimento.

De forma geral, pode-se afirmar que se representa o conhecimento para posteriormente recuperá-lo, para raciocinar com ele e para adquirir mais conhecimento. A representação do conhecimento é a formalização do conhecimento do sistema. Para que isso seja possível existem técnicas que permitem modelar o conhecimento de forma eficiente e deixá-lo pronto para ser acessado facilmente (Heinzle, 1995).

É importante considerar que uma linguagem de representação de conhecimento não deve ser caracterizada somente em termos de sua adequação, mas também em termos de sua eficácia computacional. Assim, uma representação não deve meramente prescrever como trechos individuais de informações são representados, mas deve especificar como a totalidade

da informação é estruturada e organizada, de modo que as informações relevantes possam ser recuperadas e que as inferências adequadas apresentem um nível aceitável de eficiência.

A maioria das pesquisas atual sustenta que é inútil considerar uma representação, sem considerar o raciocínio que será realizado sobre a mesma. Assim, na área de representação do conhecimento tem-se tentado aos poucos uma padronização. Sendo assim, serão explanadas algumas das formas mais conhecidas de representação:

2.5.1 REGRAS DE PRODUÇÃO

O termo “regras de produção” é usado para descrever uma família de sistemas, que têm em comum o fato de serem constituídos de um conjunto de regras, que reúnem condições e ações. A condição é constituída por um padrão que determina a aplicabilidade da regra, enquanto a ação indica o que será realizado quando a regra for aplicada.

Um sistema de produção poderá ser formado por uma ou mais bases de regras, separadas segundo as conveniências de processamento. Complementa ainda, o sistema de produção, uma estratégia de controle e estabelecendo as prioridades em que as regras serão aplicadas, bem como critérios de desempate quando houver mais regras candidatas á aplicação a um tempo só (Rabuske, 1995).

A arquitetura mais comum de sistemas especialistas é a que envolve as regras de produção (*production rules*), a causa disto reside no fato de ser natural ao humano usar o par “condição-ação” para raciocinar e decidir. Essas regras são simplesmente um conjunto de condições no estilo SE... ENTÃO..., com a possibilidade de inclusão de conectivos lógicos (E, OU, NÃO, e outros desejados) relacionando os atributos no escopo do conhecimento e o uso de probabilidades, como se vê a seguir no Quadro 1:

Quadro 1 – Regras de Produção

```
SE      carne = vermelha
E       cor preferida = tinto
OU      temperatura = ambiente
ENTÃO
        melhor vinho = exemplo CNF 70;
```

Ao se examinar uma das conclusões da regra, pode-se ter a presença do grau de confiança (CNF) na decisão. Neste exemplo citado por LIA (1995), o melhor vinho será o tinto, com uma probabilidade de 70%.

A regra em si possui a parte “condição” da regra também denominada de “lado esquerdo” ou “antecedente”, ou “premissa”, enquanto que a parte que expressa a “ação” a ser executada é denominada “lado direito” ou “conseqüente”.

Além da naturalidade para a interpretação humana a utilização de regras de produção apresenta outros aspectos positivos, como a modularidade e uniformidade. As regras podem ser manipuladas como peças independentes e novas regras podem ser incluídas a qualquer tempo o que é uma característica importante, pois o conhecimento de qualquer sistema especialista tende a aumentar com o passar do tempo. A uniformidade fica caracterizada como padrão único utilizado para todas as regras do sistema (Heinzle, 1995).

2.5.2 REDES SEMÂNTICAS

As redes semânticas são representações gráficas do conhecimento sendo desenvolvidas principalmente no campo de compreensão da linguagem e da memória. Estas redes são consideradas uma das primeiras formas de representação do conhecimento em Inteligência Artificial.

As redes semânticas são representadas através de um gráfico constituído de nós (objetos) e arcos (relações entre objetos) e são basicamente utilizadas para reconhecer pares (*maching*), fazer abstrações, deduções e comparação com outras formas de raciocínio.

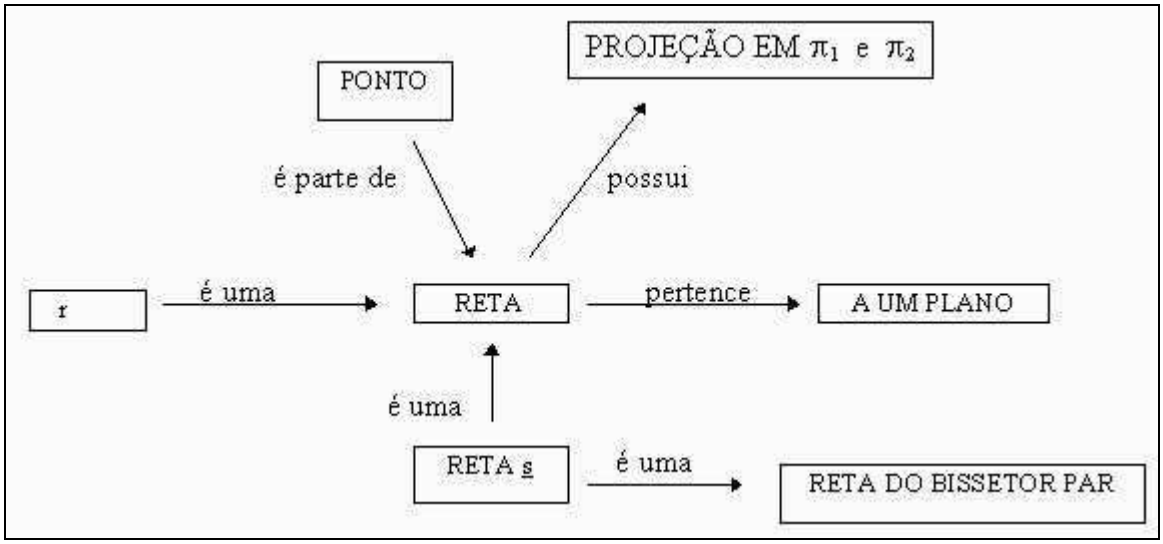
Segundo Rich (1993), “Numa rede semântica, a informação é representada como um conjunto de nós ligados, um ao outro, por um conjunto de arcos rotulados que representam as relações entre os nós”. Nestas redes, os nós geralmente representam objetos, podendo ainda representar conceitos, eventos, ações ou situações de um determinado domínio. Elas possuem grande flexibilidade, permitindo que novos fatos lhe sejam acrescentados sempre que necessário. Quando afirma-se que: reta é uma classe de objetos, r é um objeto pertencente a esta classe e “é uma” é um arco (Figura 5).

Figura 5 - Rede Semântica

Fonte: (Ulbricht, 1996).

Com relação ao fato representado na Figura 5, outros fatos podem lhe ser acrescentados, como: “toda reta tem pontos”, “toda reta tem projeção em p1 e p2”, “a reta s é uma reta do bissetor”, “toda reta pertence a um plano” (Figura 6). Na exemplificação feita, a relação “é uma” deve ser interpretada como “é um caso isolado de”, desta forma r e s são casos de “reta” e todas as propriedades de reta a elas se aplicam.

Figura 6- Representação do conhecimento com rede semântica



Fonte: (Ulbricht, 1996).

Ao anexar acontecimentos a uma base de dados, deve-se tomar cuidado com as frases que descrevem estes acontecimentos. Os lingüistas distinguem níveis de linguagem, um dos quais se denomina estrutura de superfície, que representa a parte visível, tal qual as pessoas a pronunciam, e outra é a estrutura profunda ou espaço semântico, que trata do significado das frases. A nível mnemônico, o mais importante é o espaço semântico. Para identificar a estrutura de base de um acontecimento sem se enganar com a estrutura de superfície da frase que a descreve, começa-se identificando a ação, que deve ser o nó central (Ulbricht, 1996).

A representação semântica fornece também, uma maneira de inferir as propriedades dos conceitos específicos a partir das propriedades dos conceitos genéricos. Pode-se então afirmar, que os nós dessas redes herdam as características de outros nós que lhe são relacionados. A esta propriedade, denomina-se hereditariedade.

Os nós genéricos das redes contêm informações como valores de casos típicos, também chamados de “valores de *default*”.

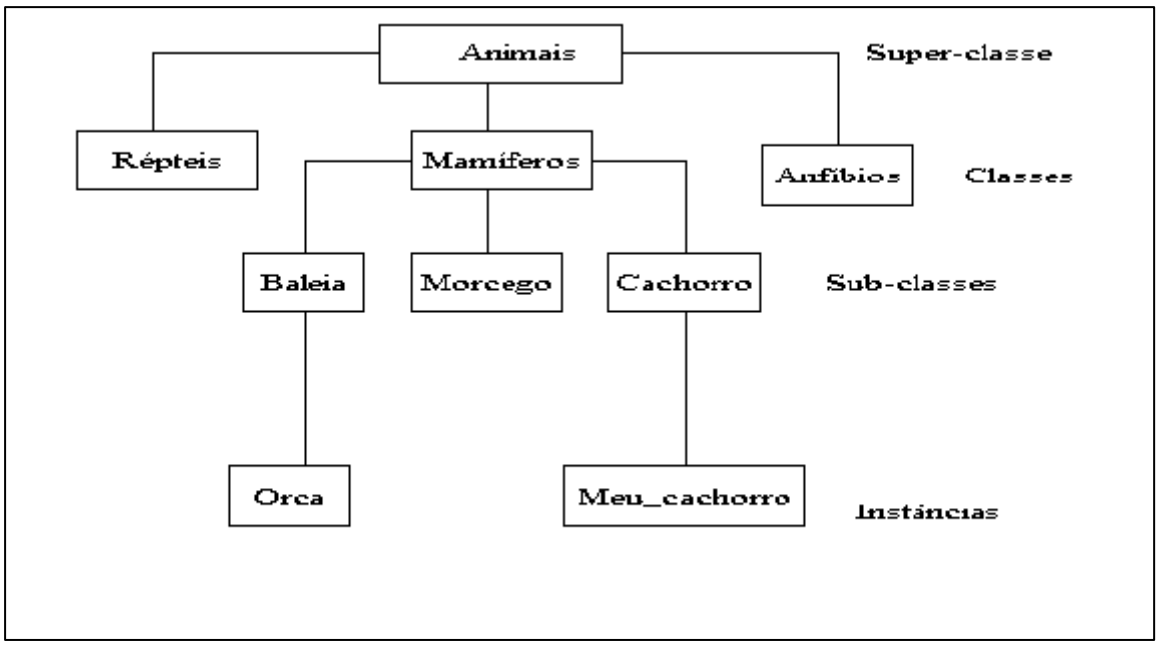
2.5.3 QUADROS E ROTEIROS

Acredita-se que os seres humanos usam conhecimentos adquiridos em experiências passadas na interpretação de situações novas. Por já conhecerem alguns objetos e seqüências que podem acontecer numa tal ocasião, o ser humano faz associações entre elas e cria expectativas na sua mente do que pode vir a acontecer realmente. Este tipo de conhecimento tem sido representado através dos quadros (do inglês, "*frames*") e sua variação, os roteiros (do inglês, ”*scripts*").

A representação do conhecimento usando quadros constitui em uma rede de nós e relações, organizada de forma hierárquica, onde os nós do topo representam conceitos gerais, enquanto que os nós inferiores representam instâncias mais específicas dos conceitos. Quadros são particularmente úteis quando usados para representar o conhecimento relacionado a eventos ou conceitos padronizados.

A Figura 7 a seguir apresenta um exemplo de representação do conhecimento usando quadros.

Figura 7 – Representação do conhecimento com quadros



Fonte: (Teive, 1997).

Conforme se observa, os quadros são organizados hierarquicamente, onde os quadros dos níveis hierárquicos inferiores herdam as propriedades dos níveis mais altos.

Na Figura 7, por exemplo, poderíamos ter, para a superclasse Animais, um conjunto de *slots* representando algumas características do reino animal, as quais valeria tanto para a classe dos répteis, quanto para as classes dos mamíferos e dos anfíbios. Estes *slots* poderiam ser:

- tipo_de_respiração

- tipo_de_alimentação

- habitat

- tipo_de_reprodução

Da mesma forma, as classes abaixo da superclasse Animais, podem ter outros *slots* particulares, que são específicos de cada classe. Como exemplo, a classe mamífero, poderia ter os seguintes *slots*:

- tempo_de_amamentação

- número_de_mamas

Por sua vez, a classe dos mamíferos pode conter várias sub-classes que compartilham características comuns, e além disto podem ter características próprias, ou seja, *slots* específicos. Por exemplo, a sub-classe Cachorro poderia ter os seguintes *slots*:

- cor_do_pêlo

- raça.

Finalmente, cada sub-classe pode possuir instâncias, que estão a nível de indivíduos, sendo a máxima particularização possível que pode ser feita. Poderíamos ter a instância Meu_cachorro, por exemplo, com os seguintes *slots*:

- nome

- idade.

Os valores armazenados nos *slots* também são herdados e considerados valores padrão ('*default*'), podendo ser substituídos a medida que os valores específicos são obtidos (Teive, 1997).

2.5.4 LÓGICA DAS PROPOSIÇÕES E DOS PREDICADOS

As técnicas de representação lógica freqüentemente utilizadas em sistemas inteligentes são: a lógica proposicional e a lógica de predicados. Ambas as técnicas utilizam símbolos para representar conhecimentos e operadores aplicados aos símbolos para produzir raciocínio lógico, como os mostrados no Quadro 2. Também representam métodos formais bem fundamentados para representação do conhecimento e raciocínio.

Quadro 2 – Símbolos e operadores

Operador	Símbolo
AND	Ù , &, Ç
OR	Ú , È , +
NOT	Ø ,~
IMPLIES	É ,®
EQUIVALENCE	º

A lógica proposicional representa e raciocina com proposições (declarações que são verdadeiras ou falsas). Trabalha com operadores lógicos como *AND, OR, NOT, IMPLIES, EQUIVALENCE,* que nos permite raciocinar com várias estruturas de regras.

Segundo Heinzle (1995), pode-se facilmente representar fatos do mundo real usando a lógica proposicional, entretanto, isto não é suficiente para fazer dela uma forma eficiente de representação do conhecimento em sistemas especialistas, pois são poucos os problemas que se resumem ao falso e verdadeiro suportados por ela. Desta limitação surge a opção da lógica dos predicados.

A lógica dos predicados, ou cálculo de predicados como também é conhecida, é uma extensão da lógica proposional e fornece uma representação mais apurada do conhecimento, facilitando o processamento ao permitir a utilização de variáveis[2] e funções (UFSC, 2000).

- Constantes:

  Exemplo: a constante carro pode representar o objeto Carro.

- Predicados:

  Exemplo: come (coelho, cenoura).

Predicado: come (relação entre argumentos coelho e cenoura).

[2] são os elementos do mundo real representados na base de conhecimento.

- Variáveis:

Exemplo: come (X, Y).

Onde a variável X representa o argumento coelho; a variável Y o argumento cenoura.

- Funções:

Exemplo: pai (Leo) = Pedro

mae (Maria)= Lia

amigos (Pedro, Lia)=amigos (pai (Leo), mae (Maria)).

Operações em Cálculo de Predicados

O cálculo de predicados utiliza as mesmas operações da lógica proposicional, apresentados no Quadro 2:

Exemplo: Joao ama Maria = ama (Joao, Maria)

Leo ama Maria = ama (Leo, Maria)

ama(X,Y) AND ama(Z,Y)

IMPLIES NOT ama(X,Z)

ou

ama(X,Y) Ù ama(Z,Y) ® Ø ama(X,Z)

O cálculo de predicados trabalha ainda com dois outros tipos de variáveis:

Variáveis quantificadoras:

quantificador universal ("): a expressão é verdadeira para todos os valores das variáveis:

Ex: " X ama (X, Maria)

quantificador existencial ($): existe ao menos um valor para o qual a expressão é verdadeira.

Ex: $ X ama (X, Maria)

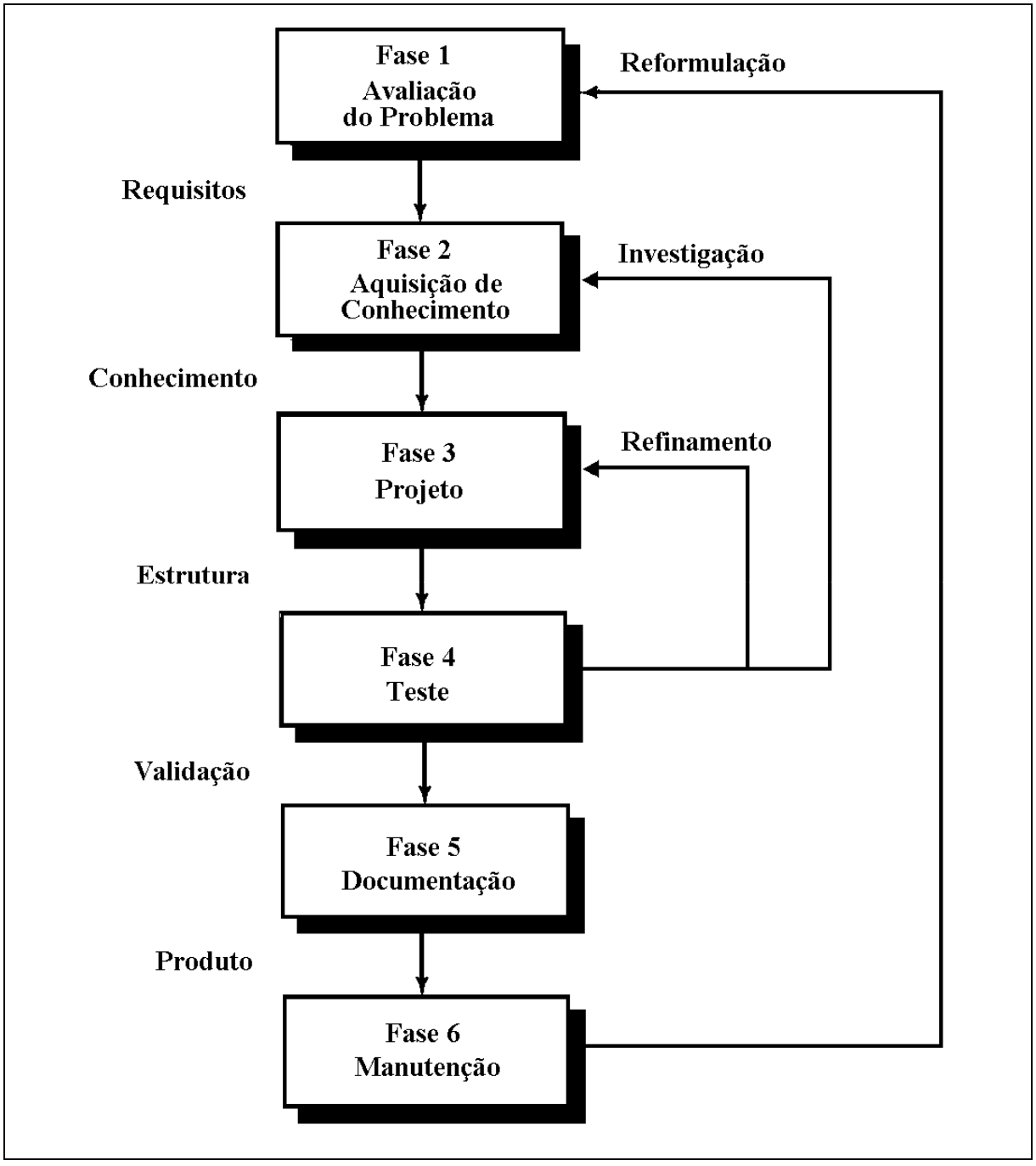
2.6 FASES DE DESENVOLVIMENTO DE UM SISTEMA ESPECIALISTA

O desenvolvimento de um sistema especialista foge da forma de desenvolvimento adotada em sistemas computacionais tradicionais, onde as fases são executadas de uma forma seqüencial e distinta. Nos sistemas especialistas, o desenvolvimento, minimiza bastante a separação em etapas, no entanto, embora alguns autores definam etapas para o desenvolvimento de sistemas especialistas, ainda existem controvérsias na definição dessas etapas e as mesmas são objetos de pesquisa e desenvolvimento. Weiss (1998) argumenta que o sucesso dos sistemas especialistas depende, fundamentalmente, de se começar o seu desenvolvimento de maneira simples e crescer, por incrementos, até que o sistema especialista torne-se consistente e significativo.

O campo de sistemas especialistas ainda não está amadurecido como a programação tradicional. A engenharia de conhecimento é mais uma arte que uma ciência, e as recomendações deste capítulo apresentam pontos que devem ser considerados no projeto de desenvolvimento de sistemas especialistas.

As etapas de projeto não ocorrem em uma seqüência clara como a que será apresentada. Na prática, a execução do projeto apresenta uma sobreposição das tarefa em um processo iterativo, com uma característica de aprendizado evolutivo sobre o problema a ser solucionado. Portanto, aqui será identificada apenas uma revisão das idéias chaves relacionadas ao projeto de desenvolvimento de sistemas especialistas, como mostra a Figura 8:

Figura 8 - Fases de desenvolvimento de um sistema especialista



Fonte: (UFSC, 2000).

2.6.1 AVALIAÇÃO DO PROBLEMA

Esta etapa estabelece uma metodologia para a avaliação da aplicabilidade de um sistema especialista para um dado problema em uma organização. Visa gerar esclarecimentos

sobre a tecnologia a ser aplicada, antes do início do projeto. A metodologia proposta consiste nas seguintes tarefas:

a) determinação da motivação da organização;

b) identificação de problemas possíveis;

c) estudo de possibilidades;

d) análise de custos e benefícios;

e) seleção do melhor projeto.

2.6.2 AQUISIÇÃO DO CONHECIMENTO

Esta etapa, como já abordado anteriormente, representa o maior desafio no desenvolvimento de um sistema especialista. A aquisição de conhecimento está ligada a um processo cíclico, cujas tarefas são:

a) coleta: é a tarefa de aquisição de conhecimento de um especialista. Requer treinamento, técnicas de entrevista, boas ferramentas de intercomunicação pessoal e habilidade para obter a cooperação do especialista;

b) interpretação: as informações coletadas são identificadas como peças chave do conhecimento, tais como conceitos, regras, estratégias, etc.;

c) análise: Envolve o estudo das peças chave do conhecimento descobertas durante a tarefa de interpretação. Relaciona a formação de teorias na organização sobre o conhecimento e estratégias de resolução de problemas;

d) projeto: com as tarefas anteriores, um novo entendimento do problema é formado. Nesta etapa, técnicas de refinamento do conhecimento são escolhidas para a próxima iteração do ciclo de atividades.

2.6.3 PROJETO

Após a aquisição do conhecimento com os especialistas, já se possui suficiente conhecimento do problema para iniciar o projeto do sistema especialista. O processo de projeto é estruturado nas seguintes tarefas:

a) seleção da técnica de representação do conhecimento: deve ser escolhida a técnica de representação do conhecimento que melhor se adeqüe a maneira como o especialista mentaliza o problema (regras de produção, *frames*, etc);

b) seleção da técnica de controle: durante os contatos com o especialista, é importante verificar como o conhecimento do sistema pode ser melhor controlado. A decisão deverá estar centrada na observação da técnica de inferência (encadeamento progressivo, encadeamento regressivo), objetivos e de outros paradigmas de resolução de problemas;

c) seleção do software de desenvolvimento: após o entendimento de como representar e controlar o conhecimento do problema, a próxima tarefa é selecionar o software para desenvolvimento do sistema especialista. É importante combinar as características do problema com a capacidade dos softwares disponíveis para desenvolvimento;

d) desenvolvimento de protótipo: nesta etapa o sistema especialista começa a ser implementado. Em muitos casos, os esforços iniciam na construção de um pequeno protótipo do sistema. Um protótipo é um modelo do sistema final cuja estrutura representa e processa o conhecimento do problema, possuindo algumas limitações;

e) desenvolvimento da interface: uma das grandes armadilhas em que os engenheiros do conhecimento caem é o de não projetar uma interface eficiente no projeto inicial. A interface deve ser definida no início do projeto e contando com a colaboração do usuário;

f) desenvolvimento do produto: durante o desenvolvimento do protótipo, conhecimentos são extraídos e testes são realizados, proporcionando novas informações como guia para o projeto do sistema especialista. A cada refinamento, as capacidades do sistema são aumentadas, e em uma forma evolucionária, o protótipo do

sistema começa a tomar a forma do sistema final. Não existe ponto fixo onde esta transição ocorre, e o protótipo gradualmente evolui no sistema completo.

2.6.4 TESTE

O sistema especialista necessitará periodicamente ser testado e avaliado para assegurar que estará convergindo aos objetivos. Deve-se definir o que será testado, como e quando o teste será conduzido, quem serão os envolvidos nos testes.

2.6.5 DOCUMENTAÇÃO

Com a maturação do projeto do sistema especialista, a quantidade de conhecimento coletado dos especialistas aumenta. Para sustentar essa situação, uma efetiva documentação deve ser elaborada desde o início do projeto.

A 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 e a cópia de material para elaboração de relatórios.

2.6.6 MANUTENÇÃO

A fase final de projeto é o sistema de manutenção. Esta fase segue a disseminação do sistema no campo onde é utilizado rotineiramente pelos usuários. As deficiências descobertas na utilização do sistema devem ser tratadas através de modificações apropriadas.

3 FERRAMENTAS PARA SISTEMAS ESPECIALISTAS

Este capítulo trata das ferramentas aplicáveis a construção de sistemas especialistas. Tais ferramentas são as diversas formas utilizadas para auxiliar os projetistas[3] na construção destes sistemas e na busca de informações, introduzindo estas informações no processo de desenvolvimento e auxiliando na tomada de decisões relativas ao projeto.

Especialmente com o advento dos sistemas especialistas, estas ferramentas têm uma variedade muito grande sendo necessário uma classificação. Desta forma, as ferramentas foram reunidas em dois grupos básicos: as linguagens de programação com características dirigidas ao desenvolvimento de aplicações na área de Inteligência Artificial, aqui representadas pelo *Arity* Prolog e as "*shells*" ferramentas para construir sistemas especialistas, baseada em algum tipo de formalismo de representação de conhecimento que visam simplificar a construção e gerência da base de conhecimentos, representadas pelo *Expert SINTA* e *SPIRIT*.

Existem várias ferramentas que não estão listadas ou apresentadas neste trabalho. Muitas destas ferramentas, freqüentemente, são criadas pelos próprios projetistas dos sistemas especialistas, para abordar informações a questões específicas de seu interesse.

Portanto, a procura por englobar totalmente o assunto é uma busca infinita e é apenas limitada pela capacidade criativa dos projetistas de buscar a sua forma pessoal. Este trabalho apresenta apenas algumas das ferramentas clássicas usadas em nossos dias na construção de sistemas especialistas.

3.1 PROLOG

Prolog palavra que expressa uma abreviação de "*PROgramming in LOGic*", é o nome de uma linguagem de programação que vem despontando como ferramenta de Inteligência

[3] são os encarregados de transportar o conhecimento humano para uma série de passos que um computador é capaz de entender.

Artificial. Foi desenvolvida no início da década de 70 em Edimburgo na Escócia e hoje é a principal linguagem de programação das que permitem representação lógica.

3.1.1 CARACTERÍSTICAS

Prolog é uma linguagem declarativa orientada para o processamento simbólico. Um programa escrito nessa linguagem consiste em um conjunto de regras que descrevem relação entre objetos. Estas relações, chamadas de predicados do programa, são escritas a partir de um subconjunto do cálculo de predicados, denominado “cláusulas de *Horn*” - cláusula que tem no máximo um literal positivo (Heinzle, 1995).

Em resumo, as principais características da linguagem Prolog são:

a) orientada para processamento simbólico;

b) representa uma implementação da lógica como linguagem de programação;

c) apresenta uma semântica declarativa inerente à lógica;

d) permite a obtenção de respostas alternativas;

e) suporta estrutura de dados que permite simular registros ou listas;

f) permite recuperação dedutiva de informação;

g) representa programas e dados através do mesmo formalismo (cláusulas);

h) incorpora facilidades computacionais extras e metalógicas.

3.1.2 FACILIDADES E RECURSOS

Segundo Heinzle (1995), esta linguagem de programação é dirigida a aplicações na área de sistemas especialistas. Algumas das características, tais como formalismo, poder de interpretação, inferência e recursos de manipulação de listas facilitam o desenvolvimento de aplicações desta natureza. A portabilidade da linguagem também é fator relevante. Ele está disponível para microcomputadores pessoais da família PC que utilizam o sistema operacional DOS.

O sistema Prolog reconhece o tipo de um objeto no programa por meio de sua forma sintática. Isso é possível porque a sintaxe do Prolog especifica formas diferentes para cada tipo de objeto. Na sintaxe adotada, comum à maioria das implementações, variáveis sempre irão iniciar com letras maiúsculas, enquanto que as constantes não-numéricas, ou átomos, iniciam com letras minúsculas. Nenhuma informação adicional, tal como tipo de dados precisa ser fornecida para que o sistema reconheça a informação com a qual está lidando.

Considera-se algumas declarações de fatos em Prolog que exprimem relações de parentesco. O fato de Carlos ser pai de José pode ser expresso como:

```
pai (carlos, jose).
```

O número de parâmetros determina a aridade do predicado, portanto o predicado pai tem aridade dois. Seja o *functor* pai como nome dessa relação, os nomes Carlos e Jose são argumentos dessa relação. A árvore genealógica de uma família pode ser definida pelo seguinte programa:

```
pai (carlos, jose).

mae(maria, jose).

pai (davi, ana).

mae(paula, ana).

pai (davi, pedro).

mae(paula, pedro).

pai (jose, marcos).

mae(ana, marcos).

pai (vitor, carol).

mae(bruna, carol).

pai (pedro, joao).
```

mae(carol, joao).

pai (pedro, carla).

mae(carol, carla).

Cada uma dessas cláusulas declara um fato sobre a relação pai e mãe. O Prolog pode apresentar algumas perguntas ou consultas sobre essa relação. Nesses questionamentos utiliza-se a seguinte sintaxe:

?pai (carlos, jose).

O Prolog pesquisará a base de dados para verificar se existe algo similar nas especificações existentes. A função da questão é basicamente testar a existência de uma relação sobre os argumentos propostos. No exemplo apresentado, a resposta (*yes*) seria positiva e ?pai(carlos, X). traria como resposta X = jose.

Outro recurso da linguagem Prolog é a utilização de regras, que especificam objetos que são verdadeiros se alguma condição for satisfeita. Portanto a regra tem:

- uma parte condicional (lado direito da regra);
- uma parte conclusiva (lado esquerdo da regra).

Se a condição pai (X, Y) é verdadeira então a conseqüência lógica é filhos (Y, X). Analogamente se a condição mae (X, Y) é verdadeira então a conseqüência lógica é filhos (Y, X).

filhos (Y,X):- pai(X,Y).

filhos (Y,X):- mae(X,Y).

Outros recursos da linguagem Prolog que acrescentam um certo diferencial são a recursividade, listas e operações sobre listas.

3.1.2.1 REPRESENTAÇÃO DO CONHECIMENTO

O Prolog dispõe de um módulo de representação de conhecimento baseado na lógica dos predicados. Essa lógica se interessa pelas sentenças declarativas, as proposições podem ser verdadeiras ou falsas. Como uma linguagem para representação de conhecimento no computador, ela deve ser definida em dois aspectos, a sintaxe e a semântica. A sintaxe de uma linguagem descreve as possíveis configurações que podem constituir sentenças. E semântica determina os fatos do mundo aos quais as sentenças se referem, ou seja, ou sistema “acredita” na sentença correspondente.

Como vantagem do uso da lógica de predicados pode-se citar, além da facilidade da representação de fatos e regras a consistência da base de conhecimento. Porém, segundo Rich (1993), deve-se considerar que existem diferenças superficiais e sintáticas entre as representações em lógica e em Prolog, incluindo:

a) em lógica, as variáveis são quantificadas explicitamente. Em Prolog a quantificação é fornecida implicitamente pelo modo como as variáveis são interpretadas. Esta distinção entre variáveis e constantes é feita em Prolog ao exigir-se que todas as variáveis comecem com letras maiúsculas ou números;

b) em lógica, há símbolos explícitos para e (Λ) e ou (V). Em Prolog, há um símbolo explícito para e (,), mas nenhum para ou. A disjunção precisa ser representada como uma lista de declarações alternativas, sendo que qualquer uma delas pode servir de base para uma conclusão;

c) na lógica, as implicações da forma “p implica q” são escritas como p $\rightarrow$q. Em Prolog, a mesma implicação é escrita “de trás para frente”, como q:-p. esta forma é natural em Prolog porque o interpretador sempre trabalha de trás para frente a partir de um objetivo, e esta forma faz com que toda regra comece com componente que precisa ser casado primeiro.

Uma diferença chave entre a representação lógica e a representação em Prolog é que o interpretador Prolog tem uma estratégia de controle fixa e, portanto, as declarações do programa em Prolog definem um determinado caminho de busca para responder a qualquer pergunta. Em contraste, as declarações lógicas definem um conjunto de respostas que elas

justificam; elas mesmas não dizem nada sobre como as respostas devem ser escolhidas, caso haja mais de uma.

3.1.2.2 INFERÊNCIA

Os objetivos do Prolog podem ter sucesso ou podem falhar. Para se ter sucesso, tem – se que unificar a cabeça da regra e todas as metas do corpo desta (no caso de construções como if/then/else, alguns objetivos podem falhar, mas a estrutura global ter sucesso). Se estas condições não forem satisfeitas, a regra falha.

Quando um objetivo falha, o Prolog se apoia em uma tentativa para satisfazer o mesmo objetivo com soluções alternativas. Enquanto faz isto, o Prolog desfaz o trabalho previamente feito para este objetivo. Prolog então continua procurando uma solução, pulando as etapas em que o objetivo falhou. Ele então repete o procedimento. Se ainda assim não vier a atingir o objetivo, se apoia mais adiante em um nível superior. Se todas as tentativas para satisfizer seu objetivo falharem, a regra falha. Este procedimento destinado a descobrir um caminho através de um espaço de problema, no Prolog é conhecido como encadeamento para trás.

3.1.2.3 APRENDIZAGEM

A aprendizagem em Prolog é feita pelo próprio projetista, que munido do conhecimento o informa junto a implementação do sistema, através das regras e fatos.

3.1.2.4 JUSTIFICAÇÃO

O Prolog não possui um sistema de justificação propriamente dito, se o projetista quiser deixar isto disponível para o usuário do sistema terá que implementar rotinas específicas que acompanhem o processo de inferência. Em uma forma mais rudimentar o Prolog dispõe de uma função em seu menu, conhecida como *Trace On* que permite o acompanhamento e visualização, porém mais a nível do projetista do sistema.

3.2 EXPERT SINTA

Desenvolvida no Ceará pelo grupo SINTA (Sistemas INTeligentes Aplicados), o *Expert SINTA* é um conjunto de ferramentas computacionais fundamentadas em técnicas de Inteligência Artificial para geração automática de sistemas especialistas.

O *Expert SINTA* é um "*shell*" implementado no ambiente de programação orientada a objetos *Borland Delphi*, dando um suporte visual de fácil operação. (LIA, 1995).

3.2.1 CARACTERÍSTICAS

O *Expert SINTA* é um software gratuito e procura ser uma ferramenta de criação geral. Entre outras características inerentes ao *Expert SINTA*, tem-se:

a) utilização do encadeamento para trás (*backward chaining*);

b) utilização de fatores de confiança (FC)[4];

c) ferramentas de depuração;

d) possibilidade de incluir ajudas on-line para cada base;

e) uso de uma máquina de inferência compartilhada;

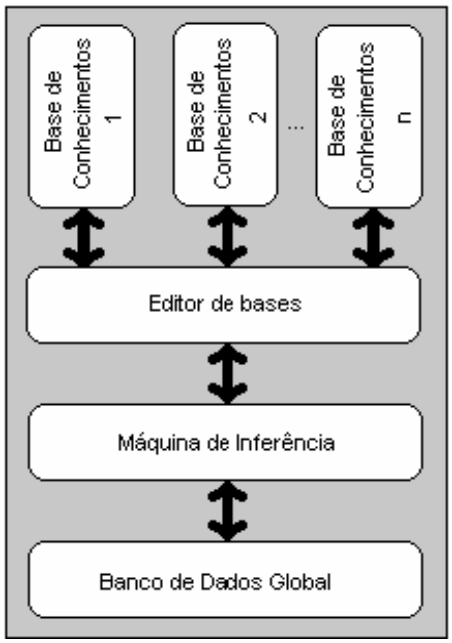
f) construção automática de telas e menus;

g) tratamento probabilístico das regras de produção.

A seguir a Figura 9 mostra a arquitetura que os sistemas especialistas gerados no Expert *SINTA* seguem:

[4] pesos numéricos atribuídos a um fato ou uma relação para indicar a confiança que se tem neste fato ou nesta relação, baseado na teoria dos conjuntos.

Figura 9 - Arquitetura simplificada do *Expert SINTA*



Fonte: (LIA, 1995).

3.2.2 FACILIDADES E RECURSOS

O *Expert SINTA* permite o desenvolvimento modular de bases de conhecimento através de uma interface de fácil manipulação e de utilitários criados para depuração. Essa ferramenta proporciona uma economia de tempo para os desenvolvedores da base de conhecimento e também um melhor aproveitamento por parte do usuário final ao permitir a inclusão de hipertextos explicativos sobre as possíveis soluções encontradas pelo sistema. Além destas facilidades, o *Expert SINTA* traz um ambiente de trabalho que possibilita, tanto ao projetista do conhecimento quanto ao usuário final, o uso das facilidades mencionadas sem que seja necessário um conhecimento aprofundado de informática. Isto é conseguido através de um modelo visual, sem o uso de pseudo-linguagens para projeto e adaptação do conhecimento do especialista, comum em outras ferramentas (Vicentin, 2000).

O *Expert SINTA* ainda permite alguns recursos diferenciais como a utilização de senhas, inclusão de *help* e acompanhamento das variáveis no processo de depuração.

3.2.2.1 REPRESENTAÇÃO DO CONHECIMENTO

Essa ferramenta permite que o criador do sistema preocupe-se somente com a representação do conhecimento do especialista, deixando para a *shell* a tarefa de interpretar o conhecimento representado e executá-lo em uma máquina.

O *Expert SINTA* utiliza um modelo de representação do conhecimento baseado em regras de produção e probabilidades. As regras de produção são bastante utilizadas por possuir a vantagem da modularidade que permite a construção passo-a-passo da base de conhecimentos, ou seja, é possível realizar vários testes com apenas um subconjunto de regras concluído. Sabe-se que menos regras implicam geralmente em um menor número de casos abrangidos.

3.2.2.2 INFERÊNCIA

O *Expert SINTA* utiliza o encadeamento para trás, o modo mais comum de utilização de um sistema especialista. O projetista deve incluir na definição da base quais os atributos que devem ser encontrados (ou seja, os objetivos - *goals* - do sistema especialista). A máquina de inferência encarrega-se de encontrar uma atribuição para o atributo desejado nas conclusões das regras (após o Então...). Para que a regra seja aprovada, suas premissas devem ser satisfeitas, obrigando à máquina a encontrar os atributos das premissas para que possam ser julgadas, acionando um encadeamento recursivo. Caso o atributo procurado não seja encontrado em nenhuma conclusão de regra, uma pergunta direta é feita ao usuário.

O *Expert SINTA* trabalha naturalmente com encadeamento para trás, mas é possível manipular a máquina de inferência de modo a simular encadeamento para frente. Para tal, as variáveis que se deseja encontrar inicialmente devem ser definidas como variáveis objetivo. (LIA, 1995).

Outro fator que deve ser considerado no aspecto da inferência é o tratamento da incerteza. O *Expert SINTA* utilizada os fatores de confiança, por trás deste método existe uma teoria matemática que calcula as probabilidades, denominada como teoria das possibilidades, não envolvendo nenhum tratamento estatístico mais aprofundado.

3.2.2.3 APRENDIZAGEM

O *Expert SINTA* foi criado de modo a permitir ao próprio analista de conhecimento implementar a base desejada. Uma base de conhecimento no *Expert SINTA* envolve os seguintes conjuntos de atributos que devem ser indicados pelo projetista da base:

- variáveis;
- regras;
- perguntas;
- objetivos;
- informações adicionais.

O *Expert SINTA* permite que se defina as variáveis que irão controlar como será realizada a inferência, dando ao sistema maior objetividade e segurança nas repostas.

3.2.2.4 JUSTIFICAÇÃO

O *Expert SINTA* mantém uma interface uniforme para a consulta de qualquer base de conhecimento criada em seu ambiente. Existem dois modos pelos quais podem ser efetuadas consultas:

a) a execução usual, na qual o usuário acompanha uma seqüência de menus de múltipla (ou única) escolha, nos quais deve-se indicar informações que resultarão nas conclusões atingidas pelo sistema especialista;

b) modo de acompanhamento, pelo qual é possível examinar o conteúdo das regras que formam o sistema, bem como acompanhar a execução passo a passo e as instâncias (valores) que cada variável possui em determinado momento. Para utilizar este modo, é preciso que o projetista da base não tenha protegido o acesso às regras por meio de uma senha.

Um sistema especialista procura atingir conclusões para determinados objetivos. Sempre que um desses objetivos é atingido, ou quando se esgotam todas as possibilidades, o *Expert SINTA* apresentará uma janela com os resultados e o acompanhamento de como se

chegou àquela conclusão (também é necessário que o projetista não tenha restringido o acesso).

Alguns sistemas também devem trazer um sistema de ajuda *on-line*, pelo qual serão dadas maiores informações de como se deve melhor utilizar as respostas conseguidas, bem como explicações mais detalhadas sobre as funções daquele determinado sistema especialista (LIA, 1995).

3.3 SPIRIT

A *shell SPIRIT* (gerador de sistemas especialistas probabilísticos), divulgada pela primeira vez no Brasil em 1993 por Kopittke (1993), foi desenvolvida pela equipe do Prof. Dr. W. Rödder na FernUniversität de Hagen - Alemanha. O sistema está programado em linguagem C++. Além das inovações realizadas nas versões, vale o destaque para a adaptação do *SPIRIT* em relação à linguagem portuguesa, o que facilita e aumenta de forma significativa seu potencial de uso e aproveitamento no Brasil. Esta adaptação foi realizada a partir de um programa de intercâmbio de pesquisa mantido entre a *FernUniversität de Hagen* e a UFSC - Universidade Federal de Santa Catarina, de Florianópolis.

O *SPIRIT* é uma interface de desenvolvimento de Sistemas Especialistas Probabilísticos, e sua sigla incorpora suas principais características, da seguinte forma (Rodder, 199l):

- *Symmetrical*: a inferência no sistema pode ser realizada nos dois sentidos: da premissa para a conclusão, e da conclusão para a premissa;

- *Probabilistic*: os prognósticos tomam como base a distribuição marginal de probabilidades;

- *Intentional*: a probabilidade dos fatos e regras que formam a estrutura de dependências entre as variáveis, podem ser aprendidas e modificadas através de observações no mundo real;

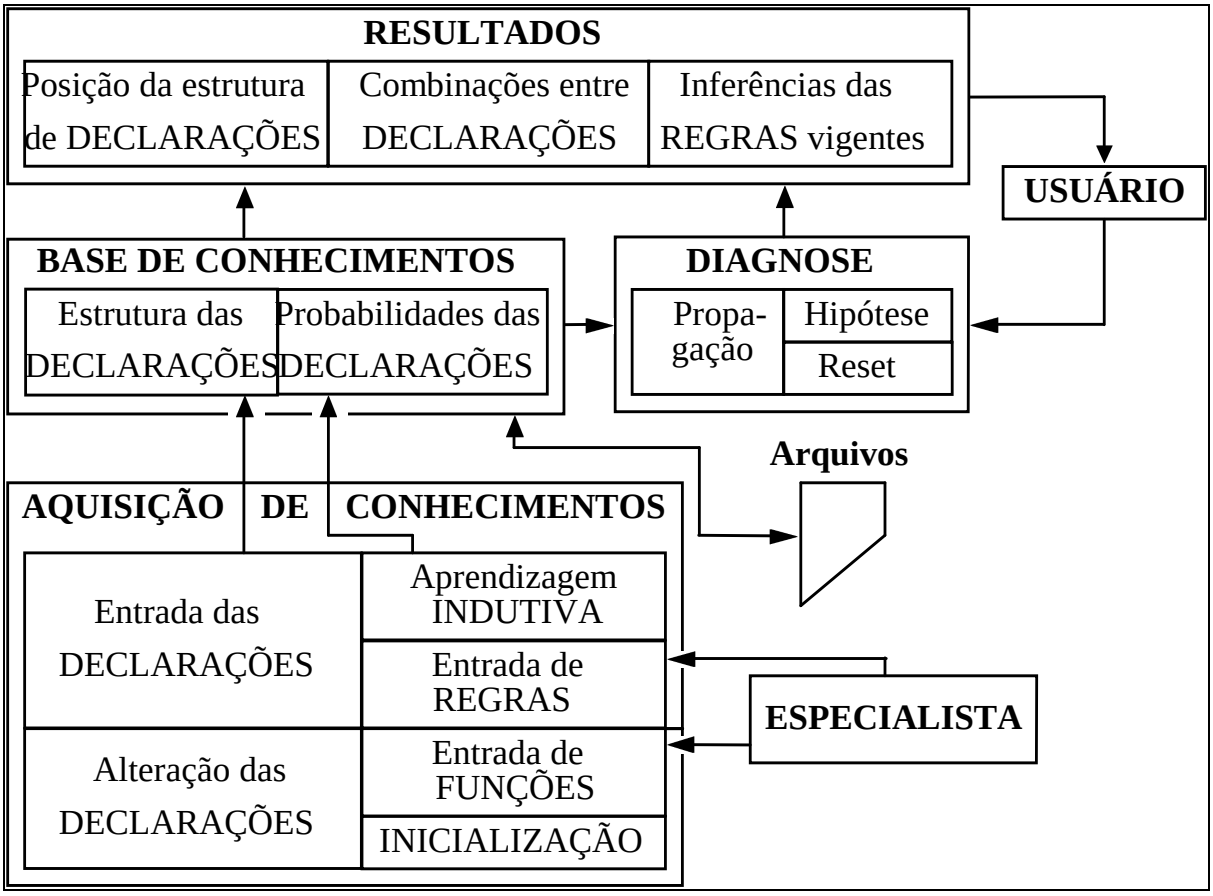
- *Reasoning*: raciocínio lógico;

- *Inference Networks*: a inferência é efetivada a partir da transformação de um grafo numa árvore de decisão;

- *Transition*: variáveis e suas relações podem estar em contínua mudança, e de acordo com um processo denominado de instanciamento de variáveis, é informado o estado em que se encontra um determinado atributo de uma ou mais variáveis.

Uma representação funcional do *SPIRIT* é dada pela Figura 10. A figura significa no *SPIRIT* a modificação temporária da base de conhecimentos.

Figura 10 - Estrutura Operacional do *SPIRIT*



Fonte: (Kopittke, 1993).

3.3.1 CARACTERÍSTICAS

As características, além das já definidas no item acima pelo próprio nome da ferramenta, podem ser descritas nos seguintes fundamentos:

a) abordagem bayesiana[5] de distribuição probabilística;

b) interpretação lógica das distribuições de probabilidades;

c) construção de distribuições através de fatos, regras e observações;

d) uso do conceito de variáveis com atributos (V/F), uma extensão do modelo permite sob certas condições o uso de variáveis discretas com múltiplos atributos;

e) uso do conceito de agrupamento local de variáveis (*Local Event Group – LEG*), que tornam possíveis o cálculo local de distribuições globais.

3.3.2 RECURSOS E FACILIDADES

A diferença fundamental entre o *SPIRIT* e os sistemas especialistas tradicionais (baseados em regras) consiste na forma de tratamento das regras. Enquanto os sistemas tradicionais tiram conclusões diretamente das regras, usando operadores extensionais, os probabilísticos visam primeiro construir uma distribuição sobre o espaço das realizações possíveis de todas as variáveis, sendo considerados os fatos e regras como probabilidades condicionais de eventos.

A principal vantagem de um sistema especialista probabilístico como os gerados pela ferramenta *SPIRIT* é o tratamento da incerteza sobre os conhecimentos. A lógica clássica na sua mais restrita forma considera uma proposição (regra) apenas como verdadeira ou falsa, o que limita severamente o tratamento de valores intermediários, muito comuns e necessários no tratamento do conhecimento subjetivo.

Exemplo: a proposição - um pássaro pode voar - pode ser considerada verdadeira, apesar dos pingüins não voarem. Mas no sentido lógico estrito, a proposição é falsa.

O *SPIRIT* é especialmente útil para representar, analisar e interpretar relações complexas. É indicado para aplicações que visam diagnose (diagnósticos e previsões), conhecer modelos (relações causa/efeito) e classificação (reconhecimento de padrões) (Kopittke, 1993).

---

[5] teoria estatística de evidências onde a noção fundamental é a probabilidade condicional.

Para usuário final também existem facilidades, pois além de utilizar uma ferramenta fácil para geração de sistemas especialistas, dispõe de uma interface gráfica, apresentada em três idiomas (inglês, português e alemão) que permite a realização de diagnósticos de uma forma simples e imediata.

3.3.2.1 REPRESENTAÇÃO DO CONHECIMENTO

A utilização de regras de produção permite modelar o conhecimento humano, o que o torna ideal para problemas de seleção, no qual uma determinada solução deve ser atingida a partir de um conjunto de seleções.

3.3.2.2 INFERÊNCIA

A inferência no sistema pode ser realizada nos dois sentidos: da premissa para a conclusão (encadeamento progressivo), e da conclusão para a premissa (encadeamento regressivo). Nesta mesma fase o *SPIRIT* gera uma distribuição conjunta de probabilidades. O sistema opera com base num tratamento matemático de distribuições marginais sobre o produto cartesiano de todos os atributos, e tanto os fatos como as regras, estabelecem as condições iniciais a partir das quais uma distribuição de probabilidade conjunta é processada (Kopittke, 1993).

Como método para tratamento da incerteza o *SPIRIT* utiliza a teoria das probabilidades com embasamento matemático no princípio da máxima entropia.

3.3.2.3 APRENDIZAGEM

Essa ferramenta pertence a classe dos sistemas de aprendizagem ativa e indutiva, pois a partir de informações do especialista são realizadas transformações na estrutura da base de conhecimentos. O processamento da base é iniciado, tomando como ponto de partida a distribuição uniforme de probabilidades. O processamento é concluído quando o processo converge para uma distribuição que atende o princípio da máxima entropia, que define as distribuições marginais das probabilidades. O processamento é efetuado de forma cíclica, para cada regra, até que um critério de término seja satisfeito.

Também relações indiretas, isto é, não diretamente observadas, são obtidas através de cálculos transitivos (propagação). A principal aptidão do sistema é estabelecer conclusões lógicas (no sentido de predicado lógico) a partir de uma dada distribuição de probabilidade conjunta.

3.3.2.4 JUSTIFICAÇÃO

Outra representação funcional do *SPIRIT* é uma janela que monitora o grafo de dependências, que funciona como um sistema de justificação, permitindo a visualização as probabilidades de sucesso. A diagnose pode ser realizada de forma direta e simples, clicando o ponteiro do mouse sobre o atributo desejado. Este procedimento é denominado instanciamento. A medida que o instanciamento é realizado, os valores das probabilidades associadas, direta e indiretamente, vão se modificando e produzindo um diagnóstico rápido, dada uma situação hipoteticamente possível, esta é a forma de visualização das inferências realizadas pelo sistema (Luchtenberg, 2000).

4 ENGENHARIA DE SOFTWARE E FERRAMENTAS

Este capítulo destina-se a introduzir o conceito de qualidade, e abordar aspectos da Engenharia de Software que serão utilizados para avaliar as ferramentas para sistemas especialistas.

4.1 QUALIDADE DE SOFTWARE

Para a atual norma brasileira sobre o assunto (NBR ISO 8482), qualidade de software pode ser definida como: “A totalidade das características de um produto de software que lhe confere a capacidade de satisfazer necessidades explícitas e implícitas”. Sendo entidade o produto que pode ser um bem ou um serviço. Necessidades explícitas são as condições e objetivos propostos pelo cliente e implícitas são as necessidades que variam de cliente para cliente como implicações éticas e de segurança.

Existem instituições normalizadoras reconhecidos mundialmente como:

- ISO – *International Organization for Standardization*;
- IEEE – Instituto de Engenharia Elétrica e Eletrônica;
- IEC – *International Eletrothcnical Comission*.

No Brasil a entidade que se responsabiliza pela adoção de normas e padrões é a ABNT (Associação Brasileira de Normas Técnicas), ela também é responsável pela pesquisa e desenvolvimento de normas muitas vezes em conjunto com os organismos internacionais.

Tanto a qualidade no produto final como no seu processo de produção são de grande importância para que se possa atingir da melhor maneira possível a satisfação do cliente, no entanto durante muito tempo apenas considerou-se o produto final. Com a evolução das pesquisas e a crescente necessidade de elevação da qualidade dos produtos devido a forte concorrência percebeu-se que o processo de construção do produto é um fator importantíssimo e de relevância fundamental na obtenção da qualidade esperada do produto (Costa, 2001).

A seguir são citadas as principais normas de qualidade de software:

- ISO 9126 - Característica da qualidade de produtos de software;
- NBR 13596 - Versão brasileira da ISO 9126;
- ISO 12119 - Característica de qualidade de Pacotes de Software (software de prateleira);
- ISO 9001- Modelos de qualidade em Projeto, Desenvolvimento, Instalação e Assistência Técnica (processo);
- ISO 9000-3 - Aplicação da norma ISO 9000 para desenvolvimento de software;
- ISO 10011 - Auditoria de sistemas de qualidade (processo);
- ISO 14598 - Guias para a avaliação da qualidade;
- ISO 15504 - SPICE, projeto da ISO para a avaliação do Processo de Desenvolvimento (*software process Improvement and Capability dEtermination*).

Sabendo-se o que é qualidade e como ela pode ser avaliada, pode se tentar aplicar estes conceitos aos produtos de software e ao processo de desenvolvimento de software. Inicialmente, encontra-se um grande problema: muitos desenvolvedores ainda acreditam que criar programas é uma arte que não pode seguir regras, normas ou padrões.

As instituições normalizadoras, citadas anteriormente, desenvolveram conjuntos de normas para serem aplicadas a produção de software. As normas foram desenvolvidas considerando-se:

a) produtos de software são complexos, até mais do que o hardware que o executam;

b) software não tem produção em série. Seu custo está no projeto e desenvolvimento;

c) software não se desgasta e nem se modifica com o uso;

d) software é invisível. Sua representação em grafos e diagramas não é precisa;

e) a Engenharia de Software ainda não está madura, é uma tecnologia em evolução;

f) não há um acordo entre os profissionais da área sobre o que é qualidade de software.

Mesmo levando-se em consideração todos esses fatores é importante ressaltar que o grande problema na definição de padrões de qualidade para a produção de software é o modo como os desenvolvedores têm feito isto durante vários anos, artezanalmente por isso é necessária a aplicação dos conceitos de qualidade na indústria de software urgentemente.

Por esse motivo proceder-se-á nesse trabalho a aplicação de uma das principais abordagens de certificação de produtos de software, a ISO/IEC 9126.

4.2 ISO/IEC 9126

A qualidade de produtos de software está descrita na norma ISO/IEC 9126, publicada em 1991, tendo sido traduzida no Brasil como NBR 13596 e publicada em agosto de 1996. Essa norma lista o conjunto de características que devem ser verificadas em um software para que ele seja considerado um "software de qualidade".

Seu objetivo é definir as principais características que devem ser encontradas em um software dito de qualidade, esta norma leva em consideração o produto pronto. Ela se divide em seis grandes características que são:

- Funcionalidade – determina se o software satisfaz as necessidades do cliente;
- Confiabilidade – determina se o software é imune a falhas;
- Usabilidade – determina se o software é fácil de ser usado;
- Eficiência – determina se o software é rápido;
- Manutenibilidade – determina se é simples de fazer manutenções no software;
- Portabilidade – determina se o software é fácil de usar em outro ambiente.

Cada uma destas características gera um conjunto de subcaracterísticas sendo que cada uma destas possui uma pergunta chave, como apresentado no Quadro 3 proposto por Costa (2001):

Quadro 3 – Conjunto de características e subcaracterísticas da ISO/IEC 9126

Característica	Subcaracterística	Pergunta chave para a subcaracterística
Funcionalidade (satisfaz as necessidades?)	Adequação	Propõe-se a fazer o que é apropriado?
	Acurácia	Faz o que foi proposto de forma correta?
	Interoperbilidade	Interage com os sistemas especificados?
	Conformidade	Está de acordo com as normas, leis, etc.?
	Segurança de acesso	Evita acesso não autorizado aos dados?
Confiabilidade (é imune a falhas?)	Maturidade	Com que freqüência apresenta falhas?
	Tolerância a falhas	Ocorrendo falhas, como ele reage?
	Recuperabilidade	É capaz de recuperar dados em caso de falha?
Usabilidade (é fácil de usar?)	Intelegibilidade	É fácil entender o conceito e a aplicação?
	Apreensibilidade	É fácil aprender a usar?
	Operacionalidade	É fácil de operar e controlar?
Eficiência (é rápido e "enxuto"?)	Tempo	Qual é o tempo de resposta, veloc. Execução?
	Recursos	Quanto recurso usa? Durante quanto tempo?
Manutenibilidade (é fácil de modificar?)	Analisabilidade	É fácil de encontrar uma falha, quando ocorre?
	Modificabilidade	É fácil modificar e adaptar?
	Estabilidade	Há grande risco quando se faz alterações?
	Testabilidade	É fácil testar quando se faz alterações?
Portabilidade (é fácil de usar em outro ambiente?)	Adaptabilidade	É fácil adaptar a outros ambientes?
	Capac. de instalação	É fácil instalar em outros ambientes?
	Conformidade	Está de acordo com padrões de portabilidade?
	Capac. para substituir	É fácil usar para substituir outro?

4.2.1 OBJETIVOS

Conforme Storch (2000), a norma ISO/IEC 9126 é um conjunto de padrões dirigido àqueles que estão envolvidos com aquisição, desenvolvimento, uso, suporte, manutenção e auditoria de software.

As organizações podem usar estes padrões em grande número de aplicações:

a) na possibilidade de determinação, pela organização, na capacidade de determinado software suportar um padrão internacional reconhecido;
b) para auxiliar a organização a melhorar seu próprio processo de desenvolvimento e manutenção de software;
c) como autojulgamento, para auxiliar uma organização a determinar sua capacidade de implementar um novo projeto de software.

5 ANÁLISE COMPARATIVA

5.1 CONSIDERAÇÕES PRELIMINARES

Para a realização de uma análise comparativa é preciso que existam critérios justificados para realizar esta tarefa. Na literatura disponível, não existe material que forneça critérios para a seleção dos parâmetros comparativos para ferramentas de sistemas especialistas. Da mesma forma, não existe um método para a coleta de informações que sustente esta seleção. Em virtude disso, buscou-se a elaboração e a fundamentação dos critérios analisando-se primeiramente como já exposto, os aspectos relacionados à qualidade de software, mais especificamente a ISO/IEC 9126, em seguida outros fatores relevantes para os sistemas especialistas, características, recursos e facilidades, abordadas no capítulo três.

Através desta análise, são propostos os seguintes critérios, para a comparação das ferramentas :

- características da qualidade e métrica ISO/IEC 9126;
- outros aspectos relevantes para sistemas especialistas.

Para a análise comparativa escolheram-se três ferramentas para sistemas especialistas, duas pertencentes a família das *shells: Expert SINTA* (versão 1.1b) e *SPIRIT* (1995) e uma linguagem de programação: *Arity Prolog* (versão1.1.88) .

Torna-se valido salientar que o foco deste trabalho não é a metodologia que será utilizada para comparar as ferramentas, nem tem-se como objetivo determinar qual a melhor ferramenta em detrimento das demais. O comparativo será realizado para que se possa investigar as potencialidades de cada uma das ferramentas submetidas.

5.2 CARACTERÍSTICAS DA QUALIDADE E MÉTRICA ISO/IEC9126

O critério proposto pela ISO/IEC 9126 consta de seis grandes características principais as quais são divididas em um conjunto de subcaracterísticas, abordado no capítulo quatro.

Segundo Storch (2000), ainda existem poucas métricas de aceitação geral, o que torna possível estabelecer modelos próprios de processo de avaliação e métodos, com as características dessa norma para atender as diversas áreas de aplicação.

Assim, neste trabalho cada atributo será avaliado de acordo com o questionário elaborado, ao qual deverá ser escolhida uma dentre as opções de resposta: Sim, Parcialmente e Não, analisando o grau de conformidade do software em relação ao atributo a ser avaliado. Como mostra o Quadro 4:

Quadro 4 – Perguntas ISO/IEC 9126

CARACTERÍSTICA	SUB CARACTERÍSTICA	PERGUNTA	ARITY PROLOG	EXPERT SINTA SHELL	SPIRIT
1.Funcionalidade	1.1 Adequação	1.1.1 O software possui todas as funções mencionadas no produto?	S	S	S
	1.2 Acurácia	1.2.1 O software é preciso na execução das funções?	S	S	S
		1.2.2 O software é preciso nos resultados?	S	S	S
	1.3 Interoperabilidade	1.3.1 O software tem restrições quanto ao número de estações trabalhando ao mesmo tempo?	N	N	N
	1.4 Conformidade	1.4.1 O software é conciso às leis vigentes?	S	S	S
	1.5 Segurança	1.5.1 O software dispõe de segurança de acesso através de senhas?	N	S	N
2. Confiabilidade	2.1 Maturidade	2.1.1 O software tem capacidade de continuar executando na ocorrência de erros de execução?	N	S	N
	2.2 Tolerância à falhas	2.2.1 O software possui advertência de erros cometidos pelo usuário?	S	S	S
		2.2.2 O software controla preenchimento de campos?	N	S	S
		2.2.3 O software tem capacidade de verificar se as informações cadastradas estão corretas?	S	S	S
		2.2.4 O software tem capacidade de evitar a inclusão de dados existentes?	N	S	S

		2.2.5 O software tem boa performance para o processamento com grande volume de dados?	S	S	N
	2.3 Recuperabilidade	2.3.1 O software tem capacidade de recuperar dados excluídos?	N	N	N
3. Usabilidade	3.1 Inteligibilidade	3.1.1 O software possui autodemonstração?	N	S	N
		3.1.2 As telas do software são autoinstrutivas, permitindo ao usuário visualizar com facilidade qual sua função?	P	S	P
		3.1.3 As telas do mesmo nível possuem o mesmo padrão?	S	S	S
		3.1.4 A ordem de apresentação dos menus segue uma lógica?	S	S	S
		3.1.5 O usuário precisa ter conhecimento na área para utilizar o software?	S	P	P
		3.1.6 Caso o usuário seja leigo, o software fornece as informações adequadas para sua perfeita utilização?	N	S	P
		3.1.7 A teoria que embasa o software é explicada com documentação impressa?	N	S	N
		3.1.8 Os jargões técnicos utilizados, são explicados para o usuário?	N	P	P
	3.2 Apreensibilidade	3.2.1 O software possui manual de operação?	S	S	N
		3.2.2 Todas as funções do software estão explicadas no manual?	S	S	N
		3.2.3 O manual apresenta todos os erros que podem ocorrer no software?	N	N	N
		3.2.4 A linguagem utilizada na mensagem de ajuda é facilmente entendida pelo usuário?	N	S	N
		3.2.5 O software mostra como o usuário deve navegar pelo arquivo de ajuda?	N	S	S
	3.3 Operacionalidade	3.3.1 Os comandos do software estão de acordo com os padrões	N	S	P

		existentes (F1, DEL, ESC, ENTER)?			
		3.3.2 É possível prever o que existe dentro de cada opção do menu?	S	S	S
		3.3.3 Existe padronização de teclas de função para todo o software?	S	P	P
4. Eficiência	4.1 Comportamento em Relação ao Tempo	4.1.1 O tempo necessário para inicializar o software é satisfatório?	S	S	P
		4.1.2 O tempo de resposta é adequado em relação ao volume de dados envolvidos?	S	S	P
5. Manutenibilidade	5.1 Analisabilidade	5.1.1 O software contém funções com objetivos específicos?	P	S	S
		5.1.2 O software possui decisões comentadas (sistema de justificação)?	N	S	N
		5.1.3 O nome de todas as variáveis do software são exclusivos?	S	S	S
		5.1.4 Todas as variáveis são inicializadas antes do uso?	N	N	N
6. Portabilidade	6.1 Adaptabilidade	6.1.1 O software pode ser facilmente modificado para atender às necessidades do usuário?	N	P	P
	6.2 Instalação	6.2.1 O software possui um programa de instalação?	S	S	S
		6.2.2 O software possui *help* de instalação?	N	N	N
	6.3 Substituição	6.3.1 O software tem capacidade de ser substituído por novas versões e continuar utilizando a mesma base de dados?	S	S	S

5.3 ASPECTOS RELEVANTES PARA SISTEMAS ESPECIALISTAS

Diante da existência de várias ferramentas para construção de sistemas especialistas (como as *shells* e linguagens de programação), a dificuldade remanescente para o projetista está em selecionar as ferramentas adequadas para que, com a máxima eficácia, consiga

projetar seus sistemas. Isto ocorre devido à grande variedade de opções, pela falta de informações que as equipes de projeto normalmente possuem acerca deste assunto e pela ausência de referências para proceder esta escolha. Esta dificuldade é ressaltada, pois apesar do constante crescimento e da variedade de recursos disponíveis para ajudar o projetista, existem poucas fontes disponíveis com informações objetivas sobre as características destas ferramentas.

O que expor-se-á neste capítulo, são justamente essas características, critérios que diferenciam sistemas especialistas. Segundo Bittencourt (1998) apud Stylianou (1992), baseado em uma pesquisa visando detectar as características mais importantes das ferramentas para sistemas especialistas, conforme usuários e projetistas. As dimensões escolhidas foram as seguintes: interface com o usuário, interface de desenvolvimento, interface com o sistema operacional, motor de inferência e representação do conhecimento.

5.3.1 INTERFACE COM O USUÁRIO

Em qualquer tipo de software, a interface com o usuário final é fundamental para o seu sucesso. Em particular, um sistema especialista, além de apresentar uma interface ergonomicamente bem projetada, deve ainda levar em conta o grau de familiarização do usuário com o domínio de trabalho do sistema e com os computadores em geral.

A atividade principal, e talvez até única das ferramentas para sistemas especialistas é receber informações e a partir delas gerar e/ou adicionar mais informações aos dados inicialmente fornecidos, para que estes sejam utilizados resolução de problemas.

A equipe de projeto deve observar se as informações recebidas, após o uso da ferramenta, são suficientes para a necessidade que apresentam. Caso o resultado não seja satisfatório, será necessário complementar esta informação com um estudo ou, até mesmo, com a utilização de uma outra ferramenta. Neste caso, a ferramenta em análise, deve ser considerada deficiente e deve ser analisada para verificar se existe algum problema em relação a interface com o usuário. Caso não seja adequado a utilização de apenas uma ferramenta, deve-se procurar o uso do menor número de ferramentas.

Algumas técnicas tornam a interface com o usuário mais amigável, por exemplo, o uso de janelas, menus, gráficos de alta resolução, animação, cores, etc. De todo modo, as telas a serem apresentadas ao usuário devem ser de fácil compreensão, e as explicações necessárias devem ser claras e diretas. Mais algumas características interessantes para interfaces com o usuário são: (i) disponibilidade de diversos tipos de interfaces, adaptadas ao tipo de usuário (iniciante, especialista); (ii) possibilidade de interromper a execução do sistema em um determinado ponto e poder retomá-la sem necessidade de reprocessamento; (iii) mensagens de erro claras e informativas; (iv) possibilidade de alterar certas entradas ao sistema e comparar os resultados obtidos; (v) capacidade de capturar e armazenar telas de execução.

Os aspectos considerados críticos foram: facilidade para explicação e documentação (Bittencourt,1998).

Considerando as ferramentas submetidas ao comparativo pode-se dizer que em relação a interface com o usuário, apresentam diferenças significativas. O programa gerado pelo *Arity Prolog* apresenta pouca utilidade para usuários que não tem conhecimento maior, a interface é considerada fraca, sem aspectos visuais relevantes e com mensagens de difícil entendimento. Já o *Expert SINTA,* ferramenta que melhor se apresentou neste aspecto, possui diversos recursos que ajudam o usuário a utilizar a ferramenta. Um sistema especialista implementado com o *Expert SINTA* comunica-se com o usuário final através de menus de múltipla escolha (ou escolha simples, se a variável em questão for univalorada). Estes menus são construídos automaticamente pela *shell*, mas alguns detalhes devem ser fornecidos pelo criador da base. Quando do seu uso, permite a definição de informações esclarecedoras sobre a base, indicação dos autores e definição de contextos de ajuda que podem ser vitais para o aproveitamento do sistema especialista. O *Expert SINTA* possibilita a inclusão de tópicos de ajuda para os valores possíveis de determinados atributos, associando a um arquivo no formato ajuda do *windows* a sua base. Apresenta ainda manual disponível na internet.

A ferramenta *SPIRIT*, ainda que levada em consideração ser uma ferramenta de origem estrangeira, apresenta-se com certa deficiência no critério interface com o usuário. Em sistemas com muitas variáveis e valores, ele apresenta alguma dificuldade, tendo a possibilidade de não serem visualizados algumas dessas variáveis. O *SPIRIT* tem o diferencial de se apresentar em três idiomas (inglês, português, alemão) e se utilizar de uma variedade de

cores. Mas não mostra de forma clara sua utilização pelo usuário, nem a justificação dos resultados e neste sentido não possui nenhum arquivo de ajuda.

5.3.2 INTERFACE DE DESENVOLVIMENTO

Quando o projetista ou a equipe de projeto utilizam uma ferramenta, estes buscam nela uma ajuda para a realização de sua atividade. Portanto, a ferramenta não pode ser um empecilho ao processo de projeto. Desta forma, se não existir uma maneira de facilitar a aprendizagem e o uso, implicará que o emprego daquela ferramenta, no processo de projeto, poderá não trazer os resultados desejados. Além de, com o tempo, o uso daquela ferramenta se tornar cansativo e desgastante, fazendo com que as pessoas responsáveis pelo desenvolvimento do sistema tendam a evitar a sua aplicação.

Voltado para este problema, o critério usabilidade (ver capítulo quatro) pode verificar qual o nível de dedicação que o projetista precisa dispensar nos processos de aprendizagem e de uso. Este critério está diretamente relacionado à maneira pela qual a ferramenta permite um uso mais fácil ou não, avaliando tanto o processo de aprendizagem como de utilização.

Por ser muito subjetivo, antes da avaliação por este critério, deve-se investigar o nível de dificuldade para o aprendizado, quais as atividades de preparação ao uso são necessárias e a maneira como é efetuado o processo de utilização. Para realizar esta investigação deve-se buscar informações sobre a necessidade de um treinamento prévio para a utilização, a leitura de manuais, preparativos ao uso, entre outros. Esta investigação deve ser repetida para cada uma das ferramentas. Porém, deve-se observar que, para determinadas aplicações, algumas ferramentas vão necessitar de uma mão-de-obra maior que outras, enquanto que para outras aplicações as mesmas ferramentas não terão o mesmo problema.

Correção de erros, manutenção de uma lista de comandos da seção, índice cruzado de símbolos e regras, visualização gráfica dos encadeamentos de regras utilizadas na solução, editor de regras, facilidades de explicação e uma boa documentação são outras características positivas para a interface de desenvolvimento.

Aspectos considerados críticos: facilidade para explicação, documentação, facilidade para customizar explicações e prototipagem rápida.

Neste critério a análise fica muito parecida com a do critério anterior, interface com o usuário. Os comentários expostos se colocam muito bem também neste critério.

O *Arity Prolog* por ser um ambiente de programação necessita do conhecimento do projetista nesta linguagem, tornado mais difícil o processo de implementação. O Prolog é deficiente no sentido da inexistência de ambientes e facilidades de programação adequados que permitam suportar o desenvolvimento e a utilização de programas de grande porte (editores potentes, facilidades gráficas, e facilidades de depuração que considerem as particularidades inerentes a uma linguagem baseada em lógica). Contudo, por ser uma linguagem consagrada, principalmente dentro da Inteligência Artificial, existem inúmeras bibliografias como referência.

Nas *shells*, o aspecto interface de desenvolvimento apresenta-se de forma bem mais clara, com geração automática do sistema. Estas ferramentas utilizam um modelo de representação do conhecimento baseado em regras de produção, tendo como objetivo principal simplificar o trabalho de implementação de sistemas especialistas através do uso de uma máquina de inferência compartilhada, da construção automática de telas e menus, do tratamento da incerteza nas regras de produção e da utilização de explicações sensíveis ao contexto da base de conhecimento modelada. Como as *shells* suprimem alguns componentes dos sistemas especialistas (capítulo dois) como quadro-negro, mecanismo de aprendizagem, dentre outros, o projetista tem que se preocupar somente com implementação da base de conhecimentos.

O *Expert SINTA* novamente aqui apresenta uma interface amigável ao desenvolvedor, com arquivo de ajuda e manual. Tanto *Expert SINTA* quanto o *SPIRIT* mostram a sua melhor forma, com editor para variáveis e regras, sendo que estes editores possuem verificação em tempo real, não permitindo inclusão de regras incompatíveis com a base de conhecimento, nem variáveis duplicadas ou exclusão de variáveis contidas nas regras existentes.

O *Expert SINTA* permite incluir variáveis com nomes em qualquer formato, inclusive com espaços em branco, dado assim maior legibilidade ao sistema.

O *SPIRIT* possui um certo controle, o nome de uma variável pode ser formado pela combinação de qualquer letra do alfabeto, sempre em letra maiúscula, com tamanho máximo de vinte e dois caracteres e os atributos podem ser criados pela combinação de qualquer letra

do alfabeto, sempre em minúsculo. Contudo, após criada as regras o SPIRIT não permite a inclusão de valores para variáveis dificultando o trabalho do projetista, que se vier a esquecer algum desses valores terá que reformular seu sistema.

Nas duas ferramentas citadas anteriormente, o nome de uma variável e/ou de seus atributos pode ser modificado a qualquer momento. Quando uma modificação desta é realizada, elas atualizam em toda a base de conhecimentos na nova denominação automaticamente.

Para se montar a base de conhecimentos as *shells* possuem um módulo próprio para inclusão de variáveis e de regras, expostos nas Figuras de 11 a 14:

Figura 11 – Editor de variáveis do *Expert SINTA*

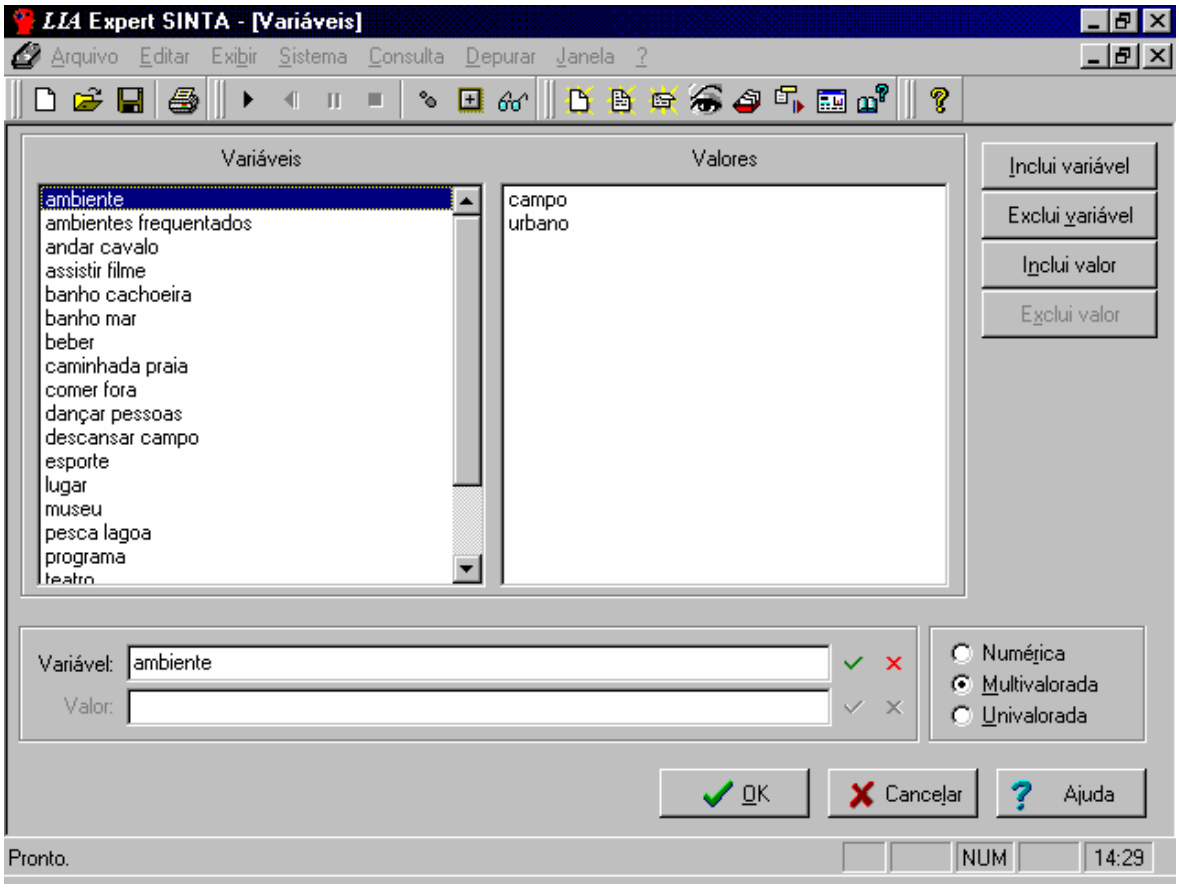


Figura 12 – Editor de regras do *Expert SINTA*

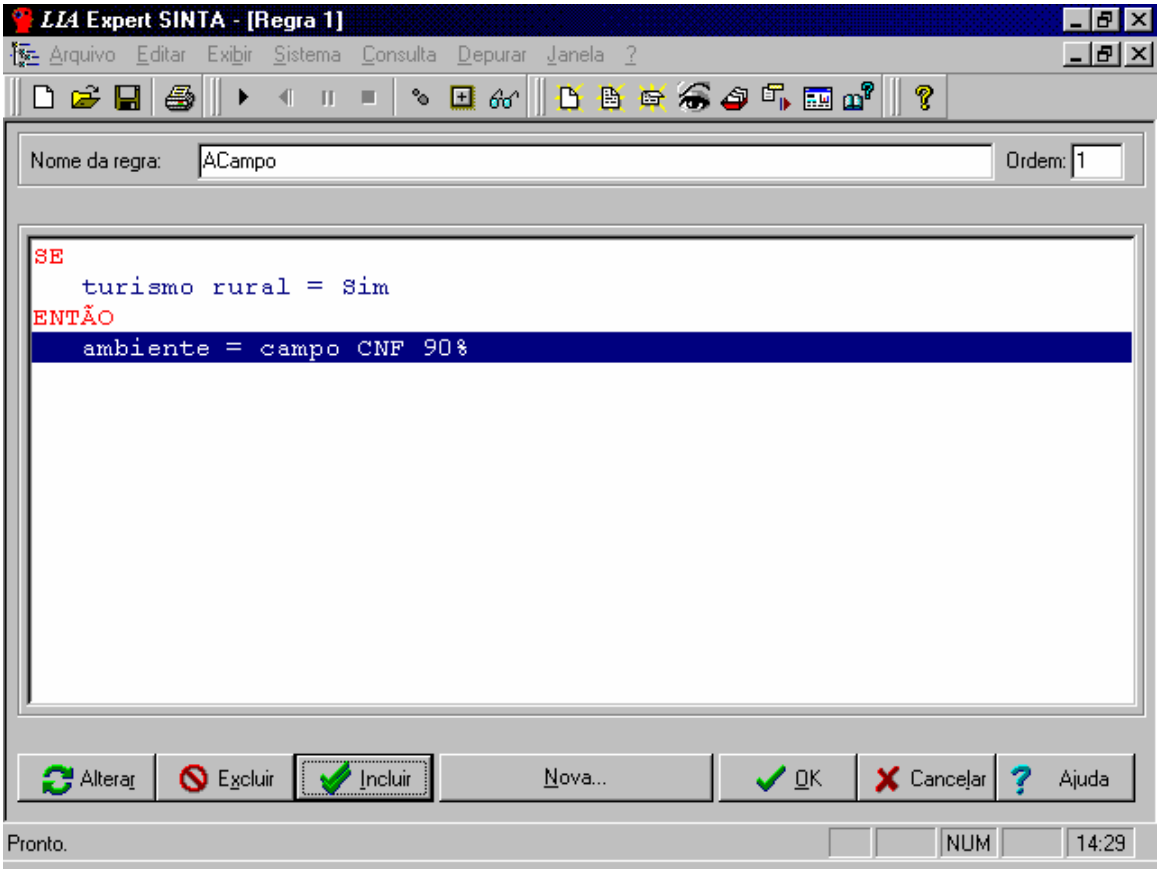


Figura 13 - Editor de variáveis do *SPIRIT*

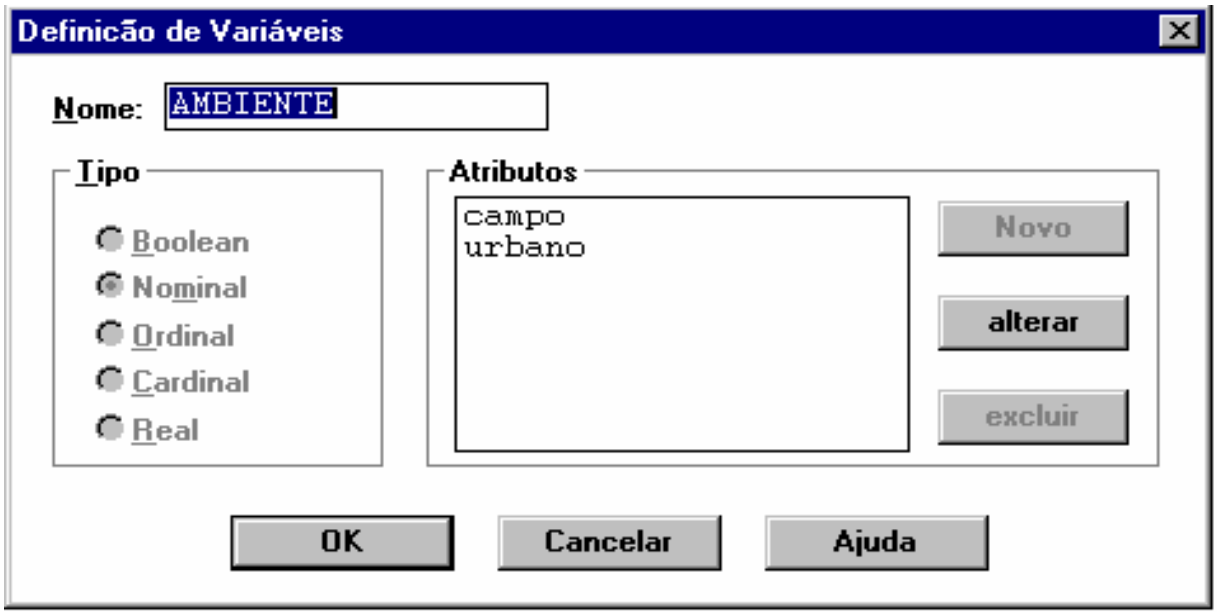
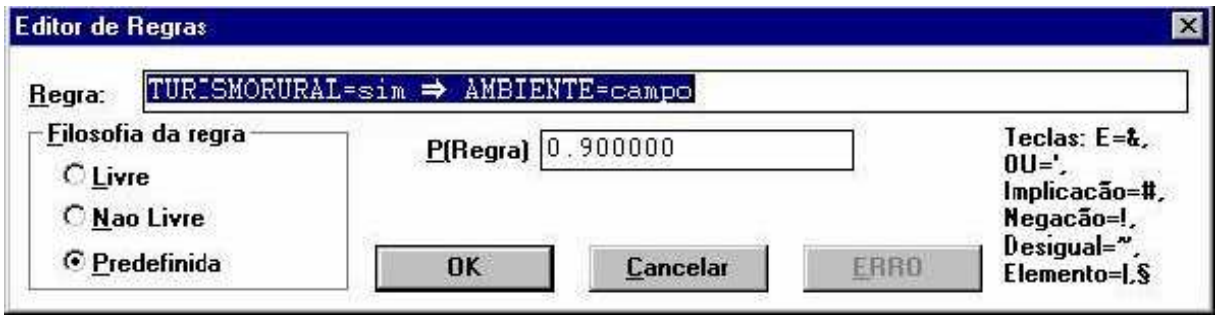


Figura 14 - Editor de regras do *SPIRIT*



5.3.3 INTERFACE COM O SISTEMA OPERACIONAL

Existem sistemas especialistas rodando em plataformas que variam de microcomputadores do tipo PC até *mainframes*, passando por estações de trabalho e máquinas especializadas em processamento simbólico, as chamadas "*Lisp Machines*". No entanto, setenta por cento dos sistemas especialistas pesquisados por Stylianou foram desenvolvidos para microcomputadores. Diversas ferramentas disponíveis no mercado rodam ainda em várias plataformas diferentes (Bittencourt, 1998).

Quanto à linguagem de programação, existem ferramentas escritas em linguagem de montagem (do inglês, *assembler*), C, Fortran, Basic, Forth, Pascal e Cobol, além, é claro, daqueles escritos em linguagens para Inteligência Artificial como *Lisp, Smalltalk* e o próprio Prolog.

A facilidade de comunicação com aplicações convencionais (como bancos de dados, planilhas e sistemas de rede) é considerada uma importante característica em um sistema especialista. O fato dessa facilidade ter sido relegada a um segundo plano nos primeiros sistemas especialistas representou uma grande dificuldade para sua disseminação em ambiente comercial. As ferramentas precisam fazer alguma coisa a mais para justificarem seu uso. Elas precisam facilitar a integração dos sistemas especialistas com outros tipos de programas. Os sistemas especialistas não podem operar no vácuo, como também não podem os humanos. Eles precisam acessar bancos de dados das corporações, e esse acesso precisa ser controlado como em outros sistemas. Eles em geral estão embutidos em programas aplicativos maiores, que usam basicamente técnicas de programação convencional. Então, uma das

características importantes que uma ferramenta precisa ter é uma interface entre o sistema especialista, e um ambiente de programação maior e provavelmente mais convencional.

Uma última característica ligada ao sistema operacional é o grau de segurança oferecido por um sistema especialista. É possível que um sistema especialista envolva informações confidenciais ou únicas que não devem ser acessadas e modificadas por qualquer usuário.

Aspectos considerados críticos: facilidade para integração com sistemas já existentes.

As três ferramentas rodam em plataforma *windows*. No aspecto relacionado a comunicação com sistemas convencionais as ferramentas *Arity Prolog* e *SPIRIT* não apresentaram nenhuma restrição contida na bibliografia pesquisada, no Expert *SINTA* sabe-se que ele tem facilidade de comunicação com ambiente de programação *Delphi*.

Quanto a linguagem de programação utilizada o *SPIRIT* utiliza C++ e *Expert SINTA* foi construído no ambiente *Borland Delphi*.

Em relação a segurança a única ferramenta que apresenta algo relacionado é *o Expert SINTA*. Ele permite três níveis de proteção (LIA, 1995):

Permitir execução e visualização

- usuário do seu sistema especialista não tem permissão para modificar a base, mas pode executá-la, depurá-la e imprimi-la. É uma opção para evitar que estranhos modifiquem a base, fazendo com que perca sua consistência.

Permitir somente execução

- um pouco mais polêmica, essa opção permite que o usuário sem senha apenas possa executar o sistema especialista, mas não pode modificá-la e menos ainda visualizá-la. Porém, um sistema especialista que não exibe o conteúdo de suas regras, não permite acompanhar sua execução e verificar como atingiu o resultado exibido não merece a mesma confiabilidade de um sistema que permite o completo acesso às suas regras. O principal motivo que levaria um projetista a escolher essa opção de restrição é o perigo de pirataria do conteúdo do sistema especialista. Mas, recomenda-se esquecer o risco e dar prioridade à confiabilidade de seu trabalho.

Além disso, um sistema especialista que não é protegido por nenhuma lei de direitos autorais dá sempre a impressão de não ser tão profissional quanto deveria.

Nenhuma permissão

- somente pessoas com senha podem utilizar esse sistema especialista.

5.3.4 MOTOR DE INFERÊNCIA

As principais características do motor de inferência disponível nos sistemas especialistas dizem respeito às seguintes funcionalidades: método de raciocínio, estratégia de busca, resolução de conflito e representação de incerteza. Dentre todos estes aspectos, foi considerada crítica a existência dos métodos de raciocínio.

5.3.4.1 MÉTODO DE RACIOCÍNIO

Em geral, os sistemas especialistas adotam apenas um modo de raciocínio; no entanto, existem alguns que permitem ambos os modos, mas de maneira independente, e ainda outros que permitem um encadeamento misto, onde os encadeamentos progressivo e regressivo se alternam de acordo com o desenvolvimento da solução do problema e com a disponibilidade de dados.

Uma característica importante do modo de raciocínio se refere à monotonicidade ou não do método de inferência. Sistemas monotônicos não permitem a revisão de fatos, isto é, uma vez um fato declarado verdadeiro, ele não pode mais tornar-se falso. Sistemas não monotônicos, por outro lado, permitem a alteração dinâmica dos fatos. O preço desta capacidade é a necessidade de um mecanismo de revisão de crenças, pois uma vez que um fato, antes verdadeiro, torna-se falso, todas as conclusões baseadas neste fato também devem tornar-se falsas (Bittencourt, 1998).

Uma vez definido o tipo de encadeamento, o motor de inferência necessita ainda de uma estratégia de busca para guiar a pesquisa na memória de trabalho e na base de regras e ao terminar o processo de busca, o motor de inferência dispõe de um conjunto de regras que satisfazem à situação atual do problema, o chamado conjunto de conflito, porém esses aspectos não estão referenciados nas *ferramentas*.

A estrutura de controle imposta a um programa em Prolog pelo interpretador é a mesma utilizada pelas *shells*. A entrada é um objetivo a ser provado e o raciocínio para trás é utilizado para tentar provar o objetivo dadas declarações do programa. O programa é lido de cima para baixo, da esquerda para direita e a busca executada é em profundidade com retrocesso (Rich, 1993).

O *Expert SINTA* trabalha naturalmente com encadeamento para trás, mas é possível manipular a máquina de inferência de modo a simular encadeamento para frente, porém somente de maneira independente. Esta ferramenta também trata o conhecimento de forma essencialmente monotômica (LIA, 1995).

Neste sentido o *SPIRIT* leva vantagens, pois a inferência no sistema pode ser realizada nos dois sentidos.

5.3.4.2 REPRESENTAÇÃO DE INCERTEZA

Diversos métodos foram propostos para tratar este problema, por exemplo, método Bayesiano, fatores de certeza, teoria de Dempster-Shafer, teoria dos conjuntos nebulosos, teoria de probabilidades subjetivas e teoria de possibilidades (Bittencourt, 1998).

De maneira geral, estes métodos atribuem aos fatos e regras uma medida numérica que represente de alguma forma a “confiança” do especialista. Os métodos utilizados não são necessariamente coerentes uns com os outros e cada método adapta-se melhor a determinados tipos de problemas. Diversos sistemas especialistas dispõem de mais de um método de tratamento de incerteza, deixando ao usuário a escolha do mais adequado ao seu problema. Uma característica freqüente desses métodos é a existência de um limite mínimo para a medida de incerteza, abaixo do qual o fato ou regra é desconsiderado. Este limite pode, em geral, ser fixado pelo usuário.

Cada uma das ferramentas propostas ao comparativo possui um modelo formal conhecido de tratamento. A seguir descreve-se os modelos matemáticos utilizados por estas ferramentas para representação da informação imperfeita.

O *Arity Prolog* por utilizar representação lógica não apresenta um tratamento para incerteza, mas permite ao programador implementar esse critério. A seguir descrever-se-á um exemplo de implementação exposto por Bratko (1990):

Primeiro deve-se entender a estrutura da regra proposta, conforme Quadro 5:

Quadro 5 – Exemplo de implementação da incerteza em Prolog

```
RuleName: If
            Condition
          Then
            Conclusion
          With
            strength(S,N).
```

Fonte: (Bratko, 1990).

A cada regra pode ser somada um "intensificador" definido por dois números reais positivos (S,N). Um formato satisfatório seria (Quadro 6):

Quadro 6 – Exemplo de implementação da incerteza

```
Ambiente1: Se
            Turismo Rural=Sim
          Então
            Ambiente =Campo
          Com
            intensidade(0.001, 2000).
Lugar1: Se
            Ambiente=Campo
            E Andar Cavalo=Sim
            E Pesca Lagoa=Sim
          Então
```

```
                Lugar=Fazenda

            Com

                intensidade(0.05, 400).
```

Outras implementações podem ser feitas através de outras procedures.

O *Expert SINTA* além do uso de graus de confiança para tratamento de incertezas, e da "não-instanciação" de variáveis, permite o uso de um valor especial, passível de ser usado por todas as variáveis. É o desconhecido, que representa uma indeterminação total sobre as instanciações de uma variável. Este valor pode ser tanto incluído nas regras como dado como resposta pelo usuário. O embasamento matemático utilizado por esta ferramenta é a teoria das possibilidades, não envolvendo nenhum tratamento estatístico mais aprofundado. Constituindo assim, um modelo que alia um grande poder de expressividade com uma grande flexibilidade para o tratamento da informação incerta. Como vê-se a seguir nos exemplos de cálculo do grau de confiança:

a) cálculo do grau de confiança com o operador E;

Se possuímos duas igualdades $var_1 = value_1$ e $var_2 = value_2$, com os respectivos graus de confiança c_1 e c_2, temos que a sentença $var_1 = value_1$ E $var_2 = value_2$ retornará como valor de confiança $c_1 \times c_2$.

b) cálculo do grau de confiança com o operador OU.

Se possuímos duas igualdades $var_1 = value_1$ e $var_2 = value_2$, com os respectivos graus de confiança c_1 e c_2, temos que a sentença $var_1 = value_1$ OU $var_2 = value_2$ retornará como valor de confiança $c_1 + c_2 - c_1 \times c_2$.

Quando uma variável recebe duas vezes o mesmo valor em pontos diferentes da consulta o grau de confiança é calculado semelhante ao operador OU. O sistema admite 50% como valor mínimo de confiança para que uma igualdade seja considerada verdadeira, porém esse valor pode ser modificado. O intervalo de grau de confiança varia de 0 a 100. É possível ainda mudar as fórmulas utilizadas.

No *SPIRIT* o sistema opera com a incerteza através do método probabilístico, mais especificamente o princípio da máxima entropia. Seguidamente tem-se utilizado esse método

em casos em que não se sabe nada sobre as relações a priori existentes entre as variáveis aleatórias. O princípio da máxima entropia visa construir uma distribuição sobre o espaço das realizações possíveis de todas as variáveis, sendo considerados fatos e regras como probabilidades condicionais de eventos.

Exemplo: Considere as variáveis binárias Sexo e Neurose com os valores m/f e s/n, respectivamente, e o seguinte conjunto de regras R:

R = { P (sexo = f | neurose = s) = 0.9,
P (sexo = m) = 0.6 }

R define um sistema linear de (in)equações, cujo conjunto de soluções caracteriza todas as distribuições possíveis. A este conjunto de soluções chamamos de (R).

O Quadro 6 abaixo mostra as realizações possíveis e as probabilidades em questão; o sistema linear determina o conjunto de soluções com parâmetro p2.

Quadro 7 - Exemplo do princípio da máxima entropia

Sexo	Neurose	P	Fatos e Regras	P
m	n	p1 = ?	p4 = 0.9 (p2 + p4)	p1 = 0.5 - p2
m	s	p2 = ?	p1 + p2 = 0.5	p3 = 0.5 - 9p2
f	n	p3 = ?	p1 + p2 + p3 + p4 = 1	p4 = 9p2
f	s	p4 = ?	pi >= 0, i = 1 a 4	

Para 0 <= p2 <= 1/18 as soluções, então, são viáveis; desta forma o conjunto de regras é consistente.

A partir de um tratamento matemático de distribuições marginais sobre o produto cartesiano de todos os atributos de todas as variáveis, e tanto os fatos como as regras, estabelecem as condições iniciais a partir das quais uma distribuição de probabilidade conjunta é processada. Contudo, um problema comum e crítico é que esta distribuição da probabilidades cresce de forma exponencial, de acordo com o número de variáveis e respectivos atributos. Isto implica em dois problemas de natureza operacional (computacional), isto é, de espaço (memória) e de tempo de processamento. O sistema admite

como padrão 50% o valor mínimo de certeza para que uma igualdade seja considerada verdadeira, esse valor também pode ser alterado.

5.3.5 REPRESENTAÇÃO DO CONHECIMENTO

Em geral, os sistemas especialistas se limitam a oferecer um único tipo de representação de conhecimento. Alguns sistemas dispõem de mais de um formalismo, os quais, no entanto, devem ser utilizados de maneira isolada. Alguns poucos sistemas especialistas possuem os chamados sistemas híbridos de representação de conhecimento que, além de possuir diversos formalismos de representação, dispõem também de algoritmos de acesso que integram os conhecimentos representados nos diversos formalismos para permitir sua utilização de maneira integrada (Bittencourt, 1998).

A linguagem Prolog é basicamente um subconjunto da lógica dos predicados. Esse subconjunto recebeu o nome de cláusulas de *Horn*. Segundo Araribóia (1989), são representações lógicas de primeira ordem, ou seja, representações que permitem quantificação sobre os indivíduos, mas não sobre os predicados. A representação em lógica de primeira ordem provém de uma sintaxe bem formada, uma semântica clara e, acima de tudo, a noção de verdade e inferência.

Tanto o *Expert SINTA* como o *SPIRIT* utilizam-se como forma de representação do conhecimento somente as regras de produção.

5.4 RESULTADOS

Para melhor visualização elaborou-se de forma sucinta um quadro com os parâmetros comparativos em relação as ferramentas (Quadro 8):

Quadro 8 – Resumo do Comparativo

		ARITY PROLOG	**EXPERT SINTA**	**SPIRIT**
Interface com o Usuário		Editor DOS, com pouca qualidade e de difícil manuseio. Interage com o usuário através de seqüências de perguntas	Interface boa, de fácil compreensão e com sistema de justificação. Comunica-se com usuário através de janelas com perguntas	Apresentada através do grafo de dependências com instanciações das variáveis através de "clicks" do mouse. Permite o acompanhamento dos valores de todas as variáveis
Interface de Desenvolvimento		Apresenta um editor próprio no formato DOS de difícil manuseio	Possui editor de regras e variáveis.	Possui editor de regras e variáveis
Interface com o Sistema Operacional		Por ser a própria linguagem de programação tem flexibilidade, podendo ser construído junto ao sistema e implementada algum tipo de proteção a base de conhecimentos	Criado em C++. Não possui sistema de proteção a base de conhecimentos	Implementado no ambiente *Borland Delphi*. Permite comunicação com sistemas convencionais e possui sistema de proteção a base de conhecimentos
Motor de Inferência	**Método de Raciocínio**	Encadeamento para trás	Encadeamento para trás e encadeamento para frente	Encadeamento para trás com possibilidade de simulação do encadeamento para frente
	Representa-ção da Incerteza	Não apresenta tratamento para incerteza, porém permite que seja implementada rotina específica	Probabilidades	Fator de Confiança
Representação do Conhecimento		Lógica dos predicados	Regras de produção	Regras de produção

6 APLICAÇÕES EXPERIMENTAIS

Como forma de fazer o comparativo entre as ferramentas estudadas neste trabalho, foi implementada a mesma base de conhecimento em cada ferramenta e observado os resultados a que elas chegaram.

6.1 APLICAÇÃO EXPERIMENTAL 1

A aplicação experimental 1 é constituída conhecimentos relativos a área de turismo, que adicionados as respostas do usuário auxiliarão na determinação do programa para o feriado.

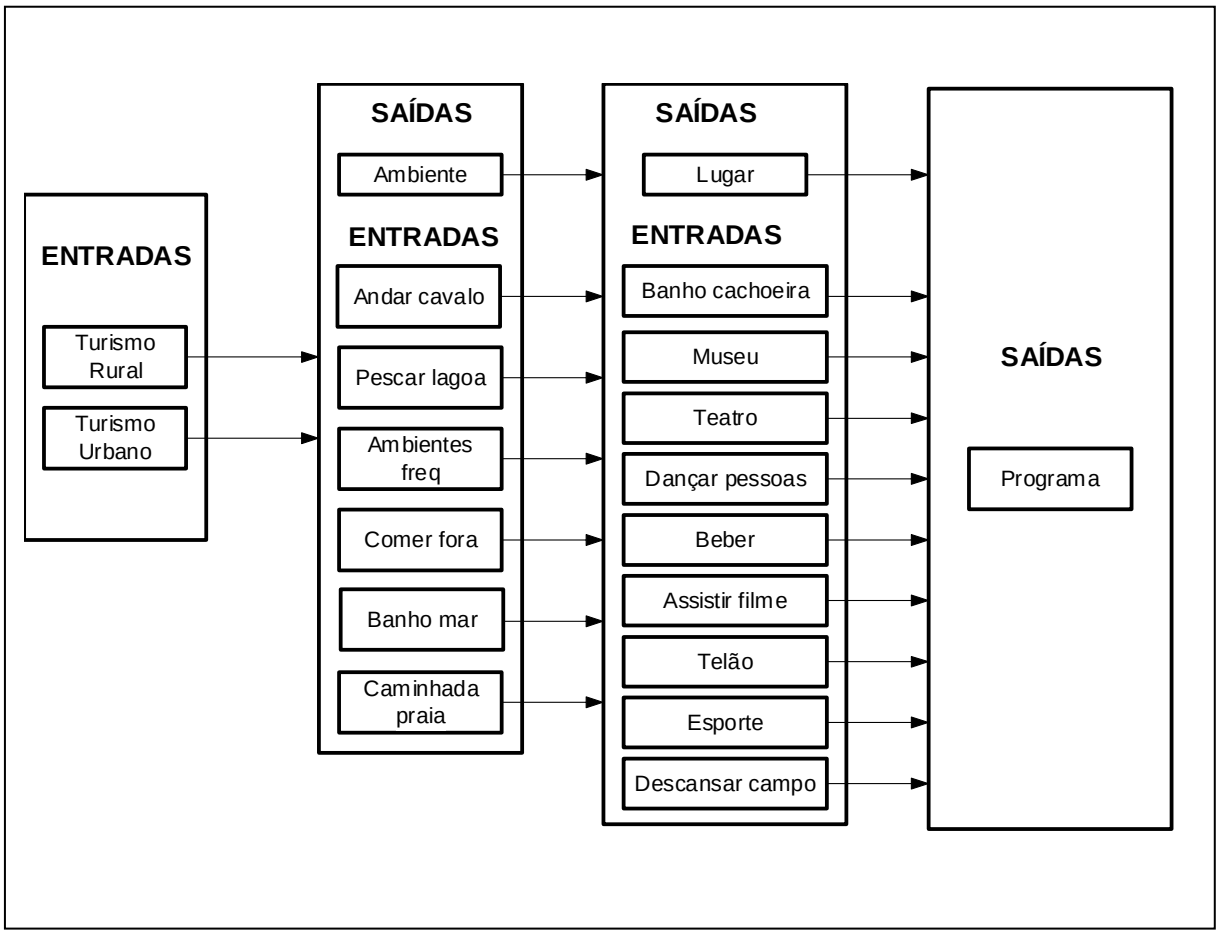
6.1.1 AVALIAÇÃO DO PROBLEMA

A elaboração da base desta aplicação envolve vinte variáveis (Anexo A) e utiliza onze regras, formando uma árvore de três níveis. As conclusões possíveis de serem alcançadas, a partir dessa base, permitem ao usuário determinar o programa para o feriado.

O desenvolvimento do sistema especialista seguiu os passos estabelecidos no capítulo dois, porém algumas das fases tiveram que ser suprimidas, pois a intenção aqui foi a criação de experimentos com fins de fazer um comparativo. Sendo assim expor-se-á somente as fases de avaliação do problema e teste.

Para melhor entendimento elaborou-se uma topologia da árvore formada pelo experimento 1, através Figura 15 torna-se possível observar os níveis e variáveis (Souza, 2001):

Figura 15- Topologia Experimento 1



6.1.2 ESPECIFICAÇÃO

A representação do conhecimento por regras de produção é a forma mais utilizada em sistemas especialistas. A justificativa é a naturalidade que representa para o homem pois o par "condição-ação", para raciocinar e decidir, também é usado pela mente humana. Desta forma escolheu-se também as regras de produção como forma de especificação do problema, como pode-se observar no Quadro 9:

Quadro 9 – Regras de produção do experimento 1

```
Se Turismo Rural=Sim
Entao
    Ambiente=Campo

Se Turismo Urbano=Sim
Entao
    Ambiente=Urbano

Se Ambiente=Campo
e  Andar Cavalo=Sim
ou Pesca Lagoa= Sim
Entao
   Lugar=Fazenda

Se Ambiente=Urbano
e  Ambientes Frequentados=Sim
ou Comer Fora=Sim
Entao
   Lugar=Cidade

Se Ambiente=Urbano
e  Banho Mar=Sim
ou Caminhada Praia=Sim
Entao
   Lugar=Litoral

Se Lugar=Fazenda
e  Banho Cachoeira=Sim
Entao
   Programa=Cachoeira

Se Lugar=Cidade
e Museu=Sim
ou Teatro=Sim
Entao
    Programa=Visitas

Se Lugar=Cidade
e Dançar Pessoas=Sim
e Beber=Sim
Entao
  Programa=Festas

Se Lugar=Cidade
e Assistir Filme=Sim
e Telao=Sim
Entao
```

```
            Programa=Cinema

         Se Lugar=Litoral
         e  Esporte=Sim
         Entao
            Programa=Esportes Aquáticos

         Se Lugar=Fazenda
         e Descansar campo=Sim
         Entao
            Programa= Hotel Fazenda
```

6.1.3 TESTES EFETUADOS

Para que pudesse proceder o comparativo foram informadas às ferramentas opções que levassem ao mesmo resultado. Já como resultado desse comparativo foi possível observar que não existe a possibilidade de serem informadas exatamente as mesmas opções, pois cada ferramenta tem seu modelo de tratar as instanciações. Sendo assim, foi aplicado um caso que levasse a conclusão que a programação para feriado seria hotel fazenda, levando em consideração as variáveis: turismo rural, andar cavalo e descansar campo como positivas.

6.1.4 ANÁLISE DOS RESULTADOS

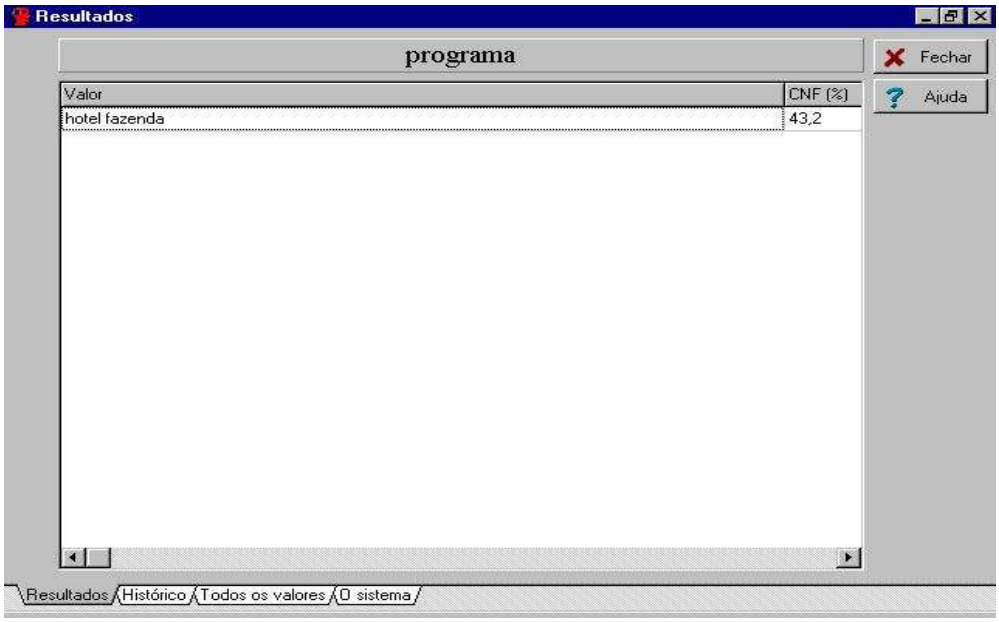
Da aplicação do experimento 1 nas ferramentas, obtiveram-se os seguintes resultados:

A linguagem de programação *Arity Prolog*, através da implementação de um programa conduziu ao resultado hotel fazenda, como mostrado pelo Figura 16:

Figura 16- Resultado experimento 1 pelo *Arity Prolog*

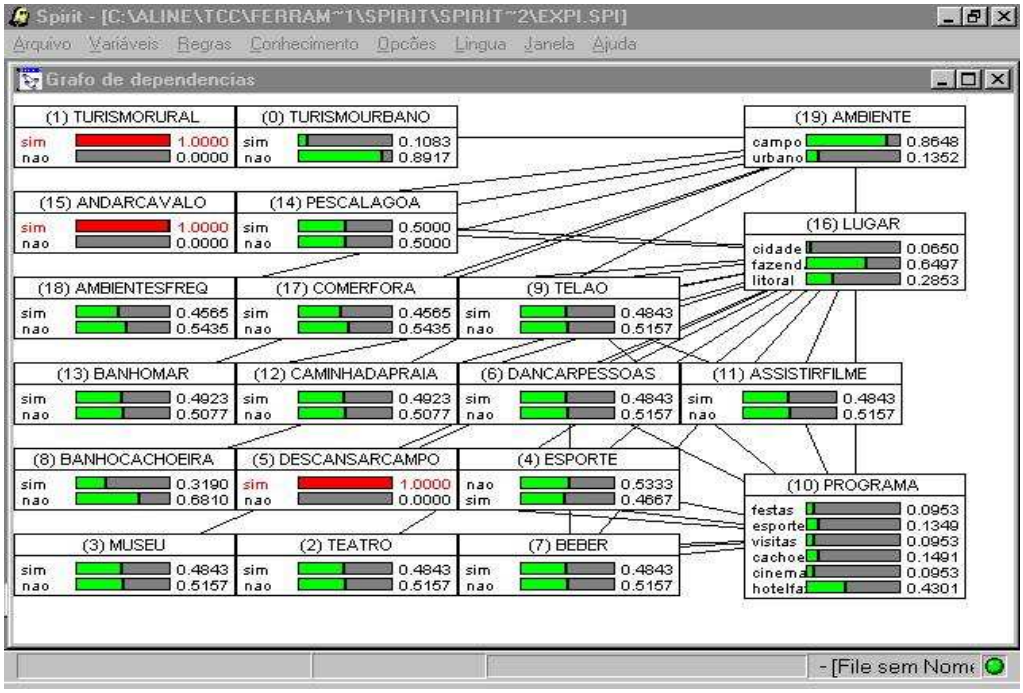
A ferramenta *Expert SINTA* obteve o resultado de programa hotel fazenda com 43,20 de fator de certeza, como mostra a tela de resultado apresentada na Figura 17:

Figura 17- Resultado experimento 1 pelo *Expert SINTA*



A ferramenta *SPIRIT* através do tratamento probabilístico obteve como resultado programa hotel fazenda com 43% de probabilidade, como mostra o seu grafo de dependências na Figura 18:

Figura 18 - Resultado experimento1 pelo *SPIRIT*



6.2 APLICAÇÃO EXPERIMENTAL 2

A aplicação experimental 2 é constituída conhecimentos relativos a área da seguridade social, que adicionados as respostas do usuário auxiliarão na determinação do tipo de aposentadoria por idade.

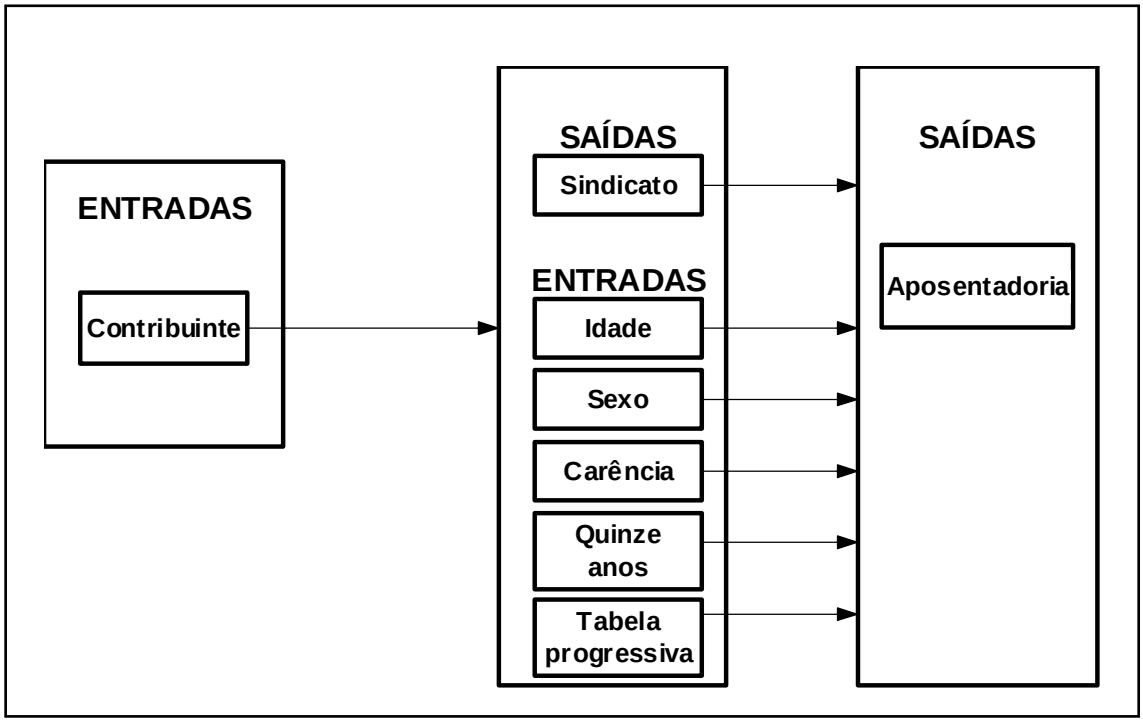
6.2.1 AVALIAÇÃO DO PROBLEMA

A elaboração da base de conhecimentos desta aplicação envolve oito variáveis (Anexo B) e utiliza seis regras, formando uma árvore de dois níveis. As conclusões possíveis de serem alcançadas, a partir dessa base, permitem ao usuário determinar o tipo de aposentadoria por idade.

Assim como o experimento 1 todas as fases de desenvolvimento de sistemas especialistas expostas pelo capítulo dois foram seguidas. A fase de aquisição do conhecimento foi realizada com base nas leis do sistema previdênciário brasileiro. E as fases de projeto, documentação e manutenção foram suprimidas.

A Figura 19 mostra a topologia da árvore formada pelo experimento 2 (Souza, 2001):

Figura 19 - Topologia Experimento 2



6.2.2 ESPECIFICAÇÃO

Obteve-se a especificação da aplicação experimental 2 através de seis regras de produção, mostradas no Quadro 10:

Quadro 10 - Regras experimento 2

```
Se Contribuinte=Rural
Entao
   Sindicato= Rural

Se Contribuinte= Empregado
ou Contribuinte= Domestico
ou Contribuinte= Autonomo
ou Contribuinte= Facultativo
ou Contribuinte= Empresario
Entao
   Sindicato= Urbano

Se Idade>= 55
e Sexo=F
e Carencia=Sim
e Sindicato= Rural
e Quinzeanos=Sim
ou Tabela Progressiva=Sim
Entao
  Aposentadoria= Idade Rural Mulher

Se Idade>= 60
e Sexo=M
e Carencia=Sim
e Sindicato= Rural
e Quinzeanos=Sim
ou Tabela Progressiva=Sim
Entao
  Aposentadoria= Idade Rural Homem

Se Idade>= 65
e Sexo=M
e Carencia=Sim
e Sindicato= Urbano
e Quinzeanos=Sim
ou Tabela Progressiva=Sim
Entao
  Aposentadoria= Idade Urbana Homem

Se Idade>= 60
e Sexo=F
e Carencia=Sim
e Sindicato= Urbano
e Quinzeanos=Sim
ou Tabela Progressiva=Sim
Entao
  Aposentadoria=Idade Urbana Mulher
```

6.2.3 TESTES EFETUADOS

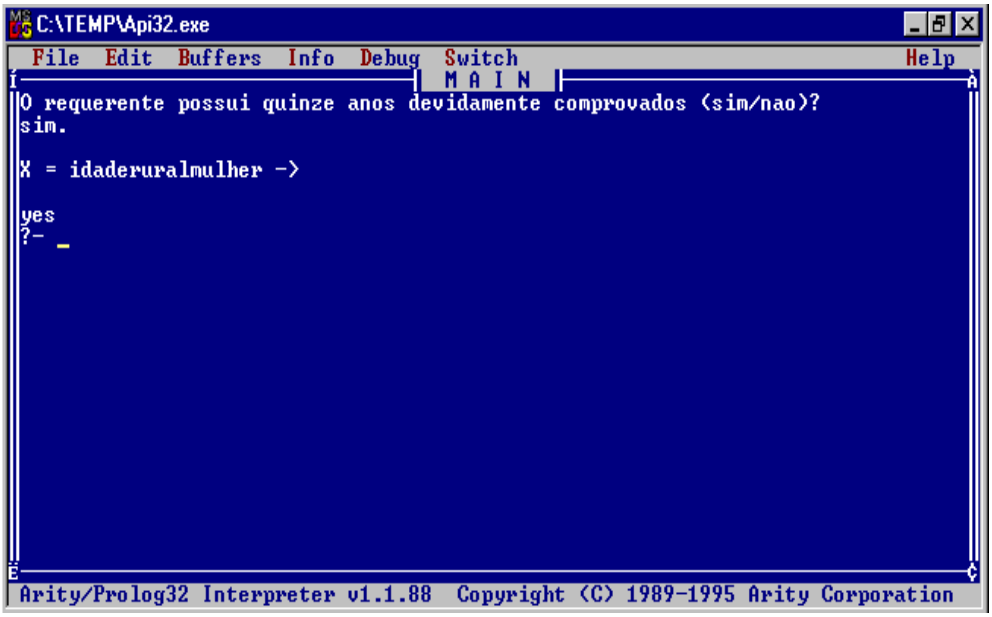
Como forma de testar a potencialidade de cada ferramenta na aplicação experimental 2, foi elaborado um caso que levasse a aposentadoria por idade rural para mulher, considerando para isso as variáveis: contribuinte=rural, idade=55, carência=sim e quinzeanos=sim.

6.2.4 ANÁLISE DOS RESULTADOS

Os resultados obtidos chegam as seguintes conclusões:

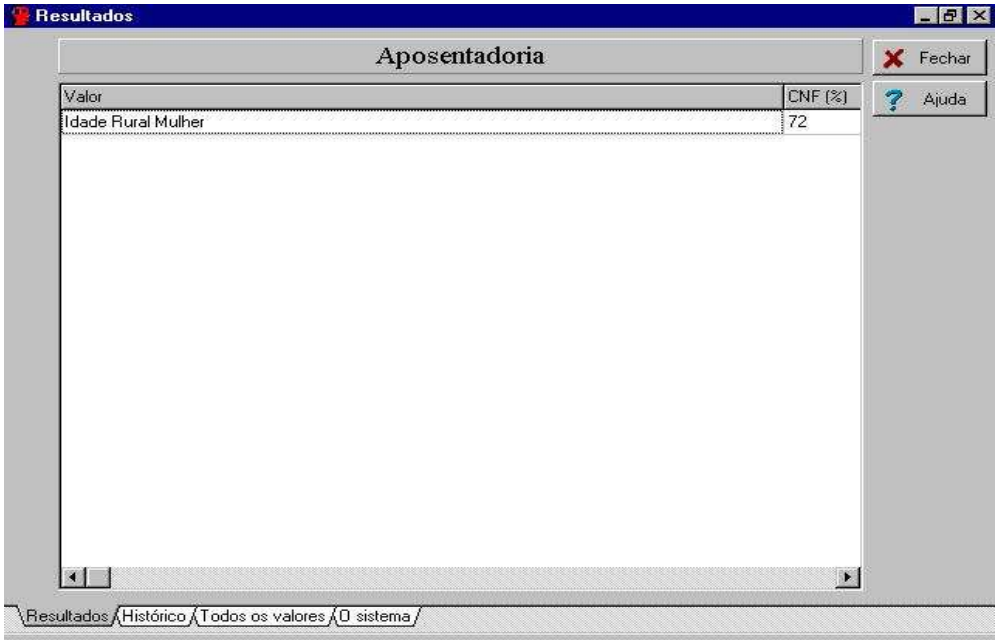
A implementação através do *Arity Prolog* chegou ao resultado aposentadoria idade rural mulher, mostrado na Figura 20:

Figura 20 - Resultado experimento 2 pelo *Arity Prolo*



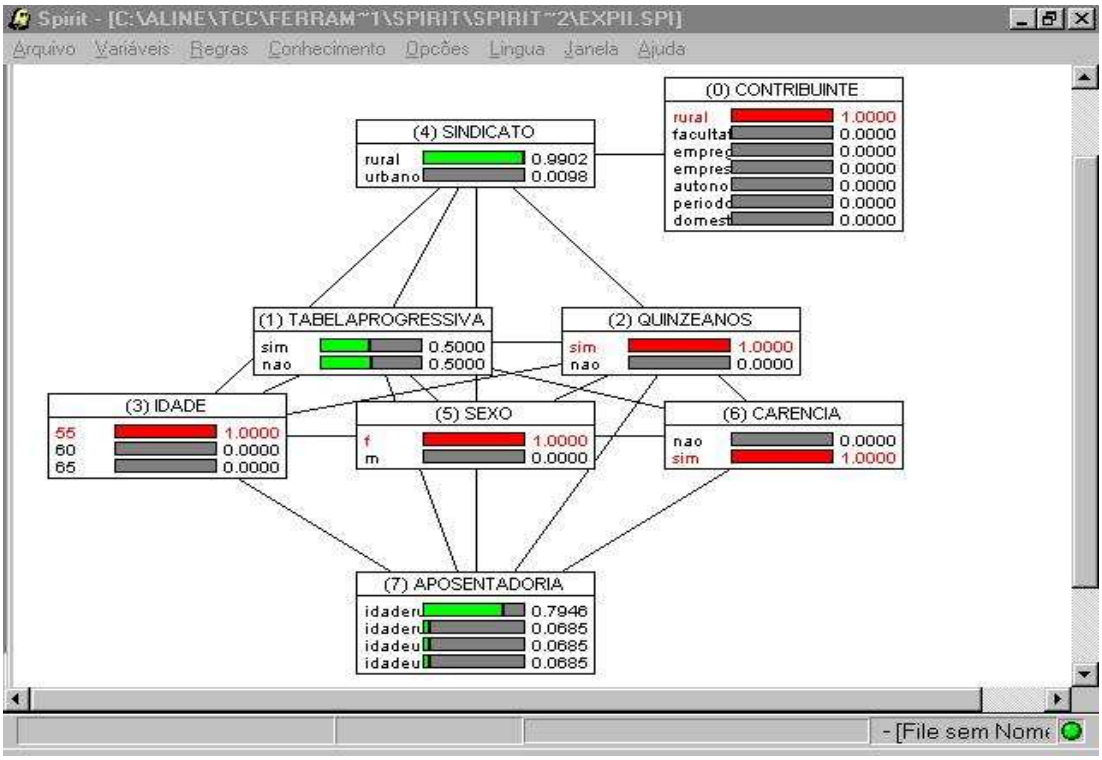
A ferramenta *Expert SINTA* obteve o resultado de aposentadoria idade rural mulher com 72 de fator de certeza, como mostra a tela de resultado apresentada na Figura 21:

Figura 21 - Resultado experimento 2 pelo *Expert SINTA*



Através das instanciações das variáveis especificadas para os testes efetuados na aplicação experimental 2 o *SPIRIT* tomou como resultado aposentadoria idade rural mulher com 79,46% de probabilidade (Figura 22).

Figura 22 - Resultado experimento 2 pelo *SPIRIT*



7 CONCLUSÕES

O desenvolvimento deste trabalho permitiu que fossem estudados aspectos pertinentes aos sistemas especialistas. Os principais benefícios da utilização de um sistema especialista são: velocidade na determinação de problemas, decisão fundamentada em uma base de conhecimento, segurança, pequeno número de pessoas para interagir com o sistema, estabilidade, dependência decrescente de pessoal específico, flexibilidade, integração de ferramentas e evitar a interpretação humana de regras operacionais.

O estudo das ferramentas para sistemas especialistas proporcionou o aumento das experiências com esses sistemas para solucionar problemas do mundo real. Através deste estudo foi possível a determinação dos parâmetros do comparativo: características e subcaracterísticas da ISO/IEC 9126, seguidamente foram escolhidos os pontos críticos que levam ao melhor desempenho dessas ferramentas: interface com o usuário, interface de desenvolvimento, interface com o sistema operacional, métodos de inferência; englobando métodos de raciocínio e tratamento da incerteza, e representação do conhecimento.

O *Arity Prolog* por ser uma linguagem de programação, mostrou-se bastante poderosa, dando maior flexibilidade na construção de sistemas especialistas, porém o projetista precisa estar munido de conhecimento relativo a linguagem para que possa implementar as rotinas que nas *shells* já estão prontas. Se um programa é considerado eficiente por resolver problemas em um tempo curto e utilizando poucos recursos, o Prolog certamente seria a melhor opção. Através das pesquisas realizadas sobre esta linguagem descobriu-se uma versão do Prolog, conhecida como LPA-Prolog que possui uma ferramenta para sistemas especialistas (FLEX), talvez esta opção se adaptasse melhor a este comparativo.

O *Expert SINTA* apresenta algumas vantagens com relação as outras ferramentas. Sua interface gráfica, com depuração integrada ao ambiente de desenvolvimento, aproveita plenamente os recursos fornecidos pelo sistema operacional, dispensando conhecimento técnico por parte do usuário. Fornece opções para tratamento da incerteza, recurso nem sempre disponível em outras ferramentas, ou nem sempre apresentado de forma transparente. A inclusão de textos explicativos referentes às conclusões obtidas em formato hipertexto é uma característica que propicia melhor aproveitamento das consultas realizadas. Contudo, a ferramenta, até a presente versão, não é capaz de identificar inconsistências lógicas. Por isso,

erros (como estouro de pilha, no caso em que a base entra em recursão infinita) podem vir a ocorrer em bases mal projetadas. Todavia, o *Expert SINTA* necessita de um estudo mais detalhado sobre funções de probabilidade, garantindo maior confiança à base de conhecimentos; uso de tendências mais modernas de Inteligência Artificial, como a Lógica Nebulosa; maior variedade de opções, como o uso do encadeamento para frente.

A ferramenta *SPIRIT* gera sistemas especialmente úteis para representar, analisar e interpretar relações complexas. É indicado para aplicações que visam diagnose (diagnósticos e previsões), conhecer modelos (relações causa/efeito) e classificação (reconhecimento de padrões). O *SPIRIT* não é um programa útil para fins de construção e simulação de sistemas em tempo real. A medida que é ampliado o número de atributos de cada variável, aumenta de forma geométrica a árvore de relacionamentos, o que implica num aumento no número de regras na mesma proporção (número de combinações possíveis entre todos os atributos). Portanto, a complexidade da base de regras e o volume de cálculos dependem destas definições. Neste sentido, o uso de variáveis ordinais e cardinais merecem um cuidado especial, pois podem conduzir facilmente a uma perda de controle sobre a base de regras, ou, elevar de forma dramática o tempo de processamento necessário para a aprendizagem dos fatos e regras informados.

O exposto acima é confirmado, a utilização de "*shells*" no desenvolvimento de sistemas especialistas reduz, significativamente, o trabalho do engenheiro de conhecimento, restando a ele a responsabilidade de, somente, fornecer os conhecimentos necessários ao sistema.

Com isto ficou claro que o aproveitamento adequado de uma ferramenta depende muito da forma como o modelo é estruturado, isto é, uma adequada definição das variáveis que compõe o sistema, seus atributos e sua estrutura de dependência e inter-relacionamentos.

A técnica de sistemas especialistas já conseguiu progressos notáveis e já possui centenas de aplicações úteis a nossa sociedade. Apesar disto, ainda não foi construído nenhum sistema cuja inteligência e capacidade de resolver problemas se aproxime da apresentada pelos especialistas.

7.2 EXTENSÕES

Como extensão deste comparativo sugere-se trabalhar com outras ferramentas para sistemas especialistas, por exemplo, ferramentas que utilizam lógica nebulosa, realizando aplicações experimentais mais consistentes.

Poder-se-ia trabalhar com mais opções de parâmetros para o comparativo.

ANEXO A: VARIÁVEIS EXPERIMENTO 1

Relação das variáveis envolvidas o experimento 1, seus respectivos tipos e valores:

Variável	Tipo	Valores
Ambiente	Multivalorada	Rural/Urbano
Ambientes freqüentados	Univalorada	Sim/Não
Andar cavalo	Univalorada	Sim/Não
Assistir filme	Univalorada	Sim/Não
Banho cachoeira	Univalorada	Sim/Não
Banho mar	Univalorada	Sim/Não
Beber	Univalorada	Sim/Não
Caminhada praia	Univalorada	Sim/Não
Comer fora	Univalorada	Sim/Não
Dançar pessoas	Univalorada	Sim/Não
Descansar campo	Univalorada	Sim/Não
Esporte	Univalorada	Sim/Não
Lugar	Multivalorada	Cidade/Fazenda/Litoral
Museu	Univalorada	Sim/Não
Programa	Multivalorada	Festas/Esportes/Aquáticos Visitas/Cachoeira/Cinema Acampar/Hotel Fazenda
Pesca Lagoa	Univalorada	Sim/Não
Teatro	Univalorada	Sim/Não
Telão	Univalorada	Sim/Não
Turismo rural	Univalorada	Sim/Não
Turismo urbano	Univalorada	Sim/Não

ANEXO B: VARIÁVEIS EXPERIMENTO 2

Relação das variáveis envolvidas o experimento 2, seus respectivos tipos e valores:

Variável	Tipo	Valores
Aposentadoria	Multivalorada	Idade Rural Mulher
		Idade Rural Homem
		Idade Urbana Mulher
		Idade Urbana Homem
Carência	Univalorada	Sim/Não
Contribuinte	Multivalorada	Rural
		Empregado
		Domestico
		Autônomo
		Facultativo
		Empresário
		Período Graça
Idade	Numérica	
Sexo	Univalorada	F/M
Sindicato	Univalorada	Urbano/Rural
Quinzeanos	Univalorada	Sim/Não
TabelaProgressiva	Univalorada	Sim/Não

REFERÊNCIAS BIBLIOGRÁFICAS

ARARIBÓIA, G. **Inteligência Artificial**: um curso prático. Rio de Janeiro: Livros técnicos e científicos, 1989. 282 p.

BATEZINI, Charles Tams. **Sistemas especialistas**. Passo Fundo 1999. Disponível em: <http://inf.upf.tche.br/home/trabalhos/estagio/99-2/charles/se.htm>. Acesso em: 07 mar. 2001.

BITTENCOURT, Guilherme. **Inteligência Artificial:** ferramentas e teorias. Florianópolis: Editora da UFSC, 1998. 362 p.

BRATKO, Ivan. **PROLOG**: Programming for artificial intelligence. Massachusetts: Addison Wesley Publishing Company, 1990. 597 p.

BRONZINO, J. D. **The Biomedical Engeneering Hand Book**. USA: CRCPRESS, IEEE Press, 1995.

CHAIBEN, Hamilton. **Inteligência Artificial na Educação**. Curitiba 1999. Disponível em : <http://www.cce.ufpr.br/~hamilton/iaed/iaed.htm >. Acesso em: 15 mar. 2001.

CRIPPA, Maurício. **Sistemas especialistas**: a engenharia do conhecimento aplicada às organizações. Florianópolis 1999. Disponível em: <http://n27.udesc.br/demo/trabalhos/alunos/mc/index.html>. Acesso em: 08 mar. 2001.

COSTA, Raimundo Machado. **Qualidade de produtos de software**. Rio de Janeiro 2001. Disponível em: <http://raimac.netdados.com.br/qualidade.htm>. Acesso em :10 abr. 2001.

FAVERO, Alexandre José. **Sistemas Especialistas**: tutorial. Maringá 1999. Disponível em: < http://www.din.uem.br/ia/especialistas/introdu.html>. Acesso em: 07 mar. 2001.

HECKMANN, Jacques Robert. **Sistematização das atuais técnicas de Inteligência Artificial e análise de sua aplicabilidade**. 1998. 69 f. Monografia (Especialização em Tecnologias de Desenvolvimento), Universidade Regional de Blumenau, Blumenau.

HEINZLE, Roberto. **Protótipo de uma ferramenta para criação de sistemas especialistas baseados em regras de produção**. 1995. 145 f. Tese (Mestrado em Engenharia de Produção), Universidade Federal de Santa Catarina, Florianópolis.

KANDEL, A. **Fuzzy Expert System**. Flórida USA: CRC Press, 1992.

KOPITTKE, B. H; WILHELM, P. P; LOPES, M. C. **Uma representação probabilística do conhecimento:** análise da interface SPIRIT. In: ENEGEP. Anais13 o Florianópolis UFSC, 1993.

LEVINE, R. I; DRANG, D. E; BARRY, E. **Inteligência artificial e sistemas especialistas:** aplicações e exemplos práticos. Tradução Maria Cláudia Santos Ribeiro Ratto. São Paulo: McGraw-Hill, 1988. 304 p.

LIA, Laboratório de Inteligência Artificial. **Expert SINTA:** uma ferramenta para criação de sistemas especialistas. Manual do usuário. Pernambuco, 1995. Disponível em: <http://www.lia.ufc.br>. Acesso em: 30 jan. 2001.

LIEBOWITZ, J. **WorldWide Expert Systems Activities and Trends**. Cognizant Communications Offices, USA, 1994.

LUCHTENBERG, Jonas. **Protótipo de um sistema especialista para área comercial utilizando a ferramenta SPIRIT.** 2000. 50 f. Trabalho de conclusão de curso (Bacharelado em Ciências da Computação), Universidade Regional de Blumenau, Blumenau.

RABUSKE, Renato Antonio. **Inteligência artificial**.. Florianópolis: Editora da UFSC, 1995. 240 p.

RICH, Elaine; KNIGHT, Kevin. **Inteligência artificial**. 2. ed. Tradução Maria Cláudia Santos Ribeiro Ratto. São Paulo: Makron Books, 1993. 772 p.

SALVATO, Gilberto J. **Sistemas especialistas:** método para a adoção em bibliotecas especializadas. 1997. 145 f. Tese (Mestrado em Administração de Políticas e Gestão Institucional), Universidade Federal de Santa Catarina, Florianópolis.

SOUZA, Marilda de. **Netica**. Blumenau, 2001. 6 transparências, color.,25 cm x 20 cm.

ULBRICHT, V. R; WAZLAWICK, R. S; SANTOS, N. **Representação do conhecimento de geometria descritiva através de redes semânticas**. In: Anais do I Congresso Internacional de Engenharia Gráfica nas Artes e no Desenho. Florianópolis, Setembro de 1996. p. 309-315.

STORCH, Mirian Mirdes. **Proposta de avaliação da qualidade de produtos de software utilizando a norma ISO/IEC 9126**. 2000. 82 f. Trabalho de conclusão de curso (Bacharelado em Ciências da Computação), Universidade Regional de Blumenau, Blumenau.

TEIVE, Raimundo C. Ghizoni. **Planejamento da expasão da transmissão de sistemas de energia elétrica utilizando sistemas especialistas**. 1997. Monografia (Pós Graduação em Engenharia de Produção), Universidade Federal de Santa Catarina.

UFSC. Universidade Federal de Santa Catarina. **Expert Systems**: in production engineering – EPS3416. Florianópolis, 2000. Apresenta textos sobre sistemas especialistas e engenharia do conhecimento. Disponível em: <http://www.eps.ufsc.br/~oscar/exp_sys/ing/>. Acesso em: 23 mar. 2001.

VICENTIN, Juliana M. **Protótipo de um sistema especialista para elaboração de roteiros turísticos personalizados.** 2000. 72 f. Trabalho de conclusão de curso (Bacharelado em Ciências da Computação), Universidade Regional de Blumenau, Blumenau.

WATERMAN, D. A. **A Guide of Expert Systems**. USA: Addison –Weslwy Publishing Company Inc, 1986.

WEISS, Sholom M.; KULIKOWSKI, Casimir. **Guia prático para projetar sistemas especialistas**. Rio de Janeiro: LTC – Livros Técnicos e Científicos S.A., 1988.