UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

ESPECIFICAÇÃO DE PRODUTOS DE IMAGENS DE SATÉLITES UTILIZANDO TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL

Dissertação submetida a Universidade Federal de Santa Catarina para obtenção do título de mestre em Engenharia de Produção.

José Carlos Pereira

Orientadora: Profª Édis Mafra Lapolli, Drª

Florianópolis, janeiro de 1995

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

ESPECIFICAÇÃO DE PRODUTOS DE IMAGENS DE SATÉLITES UTILIZANDO TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL

José Carlos Pereira

Esta dissertação foi julgada adequada para a obtenção do título de MESTRE EM ENGENHARIA DE PRODUÇÃO e aprovada em sua forma final pelo, Programa de Pós-Graduação em Engenharia da Produção.

PROF. OSMAR POSSAMAI, Dr.
Coordenador do Curso

BANCA EXAMINADORA:

PROF. ÉDIS MAFRA LAPOLLI, Drª
Orientadora

PROF. LIA CAETANO BASTOS, Drª

PROF. RICARDO MIRANDA BARCIA, Ph.D

À Luciane
À Thays Helena Pacheco Pereira

AGRADECIMENTOS

À professora Édis Mafra Lapolli que com sua experiência profissional e de vida, possibilitou o desenvolvimento deste trabalho, agradeço pelos ensinamentos e orientações transmitidas no decorrer do curso.

Agradeço a participação da especialista professora Lia Caetano Bastos e da especialista professora Ana Maria B. Franzoni pela colaboração prestada.

Ao professor Fernando A. O. Gauthier, pelas orientações prestadas.

Ao amigo Marco Aurélio pela sua incansável colaboração.

Agradeço ao professor Ricardo de Miranda Barcia, Ph.D. pelo apoio dado no transcorrer do curso.

À minha companheira Luciane Pacheco e à minha filha, Thays, agradeço pela compreensão da minha ausência, em momentos tão importantes de suas vidas.

À minha mãe pelo incentivo e ao meu pai José B. Pereira, um agradecimento especial pela sua dedicação aos filhos e ensinamentos deixados que possibilitou este estudo.

A meus camaradas que resistiram e resistem nestes tempos difíceis, com suas bandeiras e idéias.

A todos que de forma direta ou indiretamente ajudaram no transcorrer deste trabalho.

Resumo

O sensoriamento remoto da terra com satélites e aeronaves está intimamente relacionado e dependente da tecnologia em computação com processamento de dados para análise de imagens digitais desde 1960. O sensoriamento remoto ao longo da história vem abrindo novos campos de pesquisa e conduzindo a avanços no "pool" de capacidades do computador, como a incrementação de memória, velocidade e outros avanços que resultam na resolução de cores através de poderosos gráficos.

Imagens de satélite constituem uma ferramenta extremamente poderosa e de custo relativamente baixo, na solução de vários problemas de recursos naturais, como por exemplo, o monitoramento da cobertura vegetal para facilitar o controle sobre desmatamentos e queimadas; a identificação e avaliação de áreas cultivadas para auxiliar na previsão de safras e permitir o planejamento de áreas urbanas para análise e planejamentos macroeconômicos.

Embora existam disponíveis no Brasil desde 1975, ainda hoje desenvolvem-se metodologias de uso dessas imagens nas várias áreas de aplicação: Análise Ambiental, Florestas, Agricultura, Geologia e Cartografia. Apesar da falta de conhecimento técnico a respeito desta ferramenta dificultar sua maior disseminação, o uso de imagens de satélites tem aumentado ano a ano.

A utilização de técnicas de Inteligência Artificial, como Sistemas Especialistas para a especificação de produtos de imagens de Satélites e a utilização adequada dos produtos obtidos pelos diferentes sensores, faz-se necessário para auxiliar o usuário leigo em Sensoriamento Remoto na tarefa de especificar um produto imagem, dado um conjunto de recursos disponíveis, ou seja, saber qual o produto a ser utilizado para extrair a informação desejada da imagem gerada.

Abstract

The earth's remote sensing using satellites and aircrafts is closely related and dependent to the data processing technology to digital images analysis since 1960. The remote sensing all along history has been opening new research activities and leading to progress in the computer's capacities pool, such as memory increase, speed and improvement which result in colour resolution thru powerful graphics.

Satellite images are a extremely powerful tool with relatively low cost to solve several natural resources problems, for instance, the vegetation monitoring to facilitate control over jungle clearing and burned-over land; the identification and evalution of cultivated areas to help to harvest foresight and allow to the urban areas planning to macroeconomics planning and analysis.

Although available in Brazil 1975, yet today methodologies have been developed to use these images in several areas: Environment Analysis, Forests, Agriculture, Geology and Cartography. Despite the lack of tecnical knowledgement to this tool make its spreading difficult, satellites images use has been increasing year after year.

The use of Artificial Inteligence technics, such as Expert Systems to the specification of Satellites images products and the suitable use of products got by diferent sensors is necessary to help the Remote Sensing user in the task of specifying a image product, given a set of available resources, that means, one must know wich product to be used to extract desired information from the generated image.

SUMÁRIO

6. CONCLUSÕES E RECOMENDAÇÕES PARA FUTURAS PESQUISAS

ÍNDICE DE FIGURAS

Índice de Tabelas

1. INTRODUÇÃO

1.1 ORIGEM DO TRABALHO

O levantamento de recursos naturais da terra com satélites e aeronaves está intimamente relacionado e dependente da tecnologia em computação com processamento de dados para análise de imagens digitais desde 1960. Sistema de Informações Geográficas - GIS, dependem diretamente dos produtos obtidos por sensoriamento remoto para a revisão de mapas, análises ambientais, monitorização da temperatura, alterações na superfície da terra e numerosos outros estudos científicos sobre o planeta. O sensoriamento remoto ao longo da história vem abrindo novos campos de pesquisa e conduzindo a avanços no "pool" de capacidades do computador, como a incrementação de memória, velocidade e outros avanços que resultam na resolução de cores através de poderosos gráficos.

As aplicações de sistemas especialistas, embora possam ser classificadas em várias categorias, para Sensoriamento Remoto têm sido na área de interpretação, ou seja, na "inferência de descrições de situações a partir de dados de sensores".

Duas são as razões principais para concentração das aplicações na área de interpretação. A primeira é que só recentemente técnicas

de Inteligência Artificial têm sido empregadas em Sensoriamento Remoto e interpretação é um componente indispensável na maioria dos sistemas. A segunda razão é o potencial dos sistemas especialistas na tarefa de interpretação. Sistemas especialistas provêm mecanismos poderosos e flexíveis para a integração de informações de diversas fontes ([MCK87], [TAI86], [COU87]). É simples, por exemplo, incorporar dados históricos, geográficos e outros na tarefa de classificação, além da informação espectral e contextual da imagem.

Certamente a interpretação não esgota o espectro de aplicações de sistemas especialistas em Sensoriamento Remoto.

A aplicação de Sistemas Especialistas para a especificação de produtos de imagens de satélites e a utilização adequada dos produtos obtidos pelos diferentes sensores faz-se necessário para auxiliar o usuário leigo em Sensoriamento Remoto na tarefa de especificar um produto imagem, dado um conjunto de recursos disponíveis, ou seja, saber qual o produto a ser utilizado para extrair a informação desejada da imagem gerada.

1.2 OBJETIVOS DO TRABALHO

O objetivo geral deste trabalho é demonstrar a viabilidade da utilização de técnicas de Inteligência Artificial, especificamente Sistema

Especialista, na seleção de produtos de imagens de satélites e na utilização adequada de produtos obtidos pelos diferentes sensores.

Como objetivo específico tem-se a implementação do modelo computacional utilizando técnicas de inteligência artifical e programação orientada a objeto. Para tanto, foi utilizada a linguagem C++ e desenvolvido este aplicativo para o ambiente operacional Windows.

1.3 IMPORTÂNCIA DO TRABALHO

A importância deste trabalho localiza-se no fato de *Imagens de satélite constituirem-se numa ferramenta extremamente poderosa e de custo relativamente baixo, na solução de vários problemas de Recursos Naturais*, como por exemplo, o monitoramento da cobertura vegetal para facilitar o controle sobre desmatamentos e queimadas; a identificação e avaliação de áreas cultivadas para auxiliar na previsão de safras e permitir o planejamento de áreas urbanas para análise e planejamentos macroeconômicos.

Embora existam disponíveis no Brasil desde 1975, ainda hoje desenvolvem-se metodologias de uso dessas imagens nas várias áreas de aplicação: Análise Ambiental, Florestas, Agricultura, Geologia e Cartografia.

Apesar da falta de conhecimento técnico a respeito desta ferramenta dificultar sua maior disseminação, o uso de imagens de satélites tem aumentado ano a ano.

No momento, o desafio da aplicação de Sistemas Especialistas para acelerar a disseminação do uso das imagens obtidas, tanto a partir de satélites quanto de aeronaves, é de grande importância, pois pode-se produzir soluções tão eficazes ou até melhores do que se fossem produzidas por especialistas humanos.

1.4 ESTRUTURA DO TRABALHO

Este trabalho descreve os resultados obtidos na busca da realização dos objetivos acima descritos. Está dividido em cinco capítulos.

Neste capítulo, a origem, os objetivos, a importância e a estrutura do trabalho são descritos. Sua finalidade é definir o tema da pesquisa e a estrutura do trabalho.

O segundo capítulo apresenta alguns conceitos de Sensoriamento Remoto e uma breve análise dos sistemas LANDSAT e SPOT e sobre os produtos de imagem obtidos por sensoriamento remoto.

O terceiro capítulo apresenta uma abordagem teórica da Inteligência Artificial - o desenvolvimento e aplicação dos Sistemas Especialistas.

Dedica-se o quarto capítulo a especificação do modelo proposto - o SISTEMA ESPECIALISTA - procura-se demonstrar a viabilidade da aplicação de sistemas especialistas em Sensoriamento Remoto e a ferramenta utilizada para o desenvolvimento do sistema computacional para especificação de produtos de imagens de satélite.

No capítulo cinco é apresentado a validação do modelo.

O capítulo sexto é dedicado as conclusões e recomendações para futuras pesquisas.

2. SENSORIAMENTO REMOTO

2.1 INTRODUÇÃO

O uso de sensoriamento remoto data de 1858, quando TOURNACHON usou uma câmara num balão, para fotografar e estudar partes da cidade de Paris [FOR83].

Dessa época até a segunda guerra mundial, as técnicas de Sensoriamento Remoto permaneceram estagnadas. A guerra trouxe um aumento no conhecimento do potencial do Sensoriamento Remoto.

Atualmente, as técnicas de Sensoriamento Remoto estão sendo usadas para identificar, mapear e cadastrar os recursos naturais da terra e controlar sua utilização pelo homem, além dos problemas decorrentes da atividade humana, como: ocupação urbana, poluição, etc.

As observações feitas por Sensoriamento Remoto podem ser reunidas em três domínios:

- **Espacial -** tamanho, forma e distribuição;
- **Temporal -** variações no decorrer do tempo;
- **Físico -** trocas energéticas por radiação eletromagnéticas ou alterações em campos de força.

A grande maioria das atividades em sensoriamento remoto concentra-se na observação de fenômenos ligados à radiação eletromagnética, portanto, o domínio seria antes espectral do que simplesmente físico.

O domínio temporal em pequena escala, para algumas técnicas, é explorado através de medidas de fase ou tempo de resposta. Entretanto, para um contexto mais longo, são necessárias observações repetidas ao longo do tempo. Isto foi facilitado com o lançamento de satélites de recursos naturais como o LANDSAT e o SPOT [AMA78].

No domínio espacial, os esforços têm sido dirigidos para as técnicas de tratamento digital de imagens.

2.2 CONCEITOS

Sensoriamento Remoto pode ser definido como a aplicação de dispositivos que, colocados em aeronaves ou satélites, permitem obter informações sobre objetos ou fenômenos na superfície da terra [AMA78].

[STE81] *conceituam Sensoriamento Remoto como um conjunto de atividades cujo objetivo reside na caracterização das propriedades de alvos naturais, através da detecção, registro e análise do*

fluxo de energia radiante, por eles refletido ou emitido. Os mesmos autores dizem que a metodologia do Sensoriamento Remoto pode ser dividida em duas fases: a de Aquisição e a de Análise. A primeira está relacionada com os processos de detecção e registro da informação e a segunda, constituída do tratamento e da interpretação dos dados obtidos.

[NOV89] *define Sensoriamento Remoto como sendo a utilização conjunta de modernos sensores, equipamentos de transmissão de dados, aeronaves, espaçonaves etc., com o objetivo de estudar o ambiente terrestre, através do registro e análise das interações entre a radiação eletromagnética e as substâncias componentes do planeta terra em suas mais diversas manifestações.*

Existem duas categorias de *sensores remotos, quanto à fonte de energia*: **os ativos e os passivos. Os sensores ativos** são os que possuem fonte própria de energia e os **passivos** são os que necessitam de fontes externas de energia.

Quanto ao **tipo de produto - em função do tipo de transformação sofrida pela radiação detectada,** os sensores remotos podem classificar-se em **não-imageadores e imageadores** .

Sistemas sensores não-imageadores, ou seja, que não fornecem uma imagem da superfície sensoriada, seriam os radiômetros cuja

saída é em forma de dígitos ou gráficos. São essênciais para a aquisição minuciosas sobre o comportamento espectral dos objetos da superfície terrestre. Como sensores não-imageadores temos os radiômetros de banda e os espectrorradiômetros.

Sistemas Imageadores fornecem como resultado uma imagem da superfície observada. Os sensores imageadores fornecem informação sobre a variação espacial da resposta espectral da superfície observada.

Os **sistemas sensores imageadores** podem ser ainda classificados em função do processo utilizado na formação da imagem. Os sistemas de quadro ("framing systems") adquirem a imagem da cena em sua totalidade num mesmo instante. No sistema de varredura ("scanning systems"), a cada imagem da cena é formada pela aquisição seqüencial de "imagens elementares do terreno ou "elementos de resolução", também chamados de "pixels".

Os diferentes sistemas são caracterizados por sua **resolução**, ou seja, resolução é uma medida da habilidade que um sistema sensor possui de distinguir entre respostas que são semelhantes espectralmente ou próximos espacialmente.

A resolução pode ser classificada em resolução espacial e resolução espectral. **A resolução espacial** mede a menor separação angular ou linear entre dois objetos. Por exemplo, quando dizemos que um sistema possui uma resolução de 30 metros, isto significa que objetos distanciados entre sis menos de 30 metros não serão, em geral, discriminados pelo sistema.

A **resolução espectral** é uma medida da largura das faixas espectrais e da sensibilidade do sistema sensor em distinguir entre dois níveis de intensidade do sinal de retorno (resolução radiométrica). Por exemplo , um sistema sensor que opera na faixa de 0,4 e 0,5 μm tem uma resolução espectral maior que um sensor que opera na faixa de 0,4 a 0,6μm. Este sensor será capaz de registrar pequenas variações no comportamento espectral em regiões mais estreitas do espectro eletromagnético. A resolução radiométrica depende da sensibilidade dos detectores do sistema sensor. A redução da faixa espectral implica numa redução da intensidade da resposta que chega ao detector.

Os **sensores imageadores** classificam-se em sensores **fotográficos e não-fotográficos.**

Um dos ***produtos*** obtidos do **sensor fotográfico** são as chamadas *fotografias aéreas*. Essas podem ser adquiridas a partir de *filmes*: pancromático, colorido, infravermelho preto e branco, infravermelho colorido.

O filme pancromático tem a capacidade de registrar, em variações de tons de cinza, a maioria das cores do espectro visível.

O filme infravermelho preto e branco em conjunto com o filtro vermelho escuro, é projetado para registrar somente os raios de luz infravermelhos refletidos.

Os filmes coloridos são disponíveis em dois tipos : positivos e negativos. Os filmes positivos, após processados, produzem transparências que representam a cena com a mesma aparência que terá ao ser observada sob a luz solar. Os filmes negativos permitem a reprodução de cópias positivas em papel.

Os filmes infravermelhos coloridos reproduzem os objetos da natureza com cores diferentes das naturais.

Os sistemas de imageamento **eletro-ópticos (não fotográficos)** diferem dos sistemas fotográficos, porque os dados são registrados em forma de sinal elétrico, o que possibilita sua transmissão à distância.

Todo sistema de imageamento tem dois componentes básicos : o sistema óptico e o detector. O sistema óptico tem função de focalizar a energia proveniente da cena sobre o detector.

O sinal elétrico produzido pelo detector é então processado e cada nível de radiância é alocado a um conjunto de coordenadas no espaço.

Apesar destas semelhanças, os sensores de imageamento eletro-ópticos podem ser classificados em três grandes grupos quanto ao processo de formação de imagens: sensores de quadro ("frame"), sensores de varredura eletrônica e sensores de varredura mecânica (TABELA 2.1).

TABELA 2.1- TIPOS BÁSICOS DE SISTEMAS DE IMAGEAMENTO ELETRO-ÓPTICO

CARACTERISTICAS	FRAME	VARREDURA ELETRÔNICA	VARREDURA MECÂNICA
POSSIBILIDADE DE VISÃO ESTEREOSCÓPIA	BOA	BOA	ADEQUADA
SUCEPTIBILIDADE À MOVIMENTAÇÃO DA PLATAFORMA	PEQUENA	MÉDIA	GRANDE
POSSIBILIDADE DE USO MULTIESPECTRAL	PEQUENA	A SER MELHORADA	MUITO BOA
CAPACIDADE DE OPERAÇÃO NO IR	LIMITADA	ELEVADA	MÉDIA
PRECISÃO GEOMÉTRICA	PEQUENA PARA VARREDURA POR FEIXE DE ELÉTRONS	LIMITADA PELA TECNOLOGIA DE CONSTRUÇÃO DE MATRIZES DE DETETORES	ALTA PRECISÃO
NÚMERO DE ELEMENTOS DE RESOLUÇÃO POR CENA	LIMITADO PELO TAMANHO DA MATRIZ E PELO SISTEMA ÓPTICO	LIMITADO PELO TAMANHO DO ARRAY E PELO SISTEMA ÓPTICO	ILIMITADO

FONTE: [NOV89]

As imagens dos satélites incluem-se nos produtos gerados pelos sensores não-fotográficos.

Os sistemas de microondas utilizados em sensoriamento remoto de recursos naturais são sensores ativos e produzem imagens do

terreno. O sistema de imageamento mais comum é o dos radares de visada lateral (side-looking a irbone radars), que permite o imagemento contínuo do terreno.

Os sistemas de imageamento de radar evoluíram a partir dos PPI (Plan Position Indicartors) que serviam para auxiliar a navegação aérea. Esse sistema não produzia uma imagem do terreno, mas indicava a posição de certos objetos no plano de deslocamento da aeronave.

Assim é que o termo RADAR é derivado da expressão Radio Detection and Ranging (Detecção e Medida de Distância por Rádio).

2.3 SISTEMA LANDSAT

O primeiro satélite de Sensoriamento Remoto de recursos terrestres não-tripulado foi o Earth Resources Technology Satellite 1 (ERST-1) lançado em 1972 que, em 1975, passou a chamar LANDSAT 1. Esse primerio satélite foi construído a partir de uma modificação do satélite metereológico NUMBUS e inicialmente levou a bordo 2 tipos de sensores: um sistema de varredura multiespectral, conhecido como MSS (Multiespectral Scanner Subsystem) e um sistema de varredura constituído por três câmeras de televisão

(Return Beam Vidicon), conhecido como RBV. Atualmente o LANDSAT 5 mantém em operação dois sensores: o Thematic Mapper (TM) e o MSS.

O Sistema LANDSAT opera desde 1972, sendo que os dados do sistema LANDSAT são recebidos no Brasil desde 1973, que conta com toda uma infraestrutura para sua recepção, processamento e distribuição, através do Instituto Nacional de Pesquisas Espaciais.

O Sistema LANDSAT, desenvolvido pela National Aeronautics and Space Admninistration (NASA), compõe-se, até o presente, de uma série de 5 satélites (TABELA 2.2), com o objetivo de permitir a aquisição de dados espaciais, espectrais e temporais, sobre a superfície da terra de forma global, sinóptica e repetitiva.

TABELA 2.2 - SATÉLITES DA SÉRIE LANDSAT

NÚMERO DO SATÉLITE	DATA DE LANÇAMENTO	TÉRMINO DE OPERAÇÃO
LANDSAT 1	23/07/72	05/01/78
LANDSAT 2	22/01/75	27/07/83
LANDSAT3	05/03/78	07/09/83
LANDSAT4	16/07/82	-
LANDSAT5	01/03/84	-

FONTE: [FRA93]

Os três primeiros satélites da série LANDSAT estiveram inseridos numa órbita circular, quase polar, síncrona com o sol, a uma altitude aproximada de 920 Km.

Durante seu período de operação, os satélites realizavam uma órbita completa em torno da Terra a cada 103 minutos e 27 segundos, de modo a recobrir 14 faixas da superfície terrestre por dia.

A configuração da órbita dos satélites 1, 2 e 3 foi estabelecida de tal modo que a cada 18 dias eles passavam sobre a mesma região da superfície terrestre.

O angulo de inclinação da órbita do satélite em relação ao plano do Equador (99°11') fazia com que ele descrevesse uma órbita quase polar em torno da terra, garantindo o imageamento entre as latitudes de 81^0N a 81^0S. Esta inclinação também garantia que a órbita fosse "síncrona com o sol", permitindo que os dados fossem coletados sob condições local similares.

Outra característica importante é que o plano de órbita desloca-se em torno da Terra à mesma velocidade do deslocamento da Terra em relação ao Sol. Desta forma , cada vez que o satélite cruza o Equador em direção ao Sul (órbita descendente), ele o faz durante o mesmo horário local, durante todo o ano. O horário médio de passagem dos satélites pelo Equador é 9:30 horas, variando conforme a longitude.

As principais características do subsistema LANDSAT 1, 2 e 3 encontram-se na **TABELA 2.3.**

TABELA 2.3 PRINCIPAIS CARACTERISTICAS DO SUBSISTEMA MSS LANDSAT 1,2 e 3

Características	Bandas	Desempenho
Resposta Espectral	4 5 6 7 8	0,5-0,6μm (verde) 0,6-0,7μm(vermelho) 0,7-0,8μm(infra-vermelho próximo) 0,8-1.1μm(infra-vermelho próximo) 10,4-12,6μm(infra-vermelho termal)
Tipo de Detetor	4,5,6, 7 8	Tubo foto multiplicada díodo de silício(Si) Telureto de Mercúrio-Cádmio (Hg-Cd-Te)
Configuração de Detetores	4,5,6,7 8	6 detetores por banda 2 detetores por banda
IFOV (linear)	4,5,6,7 8	79 m 237 m
Nível de Radiância	4 5 6 7	24,8 W m^{-2} sr^{-1} 20,0 W m^{-2} sr^{-1} 17,6 W m^{-2} sr^{-1} 46,0 W m^{-2} sr^{-1}

FONTE: [NOV89]

Os parâmetros orbitais dos Satélites 1, 2, 3 são apresentados na TABELA 2.4.

TABELA 2.4 - PARÂMETROS ORBITAIS DOS SATÉLITES 1, 2, 3 DA SÉRIE LANDSAT

PARÂMETROS ORBITAIS	LANDSAT 1,2,3
Altitude (Km)	920
Inclinação (graus)	99,4
Período (minutos)	103
Horário de passagem pelo Equador	9h15
Duração do ciclo de cobertura (dias)	18

FONTE: [NOV89]

O LANDSAT 4 e 5 representam a ponte entre as antigas e as novas gerações de sistemas orbitais de coleta de dados [FRE83], [GOR83].

A órbita dos satélites LANDSAT 4 e 5 é semelhante à dos satélites anteriores. É repetitiva, circular, sol-síncrona e quase-polar. Sua altitude é inferior à dos primeiros satélites, estando posicionada à altura de 705 Km em relação à superfície terrestre no Equador.

Os parâmetros orbitais dos Satélites 4 e 5 são representados na TABELA 2.5.

TABELA 2.5 - PARÂMETROS ORBITAIS DOS SATÉLITES 4 e 5 DA SÉRIE LANDSAT

PARÂMETROS ORBITAIS	LANDSAT 4,5
Altitude (Km)	705
Inclinação (graus)	98,20
Período (minutos)	98,20
Horário de passagem pelo Equador	9,45
Duração do ciclo de cobertura (dias)	16

FONTE: [NOV89]

O LANDSAT 1 e o LANDSAT 2 conduziram a bordo dois tipos de sensores com resolução espacial de 80 m, o subsistema MSS (Multispectral Scanner Subsystem), com imageamento do terreno por varredura de linhas. operando em 4 faixas espectrais, e o subsistema de câmera de televisão RBV (Return Beam Vidicon) com imageamento instantâneo de toda a cena, operando com três faixas do espectro [CUR85].

Para o LANDSAT 3, o sensor RBV foi modificado, passando a operar com duas câmeras em apenas uma faixa do espectro e com resolução espacial de 40m, e ao sensor MSS foi acrescentada uma faixa espectral para operar na região do infravermelho termal, com resolução espacial de 240m.

Nos satélites LANDSAT 4 e 5, o sensor RBV e MSS foram substituídos pelo sensor TM (Thematic Mapper) de varredura multiespectral, com resolução espacial de 40m para as bandas 1, 2, 3 ,4 ,5 e 7 e de 120m para a banda 6 (termal).

A resolução espacial é a menor distância entre dois objetos que um sensor pode distinguir. A resolução espectral de um sistema sensor é determinada pelas faixas do espectro eletromagnético dos canais utilizados. Assim, uma alta resolução espectral é obtida por estreitas amplitudes de bandas, as quais agregadamente servem para fornecer uma assinatura espectral mais precisa dos objetos.

O sistema LANDSAT, como qualquer outro sistema de Sensoriamento Remoto orbital, compõe-se de 2 partes principais: o SUBSISTEMA SATÉLITE e o SUBSISTEMA ESTAÇÃO TERRESTRE, também conhecido como o segmento solo.

O subsistema Satélite tem a função básica de adquirir os dados, enquanto o subsistema Estação Terrestre tem a função de processar os dados e torná-los utilizáveis por especialistas em extração de informações de interesse.

As características dos sensores a bordo dos satélites da série LANDSAT estão apresentadas na TABELA 2.6.

TABELA 2.6 - CARACTERÍSTICAS DOS SENSORES A BORDO DOS SATÉLITES DA SÉRIE LANDSAT

	SISTEMA DE VARREDURA MULTIESPECTRAL			CÂMERAS DE TELEVISÃO	
SENSOR	MSS	MSS	TM	3 CÂMERAS RBV	2 CÂMERAS RBV
NUMERO DE CANAIS	4	5	7	3	1
RESOLUÇÃO ESPACIAL	80m	80m 240m	30m 120m	80m	40m
LANDSAT 1	X	-	-	*	-
LANDSAT 2	X	-	-	*	-
LANDSAT 3	-	X	-	-	X
LANDSAT 4	X	-	X	-	-
LANDSAT 5	X	-	X	-	-

LEGENDA:

X -> OPERANDO A BORDO DO SATÉLITE
*** -> DESATIVADO LOGO APÓS O LANÇAMENTO**
- -> AUSENTE

FONTE: [NOV89]

Na TABELA 2.7 , são apresentadas as principais aplicações potenciais das bandas TM do LANSAT 5.

TABELA 2.7 - PRINCIPAIS APLICAÇÕES POTENCIAIS DAS BANDAS TM DO LANDSAT 5

BANDA	INTERVALO ESPECTRAL	APLICAÇÕES POTENCIAIS
1	0.45 - 0,52	Estudos batimétricos em regiões litorâneas de aguá limpa até profundidade de 20 a 40m; mapeamentos de superfície de água e análise de materiais em suspensão. É denominada de banda azul. Diferenciação solo/vegetação. Sensitividade à concentração de carotenos e clorofila. Alguma possibilidade de identificação de Fe^{+3} e Mn^{+3}.
2	0,52 - 0,60	Mapeamento de vegetação sadia pela reflectância verde cujo pico se situa em 0,55μm.
3	0,63 - 0,69	Banda de absorção da clorofila; significativa na diferenciação de espécies vegetais. Distinção de variações de densidade urbanas. Estudo do uso do solo.
4	0,76 - 0,90	Estudos de volume da biomassa e delineação de corpos d'água.
5	1,55 - 1,75	Stresses de vegetação por desequilibrio de água na cobertura foliar. expectativa na identificação de mineralizações superficiais, sobretudo com os dados dda divisão da banda 5 pela banda 1. Estudo de estrutura urbana.
6	10,4 - 12,5	Propriedades termais de solo, rocha, vegetação e água. Estudos de contraste térmico entre litologias de rochas silicáticas. Estudos micro-climáticos.
7	2,08 - 2,35	Esta é considerada estritamente uma banda geológica selecionada para identificar minerais com íons hidroxilas. Potencialmente favorável à discriminação de produtos de alteração hidrotermal. Neste intervalo estão presentes algumas importantes bandas de absorção de rochas carbonáticas.

Fonte: Adaptado de INPE (ATUS - Atendimento ao Usuário) e Pereira Kurkdjian e Foresti (1989) in [FRAN93]

Os produtos resultantes do processamento eletrônico e fotográfico dos dados recebidos pelas estações terrenas são basicamente de dois tipos: fitas compatíveis com computador e produtos fotográficos.

As fitas compatíveis com computador podem ser obtidas com as características especificadas na TABELA 2.8

TABELA 2.8 - **CARACTERÍSTICAS DOS DADOS DIGITAIS MSS E TM PRODUZIDOS NO LABORATÓRIO DE CACHOEIRA PAULISTA**

Características	MSS	TM
Correções do sistema	Radiométrica	Radiométrica Correção do movimento do espelho (reamostragem)
Formato	Bandas intercaladas em pares de pixels (BIP2) Bandas intercaladas por linhas (BIL)	Bandas sequenciais (BSQ) Bandas intercaladas por linhas (BIL)
Densidade	BIP2 = 800 bpi 1600 bpi BIL 1600 bpi	1600 dpi
Canais	4, 5, 6, 7	1, 2, 3, 4, 5, 6, 7, Imagem completa 1 até 7 canais Quadrantes 3 canais 7 canais

FONTE: [NOV89]

2.4 SISTEMA SPOT

O sistema SPOT é um programa espacial francês semelhante ao programa LANDSAT. Compreende basicamente um satélite,

que pode ser modificado para acomodar diferentes "cargas úteis"; sistemas sensores, concebidos para certos objetivos e sistemas terrestres de aquisição, processamento e disseminação de dados.

O satélite do sistema SPOT foi lançado em 22/02/86 e leva a bordo dois sensores de alta resolução (HRV - Haute Resolução Visible). Esse sistema foi concebido pelo Centre National de Études Spatiales e construído em consórcio com indústrias francesas.

A altitude da órbita do satélite SPOT é de 832 Km. É uma órbita polar, sícrona com o Sol, mantendo uma inclinação de 98°7' em relação ao plano equatorial.

Os sensores HRV operam nos modos: multiespectral (XS), em três faixas do espectro eletromagnético (TABELA 2.9) e no pancromático (PAN), com resoluções espaciais, respectivamente de 20m e 10m. Os dados do satélite SPOT são encontrados em canais individuais ou composições coloridas formadas por duas bandas no visível e uma no infravermelho.

Do ponto de vista temático, a resolução espacial dos sistemas sensores tem sido um fator limitante, tendo em vista a dimensão de alvos de interesse. Dessa forma, alvos de dimensões inferiores ao poder de resolução espacial desses sistemas sensores podem não ser discriminados, tanto nas imagens fotográficas como nas digitais. Por outro lado, as imagens digitais

(no formato de fitas compatíveis com computador - CCT) podem ser submetidas a processamentos ou tratamentos, através de sistemas computadorizados, que podem melhorar o poder de discriminação entre os alvos contidos na cena imageada [PIN91].

TABELA 2.9 - FAIXAS ESPECTRAIS DO MODO MULTIESPECTRAL DO SENSOR HRV A BORDO DO SPOT

CANAIS	FAIXAS ESPECTRAIS(μm)
1	0,50 - 0,59
2	0,61 - 0,68
3	0,79 - 0,89

FONTE: [FRA93]

2.5 ELEMENTOS DE ANÁLISE DE IMAGENS

As imagens, qualquer que seja o seu processo de formação, representam o registro de energia proveniente dos objetos da superfície.

Essas imagens podem ser de diferentes resoluções e escalas. Independente disto, elas se caracterizam por apresentar os elementos básicos da interpretação, o que permite a extração de informações do terreno.

As imagens coloridas permitem que os objetos sejam diferenciados através de variações na cor. Em imagens preto e branco essas diferenciações são feitas através de variações de tonalidade ou de nível de cinza

Na TABELA 2.10 podemos encontrar as características de cor e tonalidade de objetos, tal como são observados em imagens de diferentes tipos. Entretanto, a cor e a tonalidade não são únicos atributos de um objeto e nem todos os objetos se distinguem apenas pela cor. Um campo de cultivo e uma pastagem podem apresentar uma mesma cor, quando observados em imagens obtidas por sistemas sensores. Nessas circunstâncias,outros aspectos deverão ser considerados na identificação dos objetos.[NOV89]

Além disso, a cor ou a tonalidade dos objetos (como representativa de seu comportamento espectral) estão sujeitas a variações em função das condições ambientais e da aquisição dos dados.

Um corpo de água, normalmente por atenuar a energia incidente , tem um sinal de retorno pequeno, o que o faz apresentar tonalidades escuras em fotografias aéreas pancromáticas. Entretanto, em função da geometria, sol, sensor, alvo, os corpos d´águas podem ter, neste tipo de produto, tonalidade que varia do branco (reflexão total) ao preto.

Outro aspecto importante na identificação dos objetos através de suas cores é o conhecimento do produto utilizado e o comportamento espectral dos objetos de interesse. Numa imagem correspondente ao infravermelho, uma floresta terá tonalidade cinza claro, porque nesta região há elevada reflexão de energia dos sol. Numa imagem na região do vermelho, a floresta apresentara tonalidade escura (preta), devido a elevada absorção de energia pela clorofila. A cor ou tonalidade com que

aparece um objeto é função das propriedades do objeto, mas também das características do produto sob análise.[NOV89]

As imagens não oferecem, porém, informações apenas sobre cor e/ou tonalidade. Pode-se, também, observar nas imagens a dimensão dos objetos. Entretanto, na avaliação das dimensões do objeto como elemento de identificação , deve-se levar em conta a escala da imagem que está sendo analisada.

TABELA 2.10 - EXEMPLOS DE DIFERENCIAÇÃO DE OBJETOS PELA COR

TIPO DE IMAGEM	OBJETO	COR/TONALIDADE
Composição infra-vermelho Falsa cor		vermelho azul verde
Canal MSS 7	Árvores Águas Solo	Cinza claro Preto Cinza escuro

FONTE : [NOV89]

Em imagens orbitais, a textura permite diferenciar, em alguns casos, áreas de reflorestamento de áreas florestais. As áreas de reflorestamento, por possuírem uma vegetação homogênea no tocante ao espaçamento, altura dos indivíduos e idade, apresentam, geralmente, textura mais lisa que as áreas florestais naturais.

O contexto ou a associação entre objetos é outro elemento útil para sua identificação em imagens de Sensoriamento Remoto. O indivíduo poderá ter dificuldades, por exemplo, de identificar a mancha urbanizada em uma imagem de satélite, se ela estiver localizada numa região de intenso uso agrícola. Entretanto, se ele procurar identificar elementos associados à presença de cidades, terá sua tarefa facilitada. Em geral, as cidades têm seu acesso através de estradas; dessa maneira, ela, ao contrário das parcelas agrícolas, estará associada a um sistema de linhas convergentes, formadas pelas ligações intermunicipais.

2.6 PROCESSAMENTO DIGITAL DE IMAGENS

Durante a última década, tem-se notado um crescente interesse no uso de dados de imagem digital, o que tem levado ao desenvolvimento de sistemas para processar esses dados.

Processamento Digital de Imagens é o conjunto de procedimentos relativos à manipulação e análise de imagens através do computador. Esses procedimentos englobam a entrada de dados digitais, o realce, a estatística e a geração de saídas, que podem ser imagens em tons cinzas ou coloridas.[LAP94]

O principal objetivo do uso das técnicas de processamento digital de imagens é realçá-las, melhorando a sua qualidade e seu aspecto. Dessa forma, obtêm-se melhores subsídios para a sua interpretação e, ainda, podem ser submetidas a outros processamentos.

O uso de técnicas de processamento digital de imagens aumenta consideravelmente a capacidade do analista de extrair informações da superfície terrestre, a partir de dados de sensoriamento remoto.[LAP94]

2.7 CONCLUSÃO

Observa-se que o Sensoriamento Remoto da terra com satélites e aeronaves está intimamente relacionado e dependente da tecnologia em computação com processamento de dados.

A utilização de computadores combinado com os modernos sensores, equipamentos de transmissão de dados, aeronaves, espaçonaves etc., com o objetivo de estudar o ambiente terrestre, através do registro e análise das interações entre a radiação eletromagnética e as substâncias componentes do planeta terra em suas mais diversas manifestações fazem das imagens de satélite uma ferramenta extremamente poderosa e de custo relativamente baixo.

Atualmente, as técnicas de Sensoriamento Remoto estão sendo usadas para identificar, mapear e cadastrar os recursos naturais da terra e controlar sua utilização pelo homem, além dos problemas decorrentes da atividade humana, como: ocupação urbana, poluição, etc.

Para utilizá-las na solução de vários problemas de recursos naturais desenvolve-se metodologias de uso dessas imagens nas várias áreas de aplicação. Em todos os domínios de observação feitas por Sensoriamento Remoto desenvolve-se esforços para aprimorar e criar novas técnicas.

3. INTELIGÊNCIA ARTIFICIAL

3.1 INTRODUÇÃO

Ao terminar a Segunda Guerra Mundial, grupos de cientistas ingleses e norte-americanos trabalhavam para desenvolver o que hoje se chama de computador. Cada grupo pretendia criar uma máquina eletrônica, que pudesse ser conduzida por um programa armazenado de instruções e fosse feita para executar cálculos numéricos complexos. O principal cientista britânico, Alan Turing, argumentava que tal máquina para fins gerais, uma vez desenvolvida, teria muitos usos diferentes. Refletindo o seu conhecimento das realizações da lógica formal nos anos antes da guerra, Turing argumentava que as instruções fundamentais dadas a uma máquina assim teriam de basear-se em operadores lógicos, tais como "e" "ou" e "não". Poder-se-iam então usar tais operadores muito genéricos, para montar os operadores numéricos mais especializados, necessários para operações aritméticas. Além disso, programas baseados em operadores lógicos, seriam capazes de manipular qualquer tipo de material simbólico com que se quisesse trabalhar, incluindo afirmações em linguagem ordinária. [HAR88]

Os cientistas norte-americanos, que eram mais práticos, sabiam que a construção da máquina ia sair muito cara. Além disso,

acreditavam que não poderiam construir muitas. E, como tinham confiança de que construiriam uma máquina que só faria cálculos aritméticos, decidiram contra o uso de operadores lógicos e escolheram, ao invés, usar os operadores numéricos, tais como "+", "-" e ">". Essa decisão, que logo os ingleses seguiram, também , resultou em grandes computadores que, em essência, são máquinas muito rápidas de calcular. Apesar da grande proliferação de computadores desde 1946, esta decisão pareceu razoável à maioria dos que estavam envolvidos com computadores.

Não obstante o fato de os computadores terem sido construídos como processadores numéricos, um pequeno grupo de cientistas dos computadores continuou a explorar a capacidade de os computadores manipularem símbolos não numéricos. Simultaneamente, psicólogos interessados na resolução de problemas pelo homem buscavam desenvolver programas de computador que simulassem o comportamento humano. Durante anos, indivíduos interessados tanto no processamento simbólico como na resolução de problemas pelo homem formaram essa subdivisão da informática que se chama Inteligência Artificial (IA). *Os pesquisadores em IA preocupam-se em desenvolver sistemas de computador que produzam resultados que normalmente se associariam à inteligência humana.*

Há cerca de 15 anos, um número de corporações achou que alguns dos resultados da pesquisa dos laboratórios de IA comprovariam ser úteis na área comercial. Várias empresas constituíram grupos de IA para desenvolver aplicações práticas. Em geral, estes esforços fracassaram, porque

os programas de IA custavam muito caro, eram *demasiado lentos* e não produziram bastantes resultados práticos. Os programas de IA eram, simplesmente, *demasiado complexos* para rodarem nos computadores que existiam. Apesar disso, os pesquisadores de IA continuaram a trabalhar nas Universidades e fizeram progressos teóricos. Enquanto isso, o *desenvolvimento da tecnologia da microeletrôncia* criou uma nova geração de computadores mais rápidos, mais potentes e relativamente baratos. Hoje a IA emergiu de novo dos laboratórios. Os equipamentos existentes combinados com os significativos progressos teóricos em IA, resultaram em uma tecnologia cuja hora chegou.

A *IA* pode-se subdividir em três áreas de pesquisa relativamente independentes. Um grupo de pesquisadores em IA preocupa-se sobretudo em desenvolver programas de computador que leiam, falem ou entendam a linguagem que as pessoas usam em sua conversa diária. Esse tipo de programação é designado comumente *como processamento de linguagem natural.* Outro grupo de cientistas em IA ocupa-se com o *desenvolvimento de robos inteligentes.* Interessam-se especialmente em como desenvolver programas visuais e táteis, que permitam aos robos observar as contínuas alterações que sucedem quando circulam em um ambiente. Um terceiro ramo da pesquisa em IA ocupa-se em desenvolver programas que usem o conhecimento simbólico para *simular o comportamento dos especialistas humanos.*

3.2 CONCEITOS FUNDAMENTAIS

O paradigma dominante em IA é a construção, à imagem e semelhança do homem, de máquinas pensantes e capazes de execução de tarefas diversas. A ciência artificial é a maneira pela qual um sistema de tratamento de informação - homem/computador - pode representar a informação coletada no mundo exterior e utilizá-la para elaborar suas próprias ações [SIM69]. Na sua visão a IA busca compreender como os seres humanos tomam suas decisões com a finalidade de programar os computadores da melhor forma possível.

A eficácia de um programa que se pretende inteligente está diretamente relacionada à quantidade de informações que possui sobre o assunto tratado. Por isso acredita que os sistemas de computadores poderão "produzir informação", isto é, associar dados já armazenados no equipamento a informações novas, com "elaboração" de conceitos, hipóteses, diagnósticos e terapias inéditos. [FEI68]

Partindo do princípio de que só é possível acumular um saber exaustivo em campos restritos, aposta-se que, com uma matéria bem circunscrita, "podemos produzir soluções informáticas tão eficazes, senão mais do que se fossem produzidas por especialistas humanos" [FEIN68]. Para ele, os sistemas especialistas refletem os processos de raciocínio de seus programadores, mas também a lógica matemática, a teoria das probabilidades

e o "raciocínio humano normal" que se consegue armazenar. Na sua visão, um técnico, para criar um software eficiente, precisa ser um observador atento e um programador minucioso, mas não é necessário que seja tão inteligente quanto o especialista.

3.3 SISTEMAS ESPECIALISTAS

Os Sistemas Especialistas podem ser definidos como:

1- São programas que imitam o comportamento de especialistas humanos. Usam informações que o usuário fornece para emitir uma opinião sobre um certo assunto. Assim um sistema especialista faz perguntas até que possa identificar algo que o leve a respostas. [SCH89]

2- São programas ou conjuntos de programas que usam essas técnicas para resolver problemas normalmente resolvidos por especialistas (peritos), ou seja, obter uma solução de alta qualidade num campo bem delimitado do qual ele tem profundo conhecimento prático e teórico, e, mais ainda, fazê-lo de forma eficiente. São essas as características que diferem um perito de um novato, e devem estar incorporadas da forma mais completa possível em um sistema que se proponha imitá-lo.[GEN88]

3- Um sistema especialista pode ser definido, mais ido, concretamente, como uma estrutura de programação capaz de armazenar e utilizar, com menos restrições das presentes em um programa clássico, algum tipo de conhecimento sobre uma determinada área. Tendo como características específicas armazenar e utilizar informações, capacidade de aprendizagem e o poder de inferência. [NOV86].

Sistema Especialista é *um programa inteligente de computador que usa conhecimento e procedimento inferenciais, para resolver problemas que são bastante difíceis de forma a requererem, para sua solução, muita perícia humana. O conhecimento necessário para atuar a esse nível, mais os procedimentos inferenciais empregados, pode considerar-se um modelo da perícia aos melhores profissionais do ramo.*

O conhecimento de um sistema especialista consiste em fatos e heurísticas. Os fatos constituem um corpo de informação que é largamente compartido, publicamente disponível e geralmente aceito pelos especialistas em um campo. As heurísticas são em sua maioria privadas, regras pouco discutidas de bom discernimento (regras do raciocínio plausível, regra da boa conjectura), que caracterizam a tomada de decisão a nível de especialista na área. O nível de desempenho de um sistema é função principalmente do tamanho e da qualidade do banco de conhecimento que possui. [FEI81]

Um sistema especialista pode ser definido, mais concretamente, como uma estrutura de programação capaz de armazenar e utilizar, com menos restrições das presentes em um programa clássico, algum tipo de conhecimento sobre uma determinada área. O sistema especialista é uma estrutura de programação, ou melhor, é um programa. Esta estrutura de programação é capaz, basicamente, de duas coisas: 1- pode armazenar informação, como se fosse um banco de dados, sobre o tema considerado; 2 - tem a faculdade de utilizar essa informação para obter alguns resultados que não existiam previamente no computador. Isso é realizado mediante uma técnica que realmente diferência os sistemas especialistas dos programas clássicos.

Outra características específicas dos sistemas especialistas são encontradas, além das já explicadas de uso geral (armazenar e utilizar) na capacidade de aprendizagem e poder de inferência. Os sistemas especialistas são muito mais eficazes que os programas clássicos porque podem ativar o bloco de inferência que é necessário em cada caso sem necessidade de ter que analisar todos os precedentes.

3.4 ORGANIZAÇÃO DE UM SISTEMA ESPECIALISTA

Pesquisadores de Inteligência Artificial têm observado a importância do conhecimento específico para a resolução de problemas de dificuldade significativa. Tem sido observado que o fato de uma pessoa ter um elevado QI não a torna um especialista. Na realidade, o que torna uma pessoa especialista é a quantidade de conhecimento especializado que possui.

Sistemas Baseados em Conhecimento são empregados para resolver problemas que costumam requerer a inteligência humana [Hay84]. A característica geral de tais sistemas é a representação explícita do conhecimento, que é armazenado de modo flexível e modular na *"base de conhecimento"*. Quando o sistema necessita raciocinar sobre problemas específicos do domínio em questão, a base de conhecimento é acessada de modo a obter as informações necessárias à realização da inferência. O mecanismo de controle capaz de navegar pela base de conhecimento em busca de uma solução para o problema é conhecido como *"máquina de inferência"*.

Os Sistemas baseados em Conhecimento devem ser capazes de seguir linhas de raciocínio que pareçam mais apropriadas aos dados disponíveis para a solução automática dos problemas. Além disso, devem ser capazes de incorporar novas informações à base de conhecimento e de explicar seu comportamento quando for requisitado.

O processo de construção de Sistemas Baseados em Conhecimento é frequentemente chamado de Engenharia de Conhecimento. A

tarefa de um engenheiro de conhecimento é prover o sistema com uma quantidade de conhecimento compatível à possuída pelo especialista.

A FIGURA 3.1 mostra a estrutura básica de um sistema baseado em conhecimento através de um módulo de interface.

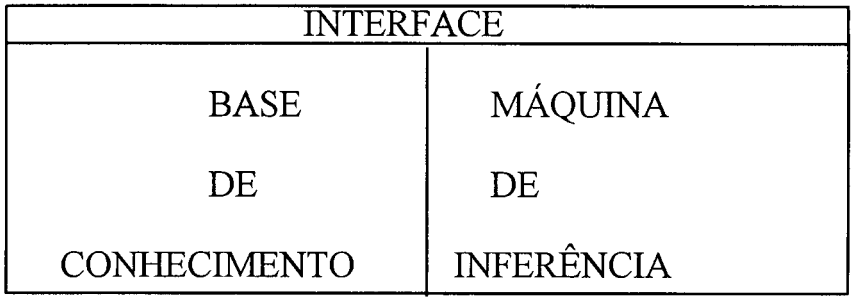

FIGURA 3.1 - Estrutura Básica de um Sistema Baseado em Conhecimento

A *estrutura de um sistema especialista* é bastante diferente de um sistema convencional, pois aqui, não existe um algoritmo explícito. O conhecimento é armazenado (normalmente sob a forma de regras do tipo "se...então...") em um módulo separado, chamado *base de conhecimento*. Já o processamento dessas regras é executado por um módulo que analisa a base de conhecimento usando técnicas de procura em espaço de estados, chamado *máquina (ou motor) de inferência*. Existe também um terceiro módulo, uma *interface* que realiza a interação com o usuário em linguagem natural (ou tão natural quanto possível), e responde a perguntas como: "Por que esta pergunta está sendo feita?", "O que significa essa pergunta que o sistema está me fazendo - não a entendi" e outras.

É importante ressaltar que, construída a máquina de inferência, o projetista pode acrescentar (ou retirar) uma ou mais regras à base de conhecimento, praticamente sem ter que se preocupar com o seu processamento, e sem ter que mudar um programa ou algoritmo. Isso se traduz em benefícios como melhor organização e clarificação do conhecimento embutido no sistema, facilidade de manutenção e construção, etc.

3.4.1 BASE DE CONHECIMENTO

A base de conhecimento dá as características de funcionamento do sistema. Esse terá o conhecimento, do que for colocado na sua base de conhecimento, isto é, se ela for projetada para receber informações de uma determinada ciência, o sistema será especialista nessa ciência. É o local onde são armazenados fatos e regras. Um novo fato pode modificar todo um processo de inferência de acordo com as regras existentes sobre ele, que estão sendo aplicadas e também sobre os novos fatos gerados pela avaliação dessas regras.

A base de conhecimento é um banco de dados que armazena informações específicas e regras sobre um certo assunto.

Para implementação concreta da base de conhecimento que nos referimos é necessário contar com uma forma adequada de representação do conhecimento.

3.4.2 MECANISMO DE INFERÊNCIA

É o elemento de um sistema especialista capaz de buscar as regras necessárias a serem avaliadas, ordenadas de uma maneira lógica e, a partir daí, ir direcionando o processo de inferência. Ou seja é a parte do sistema que usa a informação que voce fornece para encontrar um objeto correspondente.

Existem duas grandes categorias de máquina de inferência: *determinística e probabilística.* Para entender a diferença entre essas categorias poderemos dizer que a *determinística* afirma com certeza, não tendo dúvida sobre a resposta e a *probabilística* dá uma resposta provável ou com alguma possibilidade de sucesso.

Além das duas categorias de incerteza e certeza, existem *tres maneiras básicas para construir uma máquina de inferência*: encadeamento direto, encadeamento reverso e valor da regra. As diferenças entre esses métodos relacionam-se à maneira como a máquina busca o objetivo procurado.

*a- **Método de encadeamento direto (Foward-Chaining)** - Esse método é muitas vezes chamado, dirigido por dados, porque a máquina de inferência usa informação que o usuário fornece para se movimentar por meio de uma rede de E e OU lógicos até encontrar um ponto terminal que é o objeto. Se a máquina de inferência não pode encontrar um objeto usando a informação existente, então ele requisita mais. Dessa forma uma máquina de inferência de encadeamento direto começa com algumas informações e, então, tenta encontrar um objeto que contenha as informações.*

O método de encadeamento direto faz do processo de derivação da enorme porção de informação da base de conhecimento algo mais fácil, porque constrói uma árvore. Um sistema de encadeamento direto típico encontra todos os objetos possíveis que se casam com os atributos.

b- *Método do encadeamento reverso (Backward-Chaining)* - Esse método é o contrário do encadeamento direto. Uma máquina de inferência de encadeamento reverso começa com uma hipótese (um objeto) e pede informação para confirmar ou refutar. Este encadeamento é algumas vezes chamado dirigido por objeto, porque o sistema especialista começa com o objeto e tenta confirmá-lo.

c- *Método do valor da regra (Rule-Value)* - Uma máquina de inferência tipo valor da regra é teoricamente superior tanto ao encadeamento direto como ao encadeamento reverso, porque pede informações que tem grande importância de acordo com o estado atual do sistema. Ela é na verdade um melhoramento da máquina de encadeamento reverso. A teoria geral de

operação é que os pedidos do sistema, assim como sua próxima informação, removerão a maior incerteza do sistema.

Existem duas tarefas para realizar o mecanismo de inferência:

1-Inferência: Deve ser aplicada a inferência ao executar uma das regras. Estas são executadas em função do tipo de inferência que possua o sistema especialista. Para realizar as inferências é necessário ter armazenado determinadas regras de controle que ensinam o sistema como levá-las a cabo.

2-Controle: O controle se ocupa de toda a gestão do sistema; tanto da forma com que são escolhidas as regras como da resolução de conflitos ou da eliminação de loops de inferência. Sendo necessários dispor de *métodos* para decidir onde começar a analisar as regras que são armazenadas no banco de conhecimentos bem como, resolver os conflitos entre as regras e os loops de indução. Além disso é necessário que o sistema trabalhe otimamente gestionando espaço de busca de uma forma adequada.

3.5 REPRESENTAÇÃO E AQUISIÇÃO DO CONHECIMENTO

A aquisição do conhecimento necessário e a estruturação para o funcionamento adequado de um sistema especialista são as tarefas

primordiais na construção de tais sistemas, que podem ser definidas da seguinte forma: *Aquisição de conhecimento é a transferência e transformação da habilidade ou perícia para resolver problemas contida em algumas fontes de conhecimento para um programa.*

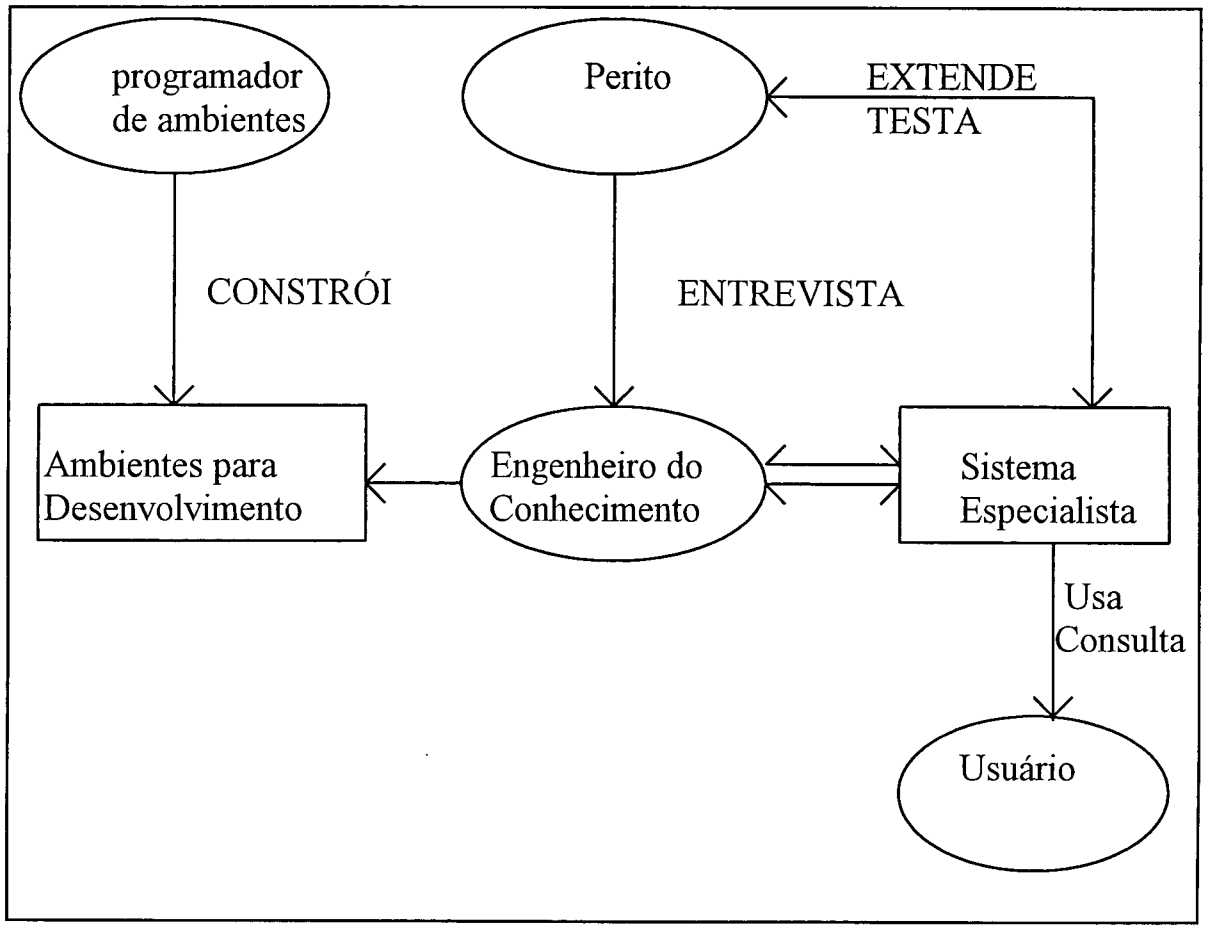

FIGURA 3.2 -Pessoal envolvido na engenharia de conhecimento.

FONTE: [GEN88]

O conhecimento pode-se originar de diversas fontes, individuais ou combinadas, como livros, registros, banco de dados, estudo de casos, dados empíricos ou experiências pessoais. Todavia, atualmente, a fonte dominante é o *conhecimento de peritos* no domínio e o engenheiro usualmente

obtém o conhecimento especializado através de interação direta com o perito. Tal interação consiste de um série de entrevistas, intensa e sistemática que usualmente se estende por vários meses. Durante este período, o engenheiro de conhecimento apresenta ao perito problemas reais para solução, do tipo que deseja que o sistema especialista projetado manuseie.

Assim, esta série de entrevistas para em uma fase inicial, onde o diálogo do engenheiro com a fonte é confuso e pouco estruturado. Nesta fase, o engenheiro estará se adaptando à terminologia, definindo os problemas específicos do domínio do conhecimento, características principais e hierarquia de subproblemas, buscando descobrir seus princípios gerais. A seguir, vem a fase intermediária onde os participantes já criaram uma linguagem comum e iniciaram a sistematização dos conhecimentos relevantes do domínio, representando-os de maneira formalizada. Aqui, o engenheiro procurará tornar explícito, conceitos e relações, estabelecer a estrutura do conhecimento e quais as estratégias que devem ser usadas. Na fase conclusiva o engenheiro codifica no programa os conhecimentos adquiridos na fonte, empregando a representação escolhida para construção de um protótipo, que, após testado e avaliado, chega à fase final, onde o programa já funciona como sistema autônomo em alguma aplicação de interesse.

O engenheiro de conhecimento deve trabalhar com o perito no contexto de resolver problemas particulares. Deve perguntar diretamente ao especialista qual regra ou método foi utilizado para a solução de um particular

problema do domínio. Os peritos normalmente têm grande dificuldade em exprimir tais regras.

Desta maneira a aquisição do conhecimento é o processo de extração e formalização do conhecimento de um perito para o uso em um sistema especialista.

Existe um consenso que todo programa de computador contém conhecimento sobre o problema que está resolvendo. No caso de programas convencionais, o conhecimento reside nos algoritmos empregados pelo programa e no procedimento de decisão que determina qual algoritmo deve ser utilizado numa determinada circunstância. Entretanto, uma característica desses programas é que o conhecimento que possuem não está representado explicitamente e não pode ser facilmente expandido e manipulado.

Esse fato contrasta com os trabalhos realizados em Inteligência Artificial, onde é absolutamente necessário um estudo profundo do conhecimento para que os sistemas de computação possam modelar o comportamento humano e tornarem-se máquinas mais eficientes. Neste estudo é fundamental saber como o conhecimento é adquirido, representado, estendido e usado pelo sistema. A *área de IA responsável por este estudo é a Representação do Conhecimento*.

O *primeiro* ponto chave no problema de representação do conhecimento é encontrar uma linguagem de representação que seja capaz de descrever o domínio de aplicação. O *segundo* ponto refere-se à capacidade de acessar os fatos implícitos na base de conhecimento *através da máquina de inferência* . Finalmente, o *terceiro* componente do problema de representação de conhecimento concerne à habilidade de capturar na base de conhecimento o que representa o entendimento do domínio pelo sistema.

É importante ressaltar que existem várias técnicas de representação de conhecimento que podem ser utilizadas na construção de sistemas especialistas. Essas técnicas podem ser utilizadas sozinhas ou em conjunto com outras, sendo que cada uma fornece ao sistema certas características próprias (tais como: maior eficiência, melhor compreensão ou maior facilidade de manutenção). Neste sentido, uma constatação interessante é que a forma de representação de conhecimento de um sistema especialista depende mais do tipo de problema a ser resolvido que do domínio específico de aplicação [Cha86].

As técnicas mais usuais de representação do conhecimento são sistemas de produção, a lógica de primeira ordem, as redes semânticas e os quadros ("frames").

Mostraremos algumas das características que devem ser analisadas ao se escolher uma técnica de representação de conhecimento:

1) *Expressividade*: Refere-se à capacidade da técnica de exprimir o conhecimento que se deseja representar.

2) *Eficiência de Raciocínio:* Refere-se à capacidade de realizar inferências num tempo aceitável de resposta.

3) *Primitivas:* Quais são as primitivas adotadas nesta representação, se existem?

4) *Meta-representação:* Como é estruturado o conhecimento na base e como é representado o conhecimento sobre esta estrutura?

5) *Incompletude*: Qual a capacidade da representação em lidar com bases de conhecimento imcompletas? Como realizar inferências com informações incompletas sobre o objeto e como revisar estas inferências quando do surgimento de novas informações?

6) *Conhecimento geral:* Como se lidar com atitudes como crenças, desejos e intenções?

Existem diferentes maneiras de representar o conhecimento, entre elas:

1- Representação baseada em regras - Nos sistemas de produção o conhecimento se expressa em regras. As regras se compõem de premissas, e ações que devem realizar-se em relação as premissas. Ou de condições e conclusões que podem ser extraídas quando são cumpridas as condições. Através de estruturas Se...então.. Tanto as premissas como as conclusões podem ser pares, atributos - valor, ou *triplas : objeto - atributos - valor*. Essa

forma de representação tem sua origem em conceitos utilizados na teoria de autômatos, e nas gramáticas formais, e em linguagem de programação.

2- Representação baseada em objetos estruturados - Existe uma grande variedade de sistema de representação do conhecimento baseado em objetos estruturados. Provavelmente a característica comum fundamental é que nestes sistemas o fator conhecimento está centrado nos objetos, a posição desse fator corresponde aos sistemas baseados em regras. Onde o conhecimento está estruturado com um conjunto de eixos , cada um referindo-se a um ou mais objetos. Nestes sistemas se definem os objetos como pertencentes a classes, das quais herdam propriedades e procedimentos. Um objeto pode em alguns sistemas pertencer simultaneamente a várias classes distintas , e herdam propriedades de todas elas. Se constituem as hierarquias que permitem uma grande economia cognitiva. Esse tipo de representação favorece a estruturação do conhecimento e permite aproveitar as estruturas que realmente existem no domínio do problema.

3- Representação baseada em lógica - A lógica é um formalismo adequado para representar declarativamente o conhecimento . Quando se utiliza a lógica como formalismo de representação na resolução do problema se encara como demonstração de teoremas. Se utiliza em geral o princípio da resolução de um algoritmo unificado. Utilizando o formalismo lógico torna-se fácil formular sistemas baseados em regras, com encadeamento para frente ou para trás , e objetos estruturados.

Existem várias técnicas que são envolvidas no campo da representação do conhecimento , e as mais importantes são as seguintes:

1- Árvore : É a maneira mais eficiente para representar o conhecimento em sistemas especialista de encadeamento reverso. À medida que o sistema especialista progride através da árvore, poda grandes partes da árvore e encontra a parte de conhecimento apropriada rapidamente. Árvores são hierárquicas por natureza e, assim , podem armazenar somente conhecimento hierárquico. Porém , esta não é uma restrição muito grande, porque grande parte do conhecimento cai nessa categoria.

2- Listas : Listas sâo consideradas o método tradicional em IA de representação do conhecimento. A primeira razão para isso é que LISP, a primeira linguagem de IA, foi desenvolvida para gerenciar eficientemente listas. O que as faz particularmente atrativas para representação de conhecimento é que elas são bem fáceis para trabalhar usando C. Embora listas possam ser processadas, apenas, seqüencialmente, elas são importantes porque podem ser usadas para representar todo tipo de conhecimento. É essa grande flexibilidade que as torna importantes em IA. Além disso, usando vários esquemas de indexação, você pode tornar listas quase tão eficiente quanto árvores.

3- Redes : A rede de conhecimento é baseada em duas condições. A primeira é que o conhecimento na rede é representado por nós num grafo não-hierarquico, na rede todos os nós tem a mesma importância e qualquer dos nós pode ser usado como ponto de partida. A segunda condição é que os nós

sejam arranjados para que tipos similares de conhecimento sejam agrupados próximos uns dos outros. Uma vantagem desta técnica é que ela pode fácilmente e eficientemente lidar com o conhecimento hierarquico e não-hierarquico.

A Aquisição de Conhecimento pode ser definida como o processo de transformação e transferência do conhecimento sobre um determinado problema para o sistema especialista.

O conhecimento de um sistema especialista pode ser originário de várias fontes: livros, banco de dados, casos, dados experimentais e experiência pessoal. Contudo, a fonte dominante de conhecimento costuma ser o próprio especialista humano. Nesse contexto, o processo de aquisição de conhecimento se refere à eliciação e codificação do conhecimento de um especialista durante a criação da base de conhecimento.

Desta forma, a construção de uma base de conhecimento é resultado do esforço cooperativo entre o engenheiro de conhecimento e o especialista. Codificar o conhecimento do especialista é uma tarefa laboriosa e sujeita a erros. São necessários grandes quantidades de tempo e recursos humanos para construir e refinar bases de conhecimento. Por esta razão, a aquisição de conhecimento costuma ser considerada o *"gargalo"* no desenvolvimento de sistemas especialistas.

A dificuldade de aquisição de conhecimento é usualmente associada a problemas de comunicação [BBB83]. Engenheiros de conhecimento não têm familiaridade com a área de aplicação do sistema, eventualmente, não são capazes de dimensionar a significância do que lhe é relatado pelos especialistas. Ao mesmo tempo, um especialista, freqüentemente, não tem idéia de como bases de conhecimento são construídas e de quais aspectos de sua especialidade necessitam ser codificados.

Outro aspecto que dificulta a aquisição de conhecimento é o fato dos próprios especialistas não terem plena consciência de como classificam os objetos e resolvem os problemas na sua área [MV88]. Embora especialistas possam facilmente descrever as classes nas quais os objetos são encontrados em sua profissão e delinear as características que formam a base de sua decisão, quando são submetidos a questões sobre atividades que realizam rotineiramente, freqüentemente oferecem respostas que não refletem completamente o seu verdadeiro comportamento. Isto decorre do fato de que o conhecimento do especialista se apresenta numa forma complicada, o que torna parte deste conhecimento praticamente inacessível.

Portanto, os muitos ciclos de construção, teste e reconstrnção de um sistema não podem ser atribuídos somente a falta de entendimento entre engenheiros de conhecimento e especialistas. Muitas das dificuldades residem na própria falta de habilidade dos especialistas em identificar como resolvem os problemas em suas profissões.

Várias técnicas costumam ser empregadas no processo de aquisição do conhecimento de um especialista ([HOF87],[GDD88]), dentre as quais são apresentadas:

- Entrevistas: São sem dúvida o modo mais tradicional de aquisição do conhecimento, tendo originado a maioria dos sistemas especialistas. Trata-se de um processo de interação direta envolvendo perguntas, respostas e discussões. As entrevistas são ditas *informais* quando empregadas com o propósito de familiarização com os conceitos, restrições e jargões do domínio do problema. Quando são aplicadas sistematicamente e determinam a forma como o especialista discorre sobre o domínio, as entrevistas são ditas *estruturadas.*

- Análise de Protocolo: É caracterizada por uma observação "*in loco*" do comportamento do especialista na solução de problemas reais. A análise de protocolo pode ser *retrospectiva* - quando o especialista é filmado resolvendo o problema e, posteriormente, é requisitado a explicar o seu comportamento - ou pode ser *concorrente* - quando pede-se aos especialista que "pense em voz alta" para que esta verbalização possa ser gravada.

- Análise de Casos e Tarefas: São baseados na análise de casos e tarefas associadas com o domínio do problema. Quando consiste na observação e interação com o especialista quando da execução de tarefas familiares ou rotineiras, a análise é chamada *tarefas familiares*. Quando requisita-se ao especialista resolver tarefas familiares sem informações usualmente

disponíveis, a análise é por *tarefas com informações limitadas.* Quando consiste em participar da análise de vários casos típicos junto ao especialista, ela é dita por *casos típicos.* Já quando os casos a serem analisados são incomuns e difíceis para o próprio especialista, a análise é por *casos atípicos.*

- Metodologia Kads: É baseada em primitivas epistmológicas que descrevem, num nível abstrato, os elementos fundamentais invariavelmente encontrados em problemas a ser solucionados por sistemas especialistas. Cinco tipos são empregadas:

  - *Objetos:* São dados e conclusões do processo de inferência. Ex.: sinais, sintomas, diagnósticos.

  - *Fontes de Conhecimento:* São partes de outros. Ex.: regras, quadros.

  - *Modelos:* São estruturas que representam um conjunto de relações complexas e que podem ser úteis na previsão de novos fatos e na explicação de um fenômeno. Ex.: modelos causais.

  - *Estruturas:* A mais comum é a hierárquica, a qual é baseada em relações do tipo sub superclasse, parte/todo, geral/específica.

  - *Estratégias:* São planos que ativam fontes de conhecimento durante a solução do problema. Ex.: encadeamento para frente, encadeamento para trás, propagação de relações.

A partir destas primitivas são desenvolvidos modelos de interpretação, os quais são modelos genéricos do processo de resolução do

problema para uma classe de tarefas. O *papel do engenheiro de conhecimento* consiste em preencher os vazios entre os dados adquiridos e o formalismo de representação de conhecimento com a ajuda destes modelos.

Obter o conhecimento necessário para a criação de um sistema especialista não é uma tarefa simples. Em alguns temas, o sistema pode aprender através da experiência, mas normalmente o especialista e o programador do sistema devem trabalhar unidos para conseguir condensar o saber em algumas certas regras lógicas. Atualmente se está trabalhando sobre certos programas que recebem a sabedoria do perito mediante sessões de ensino. O computador vai perguntando, analisando as respostas e incorporando-as ao banco em forma de regras lógicas.

Finalizando, cabe ressaltar que estas técnicas, além de serem freqüentemente complementares entre si, podem ser bastante auxiliadas pela aplicação de ferramentas interativas para o desenvolvimento e teste das bases de conhecimento. Outra alternativa mais elaborada para aquisição de conhecimento é prover ao sistema a capacidade de aprendizado automático.

3.6 SISTEMAS ESPECIALISTAS E SENSORIAMENTO REMOTO

O sensoriamento remoto da terra com satélites e aeronaves está intimamente relacionado e dependente da tecnologia em computação com

processamento de dados para análise de imagens digitais desde 1960. O Sistema de Informações Geográficas - GIS utilizam imagens obtidas por sensoriamento remoto para a revisão de mapas, análises ambientais, monitorização da temperatura, alterações na superfície da terra e numerosos outros estudos científicos sobre o planeta. Como as imagens obtidas por sensoriamento remoto são utilizados pelos cientistas do mundo todo, sendo que alguns não tem um profundo conhecimento das técnicas de processamento de imagens, existe a necessidade da aplicação de inteligência artificial para solucionar os problemas gerais da análise e interpretação de imagens.

Com o uso dos sistemas especialistas se pode incorporar ao processo de classificação o conhecimento do interprete. Dessa maneira, esses sistemas poderão servir para auxiliar outros interpretes na "tomada de decisão".

Imagens de satélite constituem uma ferramenta extremamente poderosa e de custo relativamente baixo, na solução de vários problemas de Recursos Naturais que utilizam imagens obtidas por Sensoriamento Remoto, como por exemplo, o monitoramento da cobertura vegetal para facilitar o controle sobre desmatamentos e queimadas; a identificação e avaliação de áreas cultivadas para auxiliar na previsão de safras e permitir o planejamento de áreas urbanas para análise e planejamento macroeconômicos.

Embora existam disponíveis no Brasil desde 1975, ainda hoje desenvolvem-se metodologias de uso dessas imagens nas várias áreas de

aplicação: Analise Ambiental, Florestas, Agricultura, Geologia e Cartografia. Apesar da falta de conhecimento técnico a respeito desta ferramenta dificultar sua maior disseminação, o uso de imagens de satélites tem aumentado ano a ano.

As aplicações de sistemas especialistas embora possam ser classificadas em várias categorias, a maioria dos sistemas especialistas para Sensoriamento Remoto têm sido na área de interpretação, ou seja, na "inferência de descrições de situações a partir de dados de sensores" [HAY83] . Entre essas aplicações podem-se citar a atualização de mapas florestais a partir de imagens LANDSAT [GOLD85], classificação de uso do solo urbano [WHA87], análise de fotografias aéreas ([MAT87],[NIC87],[MCK85], análise de consistência entre mapa e imagem [GOO87] e cartografia ([YEE87],[WAN87]).

Duas são as razões principais para concentração das aplicações na área de interpretação. A primeira é que só recentemente técnicas de Inteligência Artificial têm sido empregadas em Sensoriamento Remoto e interpretação é um componente indispensável na maioria dos sistemas. A segunda razão é o potencial dos sistemas especialistas na tarefa de interpretação. Sistemas especialistas provêm mecanismos poderosos e flexíveis para a integração de informações de diversas fontes ([MCK87], [TAI86], [COU87]). É simples, por exemplo, incorporar dados históricos, geográficos e outros na tarefa de classificação, além da informação espectral e contextual da imagem.

Certamente a interpretação não esgota o espectro de aplicações de sistemas especialistas em Sensoriamento Remoto. A aplicação de Sistema Especialista para a especificação de produtos de imagens e a utilização adequada dos produtos obtidos pelos diferentes sensores faz-se necessário para auxiliar o usuário leigo em Sensoriamento Remoto na tarefa de especificar um produto imagem, dado um conjunto de recursos disponíveis, ou seja, saber qual o produto a ser utilizado para extrair a informação desejada da imagem gerada.

3.7 CONCLUSÃO

A tentativa de construir máquinas que apresentem as capacidades que associamos com a inteligência humana, não é nada recente: já no século XVIII foram construídos os primeiros autômatos que tentavam emular aspectos parciais. A causa do relançamento da IA, é proveniente da *falta de algoritmos matemáticos* capazes de enfrentar situações aparentemente simples, tais como o reconhecimento da linguagem, os problemas de diagnósticos ou o reconhecimento visual de objetos .

Hoje em dia os programas considerados mais avançados são os mais sistemas baseados no conhecimento de "especialistas", embora suas capacidades sejam ainda limitadas em suas aplicações, *restringindo-se a temas muito concretos,* tais como os sistemas de diagnósticos de enfermidades

infecciosas ou de prospecções geológicas, que estão sendo aplicados com muito êxito por inúmeros hospitais e pelas companhias petrolíferas. *O resultado da união do conhecimento de um especialista com os computadores mais potentes e rápidos, está criando um novo ambiente que mudará completamente as estratégias de resolução de problemas antigos e além disso permitirá enfrentar os outros intratáveis na atualidade.* Essa busca ocorre pela tentativa de incorporar um conhecimento, uma heurística, que permita ao programa eliminar aqueles caminhos cuja possibilidade de ser corretos é mais baixa, facilitando a resolução destas situações e impulsionando o esclarecimento de numerosas questões.

Um sistema tradicional é estabelecido para resolver um problema bem especificado que pode ser colocado de forma algoritimizada, isto é, pode-se determinar uma série de etapas finais para se resolver esse problema, em tempo considerado aceitável para obtenção da resposta. Esses sistemas já são concebidos para apresentar uma solução correta. Por outro lado um sistema especialista tem seu funcionamento básico *apoiado em heurística*, por isso, ele é a solução para problemas nos quais não é possível fazer um algoritmo ou , se feito, irá obrigar a um processamento muito demorado para obtenção da solução. *Um processamento heurístico conduz a solução de maneira rápida, porém pode inclusive não conduzir a solução alguma. Um sistema especialista processa conhecimento e não processa dados. O conhecimento é armazenado em uma base de conhecimento e os dados são ajustados contra ela. O processamento é feito em cima deste conhecimento e não existe o processamento de dados.*

É difícil tentar descrever qual será o futuro de um ramo da ciência que é caracterizado por sua constante evolução e pela facilidade com que mudam seus fundamentos. Atualmente nos Estados Unidos foram definidas duas tendências muito diferentes sobre como interpretar a filosofia de utilização e desenvolvimento dos sistemas especialistas. Por um lado são encontrados os defensores do formalismo e a lógica como pilares de seu desenvolvimento. Por outro lado são encontrados os cientistas que propõem começar "entendendo" o mundo antes de tentar formalizá-lo.

Referente ao futuro dos sistemas especialistas tem-se que analisar o que se pretende conseguir com o projeto de quinta geração de computadores que segue:

- Rapidez: Conseguir que os computadores funcionem mais rapidamente, que sejam capazes de processar mais instruções e informação que os computadores atuais no mesmo tempo.

- Trabalho com lógica simbólica: Pretende-se criar *um software capaz de manejar no futuro qualquer tipo de informação* de maneira ótima. Isto permitiria uma melhora substancial no processamento do conhecimento, que esta diretamente relacionado com sistemas especialistas.

- Utilização da linguagem natural: O projeto propõe a consecução desta característica tanto em conceito de compreensão do usuário como em

geração de mensagens, inclusive de voz viva, isto é, que o computador seja capaz de falar.

- Arquitetura tipo paralelo: Até agora os computadores trabalham de uma forma seqüencial, isto é, o microprocessador realiza as tarefas atribuídas uma por uma. Pretende-se conseguir que existam muitos microprocessadores que realizem muitas tarefas de uma só vez.

Pode-se concluir que a barreira entre as soluções da IA atuais e futuras compõem-se de dois desafios muito difíceis . Um é um desafio de equipamento. Os computadores existentes não podem processar bastante conhecimento com bastante rapidez de forma a permitir que os engenheiros construam sistemas aproveitáveis para tratar de muitos tipos de problemas. Se a quantidade de conhecimento necessária para resolver um problema for realmente volumosa, teremos de esperar por computadores capazes de processarem simultaneamente grandes quantidades de regras. Os computadores que hoje estão sendo projetados para manipular este tipo de operação são designados por *sistemas de processamento paralelo ou computadores de quinta geração*. Levará vários anos até que esses computadores estejam disponíveis a preços comerciais.

Um segundo desafio a superar é um problema de programática. Os sistemas de programas atuais podem manipular problemas *não-monóticos*[1] só com bastante dificuldade. Muitos problemas requerem que os dados sejam constantemente reavaliados para levar em conta as contínuas

alterações. Para que os sistemas cognitivos manipulem tais problemas, terão de ser capazes de aprender da própria experiência e atualizar constantemente o seu próprio conhecimento. As várias abordagens à construção de sistemas que possam apreender da experiência são normalmente denominados por *aprendizagem de máquina.* Há vários problemas difíceis que os pesquisadores mal começam a entender e levará vários anos até que haja disponíveis sistemas capazes de apreenderem. Assim, o duplo problema, como construir máquinas capazes de volumoso processamento paralelo e como projetar programas que aprendam da experiência, põem um limite efetivo aos tipos de sistemas cognitivos que serão desenvolvidos nos próximos anos.

O paradigma da orientação a objetos tem se mostrado importante no desenvolvimento de software, pois ele trata de maneira adequada a complexidade inerente aos sistemas computacionais atuais, permitindo modelar a realidade baseando-se em conceitos e abstrações que aproximam-se do processo humano de compreensão da realidade.

Um passo na transposição da barreira entre as soluções da IA atuais e futuras é *a programação de sistemas especialistas a partir dos paradigmas da orientação a objetos* . O objeto cumpre o papel do *idealizado "chip do software"* em um paralelo com o hardware, onde podemos citar o *chip* como grande responsável pela produtividade da indústria. [PAC93]

1 raciocínio monótico - Um sistema de raciocínio fundado na premissa de que, uma vez determinado, um fato não se pode alterar no decorrer do processo racional. Ou seja, num sistema monótico, se o usuário ter respondido uma questão , o sistema considera que a resposta permanecerá a mesma em toda a sessão.

4. MODELO PROPOSTO

4.1 INTRODUÇÃO

Com o intuito de desenvolver um Sistema Especialista para especificação de produtos de imagens e a utilização adequada dos produtos obtidos pelos diferentes sensores, um conjunto de recursos disponíveis para sua aplicação, buscamos construí-lo sob o paradigma da orientação a objetos que tem se mostrado importante no desenvolvimento de software, *"pois ele trata de maneira adequada a complexidade inerente aos sistemas computacionais atuais, permitido modelar a realidade, baseando-se em conceitos e abstrações que aproximam-se do processo humano de compreensão da realidade."*[MEL94]

Para que o Sistema Especialista pudesse efetivamente simular o comportamento dos especialistas humanos na área de Sensoriamento Remoto, possibilitando armazenar e utilizar o conhecimento através de uma *heurística* percorreu-se algumas etapas para o desenvolvimento deste modelo computacional, as quais são abordadas neste capítulo através da: 1- **descrição do problema** - onde aborda-se a aplicabilidade de sistema especialista em sensoriamento remoto, a definição pela construção de um sistema especialista genérico e a importância de sua implementação orientada a objetos. 2- **organização do sistema** que está dividido em **base de conhecimento,** com a

representação do conhecimento baseada em regras através do método tríades Objeto-Atributo-Valor e a implementação num modelo computacional através de orientação a objetos e a implementação do **mecanismo de inferência** construído através do método de encadeamento direto (Foward-Chaining). 3 - **Aplicação do Sistema Especialista para Sensoriamento Remoto** onde descreve-se sua utilização. 4 - **Conclusão.**

4.2 DESCRIÇÃO DO PROBLEMA

O levantamento de dados de recursos naturais da terra com satélites e aeronaves está intimamente relacionado e dependente da tecnologia em computação com processamento de dados para análise de imagens digitais desde 1960.

O Sensoriamento Remoto ao longo da história vem abrindo novos campos de pesquisa e conduzindo a avanços no "pool" de capacidades do computador, como a incrementação de memória, velocidade e outros avanços que resultam na resolução de cores através de poderosos gráficos.

Imagens de satélite constituem uma ferramenta extremamente poderosa e de custo relativamente baixo, na solução de vários problemas, como por exemplo, o monitoramento da cobertura vegetal para facilitar o controle sobre desmatamentos e queimadas; a identificação e

avaliação de áreas cultivadas para auxiliar na previsão de safras e permitir o planejamento de áreas urbanas para análise e planejamento macroeconômicos.

Embora existam disponíveis no Brasil desde 1975, ainda hoje desenvolvem-se metodologias de uso dessas imagens nas várias áreas de aplicação: Análise Ambiental, Florestas, Agricultura, Geologia e Cartografia. Apesar da falta de conhecimento técnico a respeito desta ferramenta dificultar sua maior disseminação, o uso de imagens de satélites tem aumentado ano a ano.

Como os produtos obtidos por sensoriamento remoto são distribuídos a usuários que não tem um profundo conhecimento das técnicas de processamento de imagens, existe a necessidade da aplicação de técnica de inteligência artificial para solucionar os problemas gerais da seleção, análise e interpretação de imagens.

A aplicação de Sistema Especialista para a especificação de produtos de imagens de satélites e a utilização adequada dos produtos obtidos pelos diferentes sensores faz-se necessário para auxiliar o usuário leigo em Sensoriamento Remoto na tarefa de especificar um produto imagem, dado um conjunto de recursos disponíveis, ou seja, saber como selecionar os produtos para extrair a informação desejada da imagem gerada.

Para a implementação deste Sistema Especialista num modelo computacional, busca-se construí-lo sob o paradigma da orientação a objetos.

" O paradigma da orientação a objetos tem se mostrado importante no desenvolvimento de sistemas nos anos 90. Esse paradigma oferece abstrações que aproximam-se do processo de compreensão da realidade do ser humano[1]. De acordo com a edição inglesa da Enciclopédia Britânica, tem-se na Teoria da Classificação (conforme citado em [COA92]):

> "Na compreensão do mundo real, as pessoas empregam constantemente três métodos de organização, sempre presentes em todos os seus pensamentos:
>
> (1) diferenciação, baseado na experiência de cada um, de objetos particulares e seus atributos -- por exemplo, quando distinguem uma árvore, e seu tamanho ou relações espaciais, dos outros objetos;
>
> (2) distinção entre objetos como um todo e entre suas partes componentes - por exemplo, quando separam uma árvore de seus galhos;
>
> *(3) formação de, e distinção entre, as diferentes classes de objetos - por exemplo, quando formam uma classe de todas as árvores, uma outra de todas as rochas e distinguem-nas."*

[1] Coad e Yourdon, em [COA92], enumeram e explicam estas relações, utilizadas no processo de análise orientada a objetos.

A modelagem computacional[2] de uma determinada realidade pode ser feita de forma mais "natural" através do paradigma da orientação a objetos. Não é necessário impor ao modelo conceitos artificiais ao domínio do problema que esta sendo modelado. Existe uma estreita equivalência entre os requisitos do usuário do sistema e a representação do seu modelo. Devido ao seu poder de representação, o paradigma da orientação a objetos viabiliza e estimula o desenvolvimento de *software* de qualidade.

De acordo com Bertrand Meyer, em [MEY88], as características chaves que um *software* de qualidade deve possuir são: corretude, robustez, extensibilidade, reusabilidade e compatibilidade. A orientação a objetos apresenta mecanismos que permitem implementar de maneira coerente estas características:

- corretude: é a habilidade do software de produzir exatamente os resultados definidos pelos requisitos do usuário. O paradigma da orientação a objetos favorece muito este aspecto, pois utiliza, na sua representação, os processos de compreensão da realidade utilizados pelo ser humano. Pelo fato das estruturas do paradigma serem compatíveis com as estruturas do processo de pensamento humano, os requisitos são melhor transportados para as fases de análise e de projeto, conseqüentemente proporcionando à implementação, maior corretude diante dos requisitos determinados pelo usuário do sistema;
- robustez: é a capacidade do software de funcionar mesmo em condições anormais, não previstas nas especificações. Um modelo

[2] A expressão "modelo computacional" refere-se à toda documentação existente desde a análise até a efetiva implementação do sistema, a qual especifica o sistema que se deseja implementar de acordo com as necessidades do usuário.

computacional orientado a objetos minimiza a existência dessas condições, pois os objetos nos quais se baseia são, por natureza, componentes robustos. Isto deve-se ao fato de que os objetos, graças à *interface* inerentemente bem definida que apresentam e ao princípio do encapsulamento de informações, controlam plenamente as operações passíveis de execução, e não permitem a execução de outras operações que não estejam contidas nesta *interface*;

- extensibilidade: é a facilidade com que um *software* pode ser adaptado às mudanças (inevitáveis) da realidade que este modela. Os grandes sistemas possuem certa complexidade que dificulta a sua adaptação a novas especificações. Dentro deste contexto, a orientação a objetos configura-se como uma proposta adequada para diminuir a complexidade e facilitar a extensibilidade dos sistemas. Ao utilizar a análise orientada a objetos, um modelo de realidade baseia-se nos elementos mais estáveis do sistema (que são os objetos tratados pelo sistema), ao invés de definí-lo pela sua funcionalidade (o qual, geralmente, é o elemento mais volátil, o primeiro a sofrer modificações). Os objetos também oferecem uma melhor visão do domínio do problema, o que facilita em muito a determinação dos elementos a serem modificados (quando necessário) e qual a repercussão das mudanças nos restante dos elementos do sistema;
- reusabilidade: é a habilidade do software de ser reusado, totalmente ou em parte, por novas aplicações. Este é um dos mais significativos benefícios introduzidos pela orientação a objetos: facilitar a reusabilidade. Através das técnicas presentes neste paradigma, a reusabilidade torna-se algo sempre presente, desde a análise até a

implementação. A herança é o principal mecanismo responsável pela reusabilidade no paradigma, que consiste no processo de especializar e/ou generalizar classes existentes;

- compatibilidade: é a facilidade com que o *software* (ou seus componentes) pode ser combinado com outros. No paradigma da orientação a objetos, cada objeto (inclusive o sistema como um todo, que pode ser considerado com um único objeto) contém um protocolo de comunicação (sua *interface*) bem definido e estável, formalizando a compatibilidade com os outros objetos: basta utilizar este protocolo.

É justamente esta propriedade, de oferecer de forma inerente mecanismos transparentes e concisos que permitam o desenvolvimento de *software* de qualidade, que torna a orientação a objetos tão relevante no campo da engenharia de *software."*[MEL94]

Além das construções descritas acima, a programação orientada a objetos (POO) e, por conseqüinte, as linguagens de programação que são baseadas na POO, devem ter outros recursos inerentes ao paradigma, tais como:

- **Abstração**: O dicionário define abstração *como "ato de separar mentalmente um ou mais elementos de uma totalidade complexa (coisa, representação, fato), os quais só mentalmente podem subsistir fora dessa totalidade*". É interessante notar que esta definição contém dois dos conceitos-chaves definidos como parte da abordagem orientada para o objeto em relação à solução de problemas. As abstrações suportadas por um software e por uma linguagem

possibilitam o desenvolvimento de soluções de problemas de alto nível.[WIE91]

- **encapsulamento**: a possibilidade de agregar em uma única unidade funcional, os dados que retratam as características de um certo objeto (elemento de um modelo que se deseja representar) e as respectivas operações (determinadas execuções que podem ser feitas sobre os dados de um objeto) passíveis de realização sobre este objeto. Encapsulação: O dicionário define encapsulação como o resultado do "*processo de incluir ou encerrar em cápsula*". Em uma linguagem orientada para o objeto, a unidade de encapsulação é o objeto. A cápsula ou objeto inclui entidades específicas como os dados de estado e os protocolos de mensagem que os objetos respondem.[WIE91]
- **generalização-especialização**: uma relação entre duas classes, onde a primeira é a representação de um grupo de elementos com algumas características semelhantes que os identificam, enquanto a segunda classe é um subconjunto dos elementos da primeira, especializando algumas características inerentes somente a um certo número de elementos da primeira classe; essa construção tem um mecanismo associado na programação orientada a objetos, denominado de *herança*; Herança : O dicionário define herança como *"aquilo que se herda; bem, direito ou obrigação transmitido por via de sucessão ou disposição testamentária; aquilo que se recebeu dos pais, das gerações anteriores, da tradição."* A definição está claramente

voltada à herança de pessoas; no entanto, o conceito pode ser aplicado a objetos e classes como ilustram as classes biológicas. No caso da programação orientada para o objeto e dos objetos, a herança é interpretada como o recebimento de propriedades (dados de estado e protocolos de mensagem) vindas do protocolo especificado em uma classe mãe. [WIE91]

- **polimorfismo**: inexistente na estrutura estática de um *software*, o polimorfismo aparece quando, durante a execução do sistema, a instância de uma classe (um objeto) é tratada como se fosse a instância de uma outra classe. É a forma com que uma determinada operação aplicada a uma classe pode ser diferenciada quando executada sobre as classes que a herdaram (aplicando-se a construção de generalização-especialização entre a primeira classe e as demais). Polimorfismo : O dicionário define polimorfismo como a "*existência de órgãos ou plantas com diversas formas*". Na programação orientada para objeto, essa propriedade, obviamente, se aplica às mensagens e suas implementações. Significa também que uma determinada mensagem toma forma no momento da execução. Ou seja, a forma propriamente dita de implementação de uma mensagem é determinada por e está vinculada a um objeto no momento da execução.[WIE91]

4.2.1 MODELAGEM

Para que o Sistema Especialista possa efetivamente simular o comportamento dos especialistas humanos na área de Sensoriamento Remoto, possibilitando armazenar e utilizar o conhecimento através de uma *heurística,* a partir de um modelo computacional, fez-se necessário modelá-lo - o Sistema Especialista - de acordo com orientações presentes na OOA. Este modelo está representado na FIGURA 4.1.

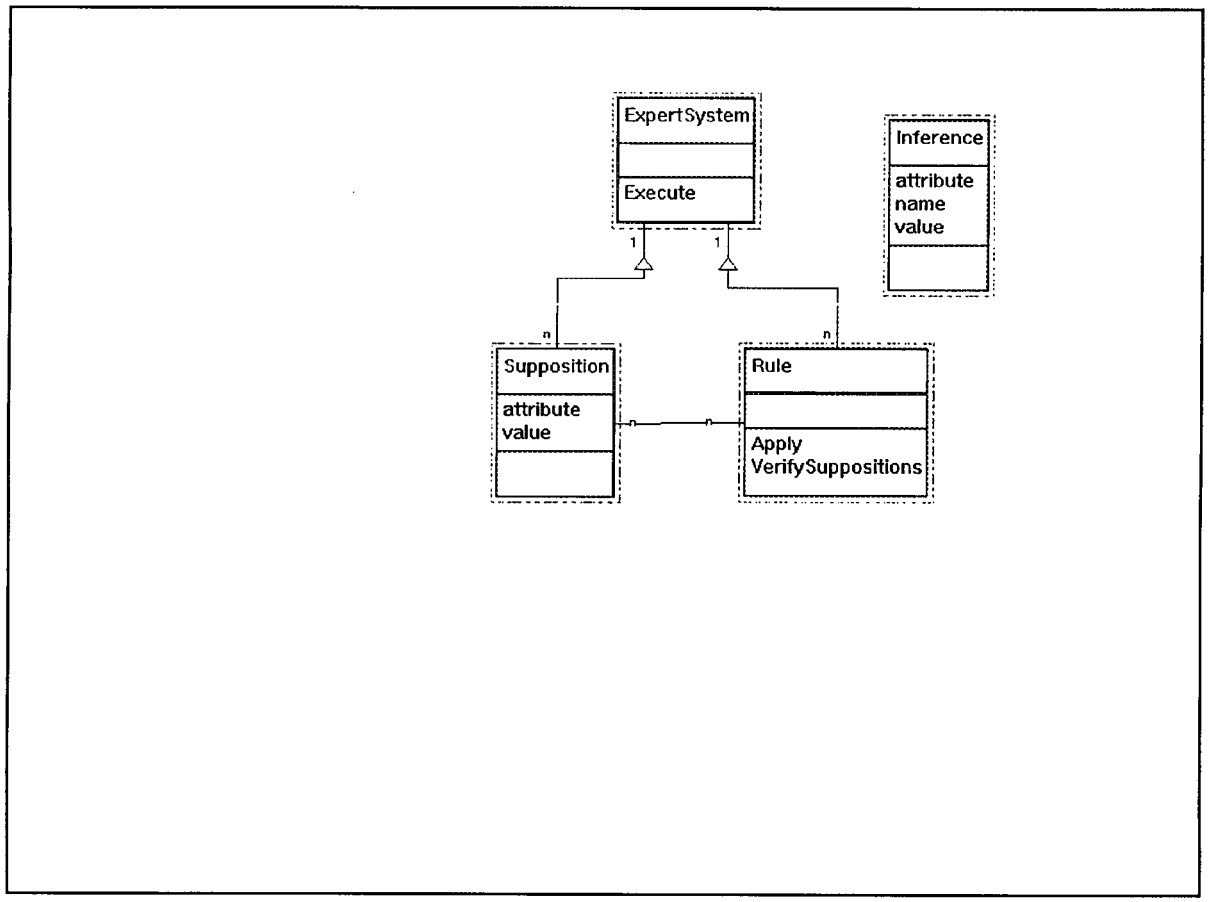

FIGURA 4.1 - modelagem do sistema especialista

O Sistema Especialista foi desenvolvido como um sistema especialista genérico como pode-se observar, porque lhe permite usar a mesma máquina de inferência com diferentes bases de conhecimento. Isso traz vantagens do tipo, pode-se alterar a base de conhecimento facilmente, incorporando novos conhecimentos.

4.2.2 IMPLEMENTAÇÃO

Na implementação do Sistema Especialista seguiu-se a prototipação para o desenvolvimento do software. Definiu-se, como requisitos básicos que o sistema especialista deveria atender seriam:

- permitir, de alguma maneira, a edição da base de regras para que se for preciso alterar, acrescentar ou aumentar a base de conhecimento não implique na alteração do código do programa e o usuário possa treinar "em campo" sem precisar reprogramar e consequente recompilar o programa.

- que a base de conhecimento esteja separada do mecanismo de inferência e esse possa ser utilizado com diferentes bases de conhecimento, dado a variedade de aplicações em sensoriamento remoto;

- o Sistema Especialista não deve perguntar sobre os mesmos atributos mais de uma vez;

- o Sistema Especialista deve rejeitar imediatamente e deixar de lado qualquer objeto que não possua um ou mais dos atributos necessários conhecidos ou que tenham um atributo que já tenha sido rejeitado.

Antes de iniciar a implementação, fez-se necessário definir qual a plataforma de funcionamento do Sistema Especialista e qual a ferramenta utilizada para desenvolvê-lo.

4.3.1.1 PLATAFORMA E AMBIENTE DE DESENVOLVIMENTO

O sistema especialista foi desenvolvido para ambiente MS-Windows, seguindo uma tendência mundial que leva às GUIs[3] (foge do escopo deste relatório discutir sobre as vantagens das *interfaces* homem-máquina desenvolvidas seguindo este paradigma).

Quanto à escolha do ambiente de desenvolvimento a utilizar, para desenvolver o sistema especialista, o escolhido, dentre os ambientes de desenvolvimento existentes para a linguagem C++, foi o Borland C++, versão 4.0, da Borland International. A escolha recaiu sobre este ambiente por suportar, de maneira adequada, conforme analise feita em [MEL94] a produção de *software* que se enquadre nos ideais buscados de

[3] GUI - Graphical User Interface.

produtividade, qualidade, robustez, facilidade de reutilização de elementos e *interface* coerente. Além disso, o índice de produtividade alcançado por um desenvolvedor que utilize o Borland C++ é bastante satisfatório (se comparado a outros ambientes de desenvolvimento C++ para MS-Windows), devido principalmente à presença de um *application framework* (a OWL) e de uma biblioteca de estrutura de dados (a CLASSLIB) no pacote.

A OWL - *Object Windows Library* - é um conjunto de classes que encapsula quase que totalmente a complexidade da API - *Application Program Interface* - do MS-Windows. A API é formada por centenas de funções, organizadas de maneira nada exemplar e cujo uso é complexo, demandando muito tempo de aprendizado e muita paciência na sua utilização. A OWL, ao fazer uma "casca" orientada a objetos em torno da API, esconde muito de seus defeitos, permitindo ao desenvolvedor de *software* preocupar-se menos com as armadilhas da API e mais com os objetivos específicos do programa.

Já a CLASSLIB é uma biblioteca de classes de estruturas de dados, inspirada fortemente na biblioteca semelhante que acompanha o Smalltalk. Ao fazer a adaptação dessas classes para C++, porém, a Borland utilizou várias características de C++ inexistentes em Smalltalk (tal como a herança múltipla), que tornam a versão C++ da biblioteca mais flexível e poderosa.

Além da OWL e da CLASSLIB, o pacote do Borland C++ contém o Resource Workshop, que é um editor visual de recursos[4]. O uso do Resource Workshop facilita o desenvolvimento da *interface* de um programa.

4.3 ORGANIZAÇÃO DO SISTEMA

Pesquisadores de Inteligência Artificial têm observado a importância do conhecimento específico para a resolução de problemas de dificuldade significativa. Tem sido observado que o fato de uma pessoa ter um elevado QI não a torna um especialista. Na realidade, o que torna uma pessoa especialista é a *quantidade* de conhecimento especializado que possui.

Sistemas Baseados em Conhecimento são empregados para resolver problemas que costumam requerer a inteligência humana [Hay84]. A caracteristica geral de tais sistemas é a representação explícita do conhecimento, que é armazenado de modo flexível e modular na "*base de conhecimento*". Quando o sistema necessita raciocinar sobre problemas específicos do domínio em questão, a base de conhecimento é acessada de modo a obter as informações necessárias à realização da inferência. O mecanismo de controle capaz de navegar pela base de conhecimento em busca de uma solução para o problema é conhecido como "*máquina de inferência*".

Os Sistemas Baseados em Conhecimento devem ser capazes de seguir linhas de raciocínio que pareçam mais apropriadas aos dados

[4] Recursos são elementos de *interface* de um programa MS-Windows, tais como menus, caixas de diálogo e *bitmaps*. Eles estão extensamente documentados em [MSC92].

disponíveis para a solução automática dos problemas. Além disso, devem ser capazes de incorporar novas informações à base de conhecimento.

A FIGURA 4.2 mostra a estrutura básica de um sistema baseado em conhecimento através de um módulo de interface.

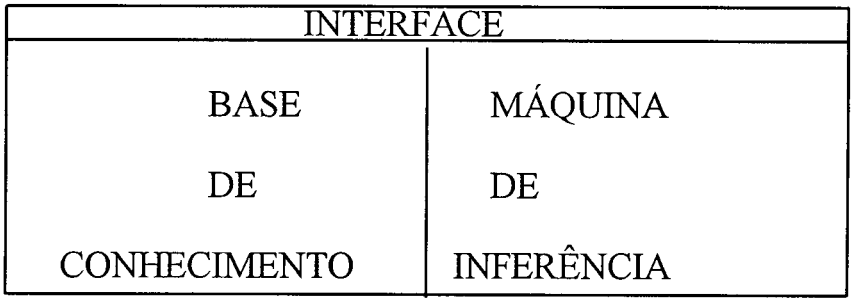

FIGURA 4.2 - Estrutura Básica de um Sistema Baseado em Conhecimento

A *estrutura de um sistema especialista* é bastante diferente de um sistema convencional. Aqui, não existe um algorítmo explícito. O conhecimento é armazenado (normalmente sob a forma de regras do tipo "se...então...") em um módulo separado, chamado *base de conhecimento*. Já o processamento dessas regras é executado por um módulo que analisa a base de conhecimento usando ténicas de procura em espaço de estados, chamado *máquina (ou motor) de inferência*. Existe também um terceiro módulo, uma *interface* que realiza a interação com o usuário em linguagem natural (ou tão natural quanto possível), e responde a perguntas como: "Por que esta pergunta está sendo feita?", "O que significa essa pergunta que o sistema está me fazendo - não a entendi" e outras.

É importante ressaltar que, construída a máquina de inferência, o projetista pode acrescentar (ou retirar) uma ou mais regras à base

de conhecimento, praticamente sem ter que se preocupar com o seu processamento, e sem ter que mudar um programa ou algorítmo. Isso se traduz em benefícios, como melhor organização e clarificação do conhecimento embutido no sistema, facilidade de manutenção e construção, etc.

4.3.1 BASE DE CONHECIMENTO

A base de conhecimento dá as características de funcionamento do sistema. Este terá o conhecimento, do que for colocado na sua base de conhecimento, isto é, se ela for projetada para receber informações de uma determinada ciência, o sistema será especialista nessa ciência. É o local onde são armazenados fatos e regras.

Para implementação concreta da base de conhecimento que nos referimos é necessário contar com uma forma adequada de representação do conhecimento.

O modelo computacional, implementado o banco de conhecimento, contém as *premissas e regras* que reúnem o conhecimento do especialista. Para representar o conhecimento baseado em regras seguimos a estratégia de abordagem de Tríades Ojeto-Atributo-Valor (O-A-V). Nesse esquema, os ***objetos*** podem ser entidades físicas, tais como uma porta ou um transistor ou podem ser entidades conceituais, tais como um porta lógica, um empréstimo bancário ou um episódio de vendas. Os ***atributos*** são características ou propriedades gerais associadas aos objetos. O tamanho, a forma, a cor são atributos típicos dos objetos físicos. A taxa de juros é um atributo do empréstimo bancário e o ambiente pode ser um atributo para um

acontecimento de vendas. O último membro da tríade é o ***valor*** de um atributo. O valor especifica a natureza específica de um atributo em uma situação particular. A cor de uma maçã, por exemplo, pode ser vermelha ou a taxas de juros do empréstimo bancário pode ser doze por cento. A tríade O-A-V é intuitivamente mais simples de usar porque contém em si, categorizações de conhecimento. A maioria dos sistemas cognitivos construídos até hoje usam tríades O-A-V e regras para representar o conhecimento que encerram.

O sistema especialista implementado é composto de - **Premissas** *atributo* , *valor* são declarados como string's - **Regras** *objeto, atributo* , *valor* são declaradas como string e - **Conclusão** *objeto, atributo, valor* também declarados como string. A base de conhecimento esta implementada atualmente como mostramos na FIGURA 4.3, mas para expandi-la basta editar o arquivo e acrescentar novas regras:

```
PREMISSAS

1;área;ambiental
2;domínio;amazônia
3;infra-estrutura;papel
4;tipo;uso da terra
5;área;agricultura
6;região;centro-sul
7;domínio;arroz
8;mês plantio;11
9;área;florestal
10;domínio;caatinga
11;relevo;nil
12;solo;nil
13;tipo;queimada
14;infra-estrutura;papel
```

```
REGRAS

1,2,3,4;PI;tipo produto;PB
1,2,3,4;PI;escala;1:50000
1,2,3,4;PI;bandas;3,4,5
1,2,3,4;PI;tipo produto;composição
5,6,7,8;PI;passagem início;1
5,6,7,8;PI;passagem fim;2
5,6,7,8;PI;satélite;TM5
9,10,11,12,13,14;PI;tipo produto;PB
9,10,11,12,13,14;PI;escala;1:100000
9,10,11,12,13,14;PI;bandas;5,4
9,10,11,12,13,14;PI;escala;1:250000
9,10,11,12,13,14;PI;passagem início;8
9,10,11,12,13,14;PI;passagem fim;10

FIM
```

FIGURA 4.3 - base de conhecimento (exemplo)

As premissas* podem ser implementadas da seguinte na forma:

nº ; atributo ; valor

1; área ; ambiental
2; domínio ; amazônia
3; infra-estrutura ; papel
4; tipo ; uso da terra

E as regras* com a seguinte representação:

nº , ... nº ; objeto ; atributo ; valor

1,2,3,4; PI ; tipo de produto ; PB

A partir da escolha do conjunto de premissas e essas satisfazendo determinada regra . A regra é "disparada".

4.3.2 MECANISMO DE INFERÊNCIA

* Não esquecer da separação ponto-virgula (;)

É o elemento de um sistema especialista capaz de buscar as regras necessárias a serem avaliadas, ordenadas de uma maneira lógica e, a partir daí, ir direcionando o processo de inferência. Ou seja é a parte do sistema que usa a informação que você fornece, para encontrar um objeto correspondente.

Uma das maneiras para construir uma *máquina de inferência* é o método de encadeamento direto métodos relaciona-se à maneira como a máquina busca o objetivo procurado. *a-* ***Metodo de encadeamento direto (Foward-Chaining)****. Dessa forma uma máquina de inferência de encadeamento direto começa com algumas informações e, então, tenta encontrar um objeto que contenha as informações. O método de encadeamento direto faz do processo de derivação da enorme porção de informação da base de conhecimento algo mais fácil, porque constrói uma árvore. Um sistema de encadeamento direto típico encontra todos os objetos possíveis que se casam com os atributos. Em um sistema de encadeamento progressivo ou direto, as premissas das regras são examinadas para ver se são ou não verdadeiras, dadas as informações à mão. Se forem, então acrescentar-se-ão as conclusões à lista de fatos que se sabem verdadeiros e o sistema examina novamente as regras. Os sistemas de encadeamento progressivo chamam-se, às vezes, sistemas conduzidos por dados. Os sistemas de encadeamento direto tornam clara a distinção entre o banco de conhecimento e a memória de trabalho. A memória de trabalho contém fatos que surgem em uma consulta. As premissas das regras em um banco de conhecimentos comparam-se ao conteúdo da memória de trabalho. Quando uma regra é aprovada, a sua conclusão é colocada na memória de trabalho. O racioncínio em um sistema de encadeamento progressivo descreve-se como um ciclo de "reconhecer-agir", ou seja,* reconize-act cycle. *Primeiro*

reconhecem-se as regras que podem aprovar, dado o conteúdo da memória de trabalho. Escolhe-se uma só regra e depois inscreve-se na memória de trabalho a ação ou conclusão. Depois, o sistema prossegue para o próximo ciclo e verifica-se novamente para ver que regras são aprovadas.Sendo necessários dispor de métodos para: 1- Decidir onde começar a analisar as regras que são armazenadas no banco de conhecimentos; 2- Resolver os conflitos entre as regras e os loops de indução. Além disso, é necessário que o sistema trabalhe otimamente, gestionando o espaço de busca de uma forma adequada.

A implementação do mecanismo de inferência segue o seguinte algoritmo:

```
ESCOLHER UMA PREMISSA PARA PERGUNTAR

FAZ A PERGUNTA Se Atributo=Valor?

        SE sim
                Valida premissa com valor do atributo = ao da premissa respondida
                Invalida Premissas Com valor do Atributo diferente da premissa
                AplicaRegrasPassiveis
                                // selecionar regras
                                // aplicar a regra
                                // eliminá-la


        SE não
                Elimina Regras Dependentes Da Premissa;

FIM SE
```

FIGURA 4.4 Algoritmo do Mecanismo de Inferência

4.4 Conclusão

O Sistema Especialista desenvolvido cumpre os seus objetivos de maneira bastante consistente, auxiliando o usuário na especificação de produto imagem e satélite. Quanto à *interface* do sistema especialista, foi adaptada a novos padrões que vem sendo ditados pelas atualizações do MS-Windows (plataforma de funcionamento do sistema especialista), tais como o uso de barras de ferramentas e barras de mensagens.

O sensoriamento remoto da terra com satélites e aeronaves está intimamente relacionado e dependente da tecnologia em computação com processamento de dados para análise de imagens digitais desde 1960. Embora existam disponíveis no Brasil desde 1975, ainda hoje desenvolvem-se metodologias de uso dessas imagens nas várias áreas de aplicação: Análise Ambiental, Florestas, Agricultura, Geologia e Cartografia. Apesar da falta de conhecimento técnico a respeito dessa ferramenta dificultar sua maior disseminação, o uso de imagens de satélites tem aumentado ano a ano. Isso comprova a necessidade da aplicação de Sistema Especialista para a especificação de produto de imagem e a utilização adequada de produtos por diferentes sensores faz-se necessário para auxiliar o usuário leigo em Sensoriamento Remoto na tarefa de especificar um produto imagem, dado um conjunto de recursos disponíveis, ou seja, saber como selecionar os produtos, para extrair a informação desejada da imagem gerada.

Se é verdadeiro a afirmação de [HAR89] que a barreira entre as soluções da IA atuais e futuras compõem-se de dois desafios muito dificieis. Sendo um deles o desafio de equipamento e que para solucionar esse problema estariam sendo projetados computadores para manipular quantidade volumosa de conhecimento necessária para resolver um problema e tem-se que esperar por computadores capazes de processarem simultaneamente grandes quantidades de regras, esse tipo de operação são designados por *sistemas de processamento paralelo ou computadores de quinta geração*. E um segundo desafio a ser superado é um problema de programática. Pois, muitos problemas requerem que os dados sejam constantemente reavaliados para levar em conta as contínuas alterações. Para que os sistemas cognitivos manipulem tais problemas, terão de ser capazes de aprender da própria experiência e atualizar constantemente o seu próprio conhecimento. As várias abordagens à construção de sistemas que possam apreender da experiência são normalmente denominados por *aprendizagem de máquina.*

Um passo na transposição da barreira - no que diz respeito ao segundo problema - entre as soluções da IA atuais e futuras é *a programação de sistemas especialistas a partir dos paradigmas da orientação a objetos* - que conforme [PAC94] o objeto cumpre o papel do *idealizado "chip do software"* em um paralelo com o hardware, onde podemos citar o *chip* como grande responsável pela produtividade da indústria.

Podemos concluir que *resultado de unir o conhecimento de um especialista com os computadores mais potentes e rápidos está criando um novo ambiente que mudará completamente as estratégias de resolução de*

problemas antigos e além disso permitirá enfrentar os outros intratáveis na atualidade.[HAR89]

5. VALIDAÇÃO DO MODELO

5.1 INTRODUÇÃO

Neste capítulo, mostra-se , através de diversas aplicações, que o modelo computacional implementado é bastante adequado para a especificação de produtos de imagens de Satélites.

Inicialmente é feito do **delineamento do experimento.**

A seguir, é apresentado um quadro de **respostas dos especialistas.**

Posteriormente faz-se uma **análise comparativa** das diversas aplicações realizadas com a execução do Sistema Especialista.

Finalmente uma **conclusão** é apresentada.

5.2 DELINEAMENTO DO EXPERIMENTO

A especificação de produtos de imagens de Satélites e a utilização adequada dos produtos obtidos pelos diferentes sensores, depende, em geral do tipo de problema a ser resolvido e do domínio ao qual o problema está associado. Cada área de aplicação, contudo requer que uma série de outros atributos sejam determinados de modo a identificar melhor o problema e permitir a especificação de um produto de imagem, dado um conjunto de

recursos disponíveis, ou seja, saber qual o produto adequado a ser utilizado para extrair a informação desejada da imagem gerada.

Para validação do modelo computacional proposto busca-se implementar na *base de conhecimento* as várias aplicações, originárias de fontes literárias, com as seguintes regras:

USO DA TERRA

Escala mapa final: 1:100.000

Recomendação:

- produto digital - Sensor TM - bandas 3, 4, 5
  ou
  - Sensor HRV - bandas XS1, XS2 e XS3

- produto fotográfico - Sensor TM - composição colorida - bandas 3, 4, 5
  escala 1: 100.000

USO DO SOLO

Escala mapa final: 1: 50.000

Recomendação:

- produto digital - Sensor TM - bandas 3, 4 e 5
  ou
  - Sensor HRV - banda pancromática

- produto fotográfico - Sensor HRV - banda pancromática
  escala 1: 50.000

EXPANSÃO DA ÁREA URBANA

Escala mapa final : 1:50.000

Recomendação:

- produto digital - Sensor TM - bandas 3, 4 e 5
  - Sensor HRV - bandas XS1, XS2 e XS3

- produto fotográfico - Sensor TM - escala 1:50.000 - banda 3
  - Sensor HRV - escala 1:50.000 - banda XS2

ÁREAS DE REFLORESTAMENTOS

Escala mapa final : 1:100.000

Recomendação:

- produto digital - Sensor TM - bandas 3, 4 e 5
  - Sensor HRV - bandas XS1, XS2 e XS3

- produto fotográfico - Sensor TM - escala 1:100.000 - banda 3
  - Sensor HRV - escala 1:100.000 - banda XS2

ESTUDOS DE SEDIMENTOS EM SUSPENSÃO EM CORPOS D'ÁGUA

Escala mapa final : 1:50.000

Recomendação:

- produto digital - Sensor TM - bandas 1, 2 e 3
- produto fotográfico - escala 1:50.000 - composição colorida - bandas 1, 2 e 3

DELIMITAÇÃO DE CORPOS DE ÁGUA

Escala mapa final: 1:50.000

Recomendação:

- produto digital - Sensor TM - bandas 3, 4 e 5
- produto fotográfico - escala 1:50.000
  - Sensor TM - banda 4
  - Sensor HRV - banda XS3

MAPEAMENTO DA REDE DE DRENAGEM

Escala mapa final: 1:50.000

Recomendação:

- produto digital - Sensor TM - bandas 3, 4 e 5
- produto fotográfico - escala 1:50.000
  - Sensor TM - banda 4
  - Sensor HRV - banda XS3

ESTRESSES NA VEGETAÇÃO

Escala mapa final: 1:100.000

Recomendação:

- produto digital - Sensor TM - banda 5

- produto fotográfico - escala 1:100.000
  - Sensor TM - banda 5

IDENTIFICAÇÃO DA REDE VIÁRIA

Escala mapa final: 1:50.000

Recomendação:

- produto digital - Sensor HRV - banda pancromática

- produto fotográfico - escala 1:50.000

- Sensor HRV - banda pancromática

IDENTIFICAÇÃO DE ÁREAS DESMATADAS

Escala mapa final: 1:50.000

Recomendação:

- produto digital - Sensor TM - bandas 3, 4 e 5
- Sensor HRV - bandas XS1, XS2 e XS3

- produto fotográfico - escala 1:50.000
- Sensor TM - banda 3
- Sensor HRV - banda XS2

MONITORAMENTO DA QUALIDADE DA ÁGUA

Escala mapa final: 1:50.000

Recomendação:

- produto digital - Sensor TM - bandas 1, 2 e 4

- Sensor HRV - bandas XS1, XS2 e XS3

ESTUDO DE ÁREAS DEGRADADAS

Escala mapa final: 1:50.000

Recomendação:

- produto fotográfico - Sensor TM - bandas 2, 3 e 4

- produto digital - Sensor TM - bandas 3, 4 e 5

MONITORAMENTO DA AÇÃO ANTRÓPICA

Escala mapa final: 1:100.000

Recomendação:

- produto fotográfico - Sensor TM - bandas 3 e 4

- produto digital - Sensor TM - bandas 3, 4 e 5

- Sensor HRV - bandas XS1, XS2 e XS3

ESTUDO DA EROSÃO

Escala mapa final: 1:50.000

Recomendação:

- produto digital - Sensor TM - bandas 3, 4 e 5

- Sensor HRV - bandas XS1, XS2 e XS3

TIPOS DE SOLO

Escala mapa final: 1:100.000

Recomendação:

- produto digital - Sensor TM - bandas 3, 4 e 7
- produto fotográfico - Sensor TM - bandas 7, 4 e 3

INVENTÁRIO FLORESTAL

Escala mapa final : 1:100.000

Recomendação:

- produto digital - Sensor TM - bandas 3, 4 e 5

IDENTIFICAÇÃO DE ÁREAS DE QUEIMADAS

Escala mapa final : 1:50.000

Recomendações:

- produto digital - Sensor TM - bandas 3, 4 e 5
- produto fotográfico - Sensor TM - banda 4
  - Sensor HRV - banda XS3

IDENTIFICAÇÃO DE ÁREAS DE SOLO EXPOSTO

Escala mapa final : 1:50.000

Recomendações:

- produto digital - Sensor TM - bandas 3, 4 e 5
- produto fotográfico - Sensor TM - banda 3
  - Sensor HRV - banda XS2

IDENTIFICAÇÃO DE ÁREAS DE MINERAÇÃO

Escala mapa final: 1:50.000

Recomendação:

- produto digital - Sensor TM - bandas 3, 4 e 5
- produto fotográfico - Sensor TM - banda 4
  - Sensor HRV - banda XS3

IDENTIFICAÇÃO DE ÁREAS IRRIGADAS

Escala mapa final: 1:50.000

Recomendação:

- produto digital - Sensor TM - banda 3, 4 e 5

- produto fotográfico - Sensor TM - banda 4
  - Sensor HRV - banda XS3

IDENTIFICAÇÃO DE ÁREAS OCUPADAS COM PINUS E EUCALIPTOS

Escala mapa final: 1:100.000

Recomendação:

- produto fotográfico - Sensor TM - banda 4
  - Sensor HRV - banda XS3
  - escala 1:100.000

IDENTIFICAÇÃO DE DUNAS

Escala mapa final : 1:50.000

Recomendação:

- produto digital - Sensor TM - banda 1, 2 e 3
  - Sensor HRV - banda XS3

ESTUDO DE ASSOREAMENTO

Escala mapa final: 1:50.000

Recomendação:

- produto digital - Sensor TM - banda 1, 2 e 3

- produto fotográfico - Sensor TM - banda 3
  - Sensor HRV - banda XS2

5.3 RESPOSTAS DOS ESPECIALISTAS

APLICAÇÃO	RECOMENDAÇÃO	ESP01	ESP02	ESP03
USO DA TERRA	EMF 1:100.000 - PD - SENSOR TM - 3,4,5	X	X	X
USO DO SOLO	EMF 1:50.000 - PD - SENSOR TM - 3,4,5	X	X	X
EXPANSÃO DA ÁREA URBANA	EMF 1:50.000 - PD - SENSOR TM - 3,4,5 EMF 1:50.000 - PD - SENSOR HRV - XS1,XS2,XS3	X -	X -	- X
ÁREAS DE REFLORESTAMENTO	EMF 1:100.000 - PD - SENSOR TM - 3,4,5 EMF 1:100.000 - PF - SENSOR TM - E 1:100.000 - 3	- **1**	X -	X -
ESTUDOS DE SED EM SUSP EM C D'ÁGUA	EMF 1:50.000 - PD - SENSOR TM - 1,2,3	X	X	**2**
DELIMITAÇÃO DE CORPOS DE ÁGUA	EMF 1:50.000 - PD - SENSOR TM - 3,4,5	X	X	X
MAPEAMENTO DA REDE DE DRENAGEM	EMF 1:50.000 - PD - SENSOR TM - 3,4,5 EMF 1:50.000 - PF - SENSOR TM - 4 EMF 1:50.000 - PF - HRV - XS3	X -	X -	- **3**
ESTRESSES NA VEGETAÇÃO	EMF 1:100.000 - PD - SENSOR TM - 5	X	X	X
IDENTIFICAÇÃO DA REDE VIÁRIA	EMF 1:50.000 - PD - SENSOR HRV - BANDA PAN EMF 1:50.000 - PF - SENSOR HRV - BANDA PAN	X	X	-
IDENTIFICAÇÃO DE ÁREAS DESMATADAS	EMF 1:50.000 - PD - SENSOR TM - 3,4,5 EMF 1:50.000 - PD - SENSOR HRV - XS1,XS2,XS3	X -	X -	- X
MONITORAMENTO QUALIDADE DA ÁGUA	EMF 1:50.000 - PD - SENSOR TM - 1,2,4 EMF 1:50.000 - PD - SENSOR HRV - XS1, XS2, XS3	- X	**4** -	X -
ESTUDO DE ÁREAS DEGRADADAS	EMF 1:50.000 - PF - SENSOR TM - 2,3,4 EMF 1:50.000 - PD - SENSOR TM - 3,4,5	X -	- X	- X
MONITORAMENTO DA AÇÃO ANTRÓPICA	EMF 1:50.000 - PD - SENSOR TM - 3,4,5 EMF 1:50.000 - PD - SENSOR HRV - XS1,XS2,XS3	X -	X -	- X
ESTUDO DA EROSÃO	EMF 1:50.000 - PD - SENSOR TM - 3,4,5 EMF 1:50.000 - PD - SENSOR HRV - XS1, XS2, XS3	- X	**5** -	X -
TIPOS DE SOLO	EMF 1:100.000 - PD - SENSOR TM - 3,4,7	X	X	**6**
INVENTÁRIO FLORESTAL	EMF 1:100.000 - PD - SENSOR TM - 3,4,5	X	X	X
IDENTIF DE ÁREAS DE QUEIMADAS	EMF 1:50.000 - PD - SENSOR TM - 3,4,5	X	X	X
IDENTIF DE ÁREAS DE SOLO EXPOSTO	EMF 1:100.000 - PD - SENSOR TM - 3,4,5	X	X	X
IDENTIF DE ÁREAS DE MINERAÇÃO	EMF 1:100.000 - PD - SENSOR TM - 3,4,5	X	X	X
IDENTIF DE ÁREAS IRRIGADAS	EMF 1:100.000 - PD - SENSOR TM - 3,4,5	X	X	X
IDENTIF DE ÁREAS OCUP PINUS E EUC.	EMF 1:100.000 - PD - SENSOR TM - 3,4,5	X	X	X
IDENTIF DE DUNAS	EMF 1:100.000 - PD - SENSOR TM - 3,4,5	X	X	X
ESTUDO DE ASSOREAMENTO	EMF 1:100.000 - PD - SENSOR TM - 4	X	X	X

1- SENSOR TM - ESCALA 1:100.000 - 5
2- SENSOR TM - 3,4,5
3- SENSOR HRV - XS1,XS2,XS3
4- SENSOR HRV - 3,4,5
5- SENSOR TM - 1,2,3
6- SENSOR HRV-XS1,XS2,XS3

5.4 ANÁLISE COMPARATIVA

Fazendo-se uma análise comparativa podemos observar que as respostas dadas pelo Sistema Especialista teve um acerto de **95,23% frente ao ESPECIALISTA-01**; **90,47% frente ao ESPECIALISTA-02**; e **85,71% frente ao ESPECIALISTA-03**.

A resposta do Sistema Especialista para a Aplicação em Áreas de Reflorestamento recomenda para Escala Mapa Final 1:100.000 - Produto Fotográfico - Sensor TM - Escala 1:100.000 - banda 3 enquanto que para o ESPECIALISTA-01 a resposta é Sensor TM - Escala 1:100.000 - banda 5.

Para a Aplicação em Estudos de Sedimentos em Suspensão em Corpos D´Água o SE recomenda para Escala Mapa Final 1:50.000 - Produto Digital - Sensor TM - bandas 1,2,3 enquanto que para o ESPECIALISTA-02 a resposta é Sensor TM - bandas 3,4,5.

Em Aplicação de Mapeamento da Rede de Drenagem o SE recomenda para Escala Mapa Final 1:50.000 - Produto Fotográfico - Sensor TM - banda 4 e EMF 1:50.000 - Produto Fotográfico - Sensor HRV - banda XS3 para o ESPECIALISTA-03 a resposta é Sensor HRV - bandas XS1, XS2, XS3.

Na Aplicação de Monitoramento da Qualidade da Água o SE recomenda para Escala Mapa Final 1:50.000 - Produto Digital - Sensor TM - banda 1,2,4 para o ESPECIALISTA-02 a resposta é Sensor HRV - bandas 3,4,5.

Para Aplicação de Estudo da Erosão o SE recomenda para Escala Mapa Final - 1:50.000 - Produto Digital - Sensor TM - bandas 3,4,7 para o ESPECIALISTA-02 a resposta é Sensor TM - bandas 1,2,3.

Em Aplicação de Tipos de Solo o SE recomenda para Escala Mapa Final - 1:100.000 - Produto Digital - Sensor TM - banda 3,4,7 para o ESPECIALISTA-03 a resposta é Sensor HRV - bandas XS1, XS2, XS3.

5.6 CONCLUSÃO

O Sistema Especialista desenvolvido cumpre os seus objetivos de maneira bastante consistente, auxiliando o usuário na especificação de produto imagem e satélite. As repostas divergentes dadas pelos especialistas, em contato posterior com esses, pudemos observar que tais respostas podemos qualificá-las como uma nova opção que deve ser incorporada à base de conhecimento não sendo uma resposta errada do Sistema Especialista e sim o enriquecimento do conhecimento.

6.0 CONCLUSÕES E RECOMENDAÇÕES PARA FUTURAS PESQUISAS

6.1 CONCLUSÕES

A maioria dos esforços da Inteligência Aplicada estão nos laboratórios de pesquisa. Contudo, uma coleção de técnicas de IA que capacitam os computadores a ajudarem as pessoas a analisar problemas e atomar decisões, chamados Sistemas Especialistas Baseados em Conhecimento, comprovou recentemente o seu valor e numerosas aplicações comerciais estão a caminho. Os sistemas especialistas estão sendo desenvolvidos para ajudar os administradores nas complexas tarefas de planejar e esquematizar, diagnosticar doenças, localizar depósitos minerais, configurar equipamentos de computador complexos e auxiliar a mecânica a consertar problemas de locomotivas.

Os sistemas especialistas vão mudar a forma como as empresas operam, alterando a maneira de pensar das pessoas sobre a resolução de problemas. Esta nova tecnologia tornará possível criar respostas rápidas, pragmáticas, para ampla faixa de problemas que desafiam presentemente qualquer solução efetiva.

A utilização de técnicas de Inteligência Artificial, como Sistema Especialistas para a especificação de produtos de imagens de Satélites e a utilização adequada dos produtos obtidos pelos diferentes sensores faz-se

necessário como ferrramenta para especificar um produto de imagem, dado um conjunto de recursos disponíveis, ou seja, saber qual o produto a ser utilizado para extrair a informação desejada da imagem gerada para que de fato mais e mais pessoas leigas em Sensoriamento Remoto desfrutem das vantagens de utilização de imagens de satélite. Pois, apesar da falta de conhecimento técnico a respeito desta ferramenta dificultar sua maior disseminação, o uso de imagens de satélites tem aumentado ano a ano.

O sensoriamento remoto da terra com satélites e aeronaves está intimamente relacionado e dependente da tecnologia em computação com processamento de dados para analise de imagens digitais desde 1960. Isso demonstra o quão é importante o uso de Sistema Especialista em problemas de sensoriamento remoto.

6.2 RECOMENDAÇÕES PARA FUTURAS PESQUISAS

As sugestões para continuidade deste trabalho, em primeiro lugar, envolvem basicamente mudanças na interface, base de conhecimento e mecanismo de inferência do tipo:

- ser integrado ao sistema especialista um editor para que a base de conhecimento fosse facilmente alterada pelo usuário. Desta forma, não seria mais necessário utilizar um editor separado do sistema especialista, tornando o uso da ferramenta mais descomplicado.

- ser acrescentado na escolha de cada premissa uma opção de Ajuda para que o usuário possa receber a informação necessária para auxiliá-lo na especificação do produto.

- o Sistema Especialista pode apresentar o caminho percorrido e as regras disparadas, mostrando-as.

- implementar o metodo de encadeamento reverso.

Outras sugestões que modificariam a estrutura do Sistema Especialista seria a implementação de redes neurais buscando fornecer ao sistema capacidade de aprendizado.

BIBLIOGRAFIA

AMARAL, G.; MOREIRA, J.C. Interpretação Automática de Imagens Landsat para Exploração Mineral na Região Amazônia. I Simpósio Brasileiro de Sensoriamento Remoto. São José dos Campos, 1978, p.677-685.

AKSIT, Mehmet & BERGMANS, Lodewijk. Obstacles in object oriented development. *in* OOPSLA '92. pp. 341-358. Estados Unidos, Association for Computing Machinery, 1992.

BAILIN, S. C. An object-oriented requirements specification method. in Communications of ACM. Volume 32, n° 5, pp. 608-623. Estados Unidos, Association for Computing Machinery, 1989.

BARR, A., FEIGENBAUM, E.A. (Eds.), "The Handbook of Artificial Intelligence". Volume 1, Los Altos, Calif.: Morgan Kaufmann, 1981.

BARR, A., FEIGENBAUM, E.A. (Eds.), "The Handbook of Artificial Intelligence". Volume 2, Los Altos, Calif.: Morgan Kaufmann, 1982.

BARR, A., FEIGENBAUM, E.A. (Eds.), "The Handbook of Artificial Intelligence". Volume 3, Los Altos, Calif.: Morgan Kaufmann, 1982.

BECK, Kent & CUNNINGHAM, Ward. A laboratory for teaching object-oriented thinking. in OOPSLA '89. Estados Unidos, Association for Computing Machinery, 1989.

BOOCH, Grady. Object-oriented design with applications. The Benjamin/Cummings Publishing Company Inc., 1991.

BOOCH, Grady. Object-oriented design with applications. The Benjamin/Cummings Publishing Company Inc., 1991.

BOOCH, Grady. What is and what isn't object-oriented design?. *in* American Programmer. Volume 2, n° 7-8, pp. 14-21. Estados Unidos, 1989.

BROOKS, F. The mithical man-month. Estados Unidos, Addison-Wesley, 1975.

COAD, Peter e YOURDON, Edward . Projeto baseados em objetos . Editora Campus , Série Yourdon Press, 1ª Edição. Rio de Janeiro - RJ, 1993.

COX, Brad. Object-oriented programming. Estados Unidos, Addison-Wesley, 1986.

DeMARCO, Tom. Structured analysis and system specification. Estados Unidos, Yourdon Inc., 1978.

DONY, Christophe & MALENFANT, Jacques & COINTE, Pierre. Prototype-based languages: from a new taxonomy to constructive proposals and their validation. in OOPSLA '92. pp. 201-217. Estados Unidos, Association for Computing Machinery, 1992.

FAUST, Richard. Contribuições da semiótica e psicologia cognitiva à construção de software: uma investigação. Projeto de bacharelado em Ciências da Computação pela UFSC. Florianópolis, 1992.FICHMAN, Robert G. & KEMERER, Chris F. Object-oriented and conventional analysis and design methodologies. *in* Computer. Outubro/92, pp. 22-39. Estados Unidos, IEEE, 1992.

HEINY, Loren. Programação gráfica para Windows com Borland C++. Livros Técnicos e Científicos Ed.. 1ª Edição. Rio de Janeiro , 1993.

PAPPAS, Chris H. & MURRAY, William H.. Borland C++ 4.0. Makron Books do Brasil Editora Ltda. São Paulo, 1995

RICH, Elaine. Inteligência artificial. McGraw-Hill, São Paulo, 1988.

RUMBAUGH, J. *et alli*. Object-oriented modeling and design. Estados Unidos, Prentice Hall, 1991.

SENNE, Edson Luiz França; MONTEIRO, Antônio Miguel V.; BINS , Leonardo S. ; JUNIOR, Antônio F. ; VELASCO, Flávio R. D. Um sistema especialista para especificação de produtos de imagens de satélite. Texto - Base do Projeto ESTRA, da SID-Informática - Instituto de Pesquisas Espaciais - INPE . Guaratinguetá e São José dos Campos - SP, 1987.

SHLAER, Sally & MELLOR, Steve J. Object-oriented analysis: modeling the world in data. Estados Unidos, Yourdon Press, 1988.

WANGLER, Michael F. & HANSEN, Peeter. Visualizing objects: methods for exploring human-computer interactions concepts. in OOPSLA '92. pp. 146-153. Estados Unidos, Association for Computing Machinery, 1992.

WASSERMAN, A. I. & PIRCHER, P. A. & MULLER, R. J. An object-oriented structured design method for code generation. in Software Eng. Notes. Volume 14, n° 1, pp. 32-55. Estados Unidos, 1989.

WATERMAN, D.A. A guide to expert systems. Reading, MA, Addison Wesley, 1986.

WINBLAD, Ann L. e EDWARDS, Samuel e KING, David R. Software Orientado ao Objeto. Makron Books dos Brasil Editora Ltda. 1ª Edição. São Paulo - SP, 1993.

WIRFS-BROCK, R. & WILKERSON, B. & WIENER, L. Designing object-oriented software. Estados Unidos, Prentice Hall, 1990.

YOURDON, Edward & CONSTANTINE, L. Structured design: fundamentals of a discipline of computer programming and design. 2ª edição. Estados Unidos, Prentice Hall, 1979.

YOURDON, Edward & CONSTANTINE, L. Structured design: fundamentals of a discipline of computer programming and design. 2ª edição. Estados Unidos, Prentice Hall, 1979.

YOURDON, Edward. Modern structured analysis. Estados Unidos, Yourdon Press, 1989.

YOURDON, Edward. Modern structured analysis. Estados Unidos, Yourdon Press, 1989.

YOURDON, Edward. Object-oriented obeservations. *in* American Programmer. Volume 2, n° 7-8, pp. 3-7. Estados Unidos, 1989.

YOURDON, Edward. Object-oriented obeservations. in American Programmer. Volume 2, n° 7-8, pp. 3-7. Estados Unidos, 1989.

ZIELINSKI, Roman. OOTher: OO Documentation Tool. Manual do usuário. Documento em mídia eletrônica. Suécia, 1992.

Referências bibliográficas

[AMA78] AMARAL, G.; MOREIRA, J.C. Interpretação Automática de Imagens Landsat para Exploração Mineral na Região Amazônia. I Simpósio Brasileiro de Sensoriamento Remoto. São José dos Campos, 1978, p.677-685.

[BBB83] BUCHANAN, B. G., BARSTOW, D., BECHTAL, R., et alii, "Constructing an expert system". In Building Expert Systems, Hayes-Roth, F., Waterman, D. A., Lenat, D. (Eds.), Reading, Mass.: Addison-Wesley, p.127-167, 1983.

[COA92] COAD, Peter & YOURDON, Edward. Análise baseada em objetos. 2ª ed. Rio de Janeiro, Campus, 1992.

[COA93] COAD, Peter & YOURDON, Edward. Projeto baseado em objetos. Rio de Janeiro, Campus, 1993.

[COU87] COULSON, R. N.; FOLSE, L. J.; LOH, D. K. Artificial Intelligence and natural resource management. Science, 237: 262-267, july 1987.

[FAI85] FAIRLEY, Richard E. Software engeneering concepts. Estados Unidos, McGraw-Hill, 1985.MARTIN, J. Information eng., books I, II e III. Estados Unidos, Prentice Hall, 1990.

[FRA93] FRANZONI, ANA M. B. Aplicação do Sensoriamento Remoto no Monitoramento de Áreas Sujeitas a Degradação Ambiental: o Caso da Bacia Hidrográfica do Sangão - SC. Dissertação de Mestrado em Geografia, Área de Concentração: Utilização e Conservação de Recursos Naturais - UFSC. Florianópolis, S.C., 1993.

[GEN88] GENARO, Sergio. Sistemas Especialistas, o conhecimento artificial. São Paulo, 1988.

[GOO87] GOODENOUGH, D. G.; GOLDBERG, M.; PLUNKETT, G,: ZELEK, J. An expert system for remote sensing. IEEE Transactions on Geoscience and Remote Sensing, GE-25(3): 349-359, May 1987.

[GUN78] GUNTHER, R. Managment methodology for software product. Estados Unidos, Wiley Interscience,1978.

[HAR88] HARMON, Paul & KING, David Sistemas Especialistas. Editora Campus - Rio de Janeiro : Campus, 1988.

[HAY83] HAYES-ROTH, F.; WATERMAN, D. A.; LENAT, D. B. Building Expert Systems, Addison-Wesley, 1983.

[HAY84] HAYES-ROTH, F., "The Konowledge-Based Expert System: A Tutorial". IEEE Computer, 17, 9, p.11-28, 1984.

[HOF87] HOFFMAN, R.R., "The problem of Extracting the Knowledge from the Perspective of Experimental psychology". AI Magazine, p.53-66, Summer, 1987.

[LAP94] LAPOLLI, Édis M..Processamento de Imagens Digitais: Uma Abordagem Utilizando Conjunto Difusos. Tese de Doutorado em Engenharia da Produção na Universidade Federal de Santa Catarina. Santa Catarina, 1994.

[MAT87] MATSUYAMA, T. Knowledge-based aerial image understanding systems and expert systems for image processing. IEEE Transactions on Geoscience and Remote Sensing, GE-25(3): 305-316, May 1987.

[MCK87] MCKEOWN, D. M. The role of Artificial Intelligence in the integration of remotely sensed data with Geographic Information Systems, . IEEE Transactions on Geoscience and Remote Sensing, GE-25(3): 330-348, May 1987.

[MEY88] MEYER, Bertrand. Object-oriented software construction. Reino Unido, Prentice Hall, 1988.

[MSC92] Microsoft Windows software development kit - user guide. Estados Unidos, Microsoft Press, 1991.

[NOV89] NOVO, E. L. de M..Sensoriamento Remoto - Princípios e Aplicações, Ed. Edgard Blücher Ltda, São Paulo - SP. 308 p., 1989.

[PAC94] PACHECO, Roberto & MONTENEGRO, Fernando. Orientação a objetos em C++. Editora Ciência Moderna Ltda, Rio de Janeiro, 1994.

[PIN91] PINTO, S. A. F., (1991). Sensoriamento Remoto e Integração de Dados Aplicados no Estudo da Erosão dos Solos: Contribuição Metodológica. Tese de Doutorado em Ciências/Geografia Física, INPE, São José dos Campos - SP. (INPE - 5311 - TAE/09), 134p.

[RUM91] RUMBAUGH, J. et alli. Object-oriented modeling and design. Estados Unidos, Prentice Hall, 1991.

[SIM93] SIMON, H.A. in Byte Brasil. Editora Voice Ltda. p. 50-54. São Paulo, Outubro 1993.[SCH89] SCHILDT, Herbert. Inteligência artificial utilizando linguagem C. McGraw-hill, São Paulo, 1989.

[STE81] STEFFEN, C.A. et al. Sensoriamento Remoto: Princípios Físicos; Sensores e Produtos, e Sistemas LANDSAT, São José dos Campos-SP. (INPE-2226-MD/013).81p., 1981.

[STE82] STEFIK, M.; AIKINS, J.; BALZER, R.; BENOIT, J.; BIRNBAUM, L.; HAYES-ROTH, F.; SACERDOTI, E. The organization of expert systems: a tutorial. Artificial Intelligence, 18(2): 135-173, Mar. 1982.

[TAI86] TAILOR, A.; CROSS, A.; HOGG, D. C.; MASON, D. C., Knowledge-based interpretation of remotely sensed images, Image and Vision Computin, 4(2): 67-83, Mai. 1986.

[WHA87] WHARTON, S. W. A spectral-knowledge-based approach for urban landcover discrimination. IEEE Transactions on Geoscience and Remote Sensing. GE-25(3): 272-282, May 1987.

[WIE91] WIENER, Richard S. & PINSON, Lewis J. C++: Programação Orientada para Objeto: Manual Prático e Profissional , Makron, McGraw-Hill, São Paulo, 1991.

[YEE87] YEE, B. An expert system for planimetric feature extraction. IGARS'87 Sysmposium. Proceedings. Ann Arbor, p. 321-325, May 1987.

Anexo I - Resumo da metodologia OOA

Neste anexo são apresentadas, de maneira resumida, as notações e estratégias da OOA - *Object Oriented Analysis*, metodologia de análise orientada a objetos proposta por Peter Coad e Edward Yourdon, em [COA92], e suportada pelo SISTEMA ESPECIALISTA. Este anexo foi elaborado baseando-se no Apêndice A da referida obra.

Notações

Nas figuras seguintes estão resumidas as notações empregadas pela OOA na definição classes e objetos, estruturas e conexões de ocorrências de um modelo.

Classes e objetos

Na FIGURA 4.5 estão representados os símbolos utilizados pela OOA para a representação de classes e objetos.

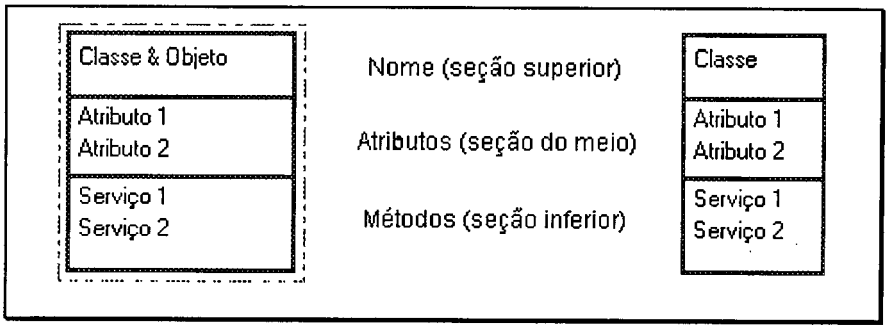

FIGURA 4.5 - Símbolos de classes e objetos da OOA

Em OOA, o símbolo de "Classe & Objeto" (à esquerda na FIGURA 4.5) é utlizado para representar classes que podem ser instanciadas diretamente, enquanto que o

símbolo de "Classe" (à direita na figura) é utilizado para classes abstratas, que não podem ter instâncias.

Estruturas

Um diagrama OOA modela estruturas de Generalização-Especialização (Gen-Spec) e Todo-Parte. Os símbolos que representam estas estruturas estão na FIGURA 4.6.

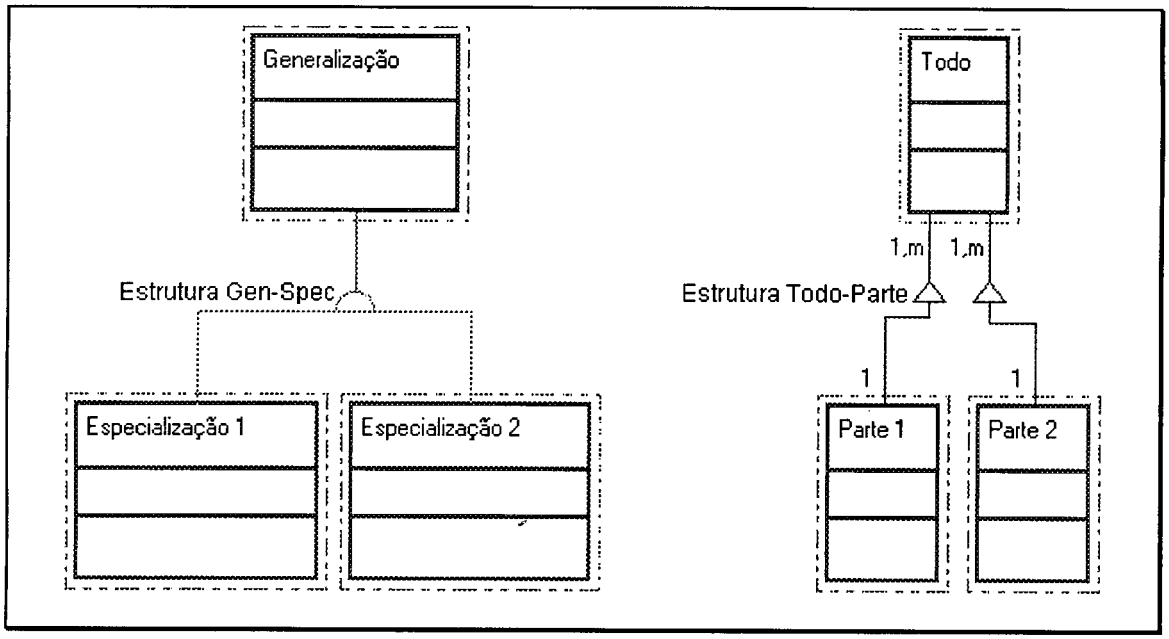

FIGURA 4.6 - Símbolos de estruturas da OOA

Conexões de ocorrência

O símbolo utilizado para modelar uma Conexão de Ocorrência em um modelo OOA está na FIGURA 4.7.

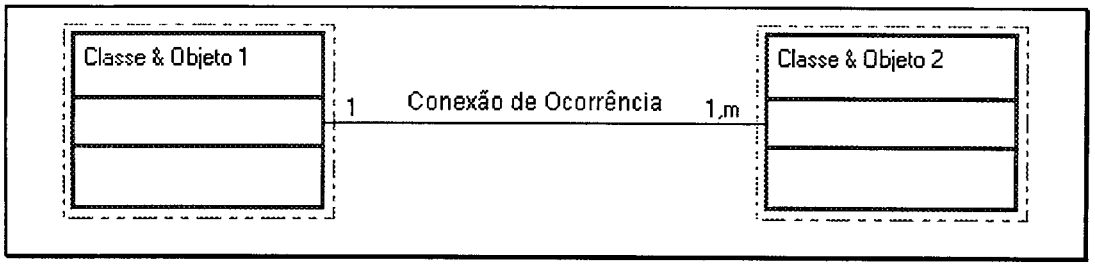

FIGURA 4.7- Símbolo de Conexão de Ocorrência da OOA

Estratégias

Coad-Yourdon propõem uma série de estratégias para identificação de classes e objetos, identificação de estruturas, identificação de conexões de ocorrência, identificação de assuntos, definição de atributos e definição de serviços. Estas estratégias estão resumidas a seguir.

Identificação de classes e objetos

Segundo [COA92]:

*"**Objeto** é uma abstração de alguma coisa em um domínio de problemas, exprimindo as capacidades de um sistema de manter informações sobre ela, interagir com ela ou ambos; um encapsulamento de valores de atributos e de seus serviços exclusivos.*

***Classe** é uma descrição de um ou mais objetos por meio de um conjunto uniforme de atributos e serviços, incluindo uma descrição de como criar novos objetos na classe.*

***Classe e objeto** significa uma classe e objetos desta classe.*

Como nomear: atribua um nome único ou adjetivo e nome; descreva um objeto único na classe; use o vocabulário padronizado para o domínio do problema.

Onde procurar: faça uma observação inicial; ouça atentamente; verifique o resultado de OOAs anteriores; outros sistemas; leia, leia, leia; faça protótipos.

O que procurar: procure por estruturas, outros sistemas, dispositivos, coisas ou eventos lembrados, papéis executados, procedimentos operacionais, locais e unidades organizacionais.

O que considerar e recusar: lembranças necessárias, processamentos necessários, atributos múltiplos (normalmente), mais de um objeto em uma classe (normalmente), atributos sempre aplicáveis, serviços sempre aplicáveis, requisitos baseados em domínio e resultados não simplesmente derivados."

Identificação de estruturas

Conforme [COA92]:

*"**Estrutura** é uma expressão da complexidade do domínio do problema pertencente às responsabilidades do sistema. O termo "estrutura" é usado como um termo global, podendo ser usado para estruturas Gen-Spec e Todo-Parte.*

Estrutura Gen-Spec

Considere cada classe como uma generalização. Para suas especializações potenciais, pergunte: Ela está no domínio do problema? Ela faz parte das responsabilidades do sistema? Haverá herança? As especializações atenderão ao critério "o que considerar e recusar" para classes e objetos?

Além disso, de uma forma semelhante, considere cada classe como uma especialização. Para suas generalizações potenciais, faça as mesmas perguntas.

Verifique o resultado de OOAs anteriores em domínios de problemas iguais e semelhantes.

Se forem possíveis muitas especializações, considere a especialização mais simples e a mais elaborada, e depois prossiga com as várias especializações intermediárias.

A forma mais comum de uma estrutura Gen-Spec é uma hierarquia Gen-Spec. Contudo, um entrelaçamento pode ser usado para realçar especializações adicionais, capturar explicitamente elementos comuns ou aumentar apenas modestamente a complexidade do modelo.

Se uma estutura de entrelaçamento tornar-se confusa, reorganize parte dela como uma hierarquia. A hierarquia pode ser mais clara e facilitar a comunicação do domínio do problema e das responsabilidades do sistema.

Evite dar nomes conflitantes em um entrelaçamento. Caso contrário, uma especialização que herda nomes conflitantes terá que incluí-los e depois selecionar o que é necessário para a especificação correspondente.

Estrutura Todo-Parte

O que procurar: considere estas variações - montagem-partes, recipiente-conteúdos e conjunto-membros.

Verifique resultados de OOAs anteriores em domínios de problemas iguais e semelhantes.

O que considerar e recusar: considera cada objeto como um todo. Para cada uma das suas partes potenciais, pergunte: Ela está no domínio do problema? Ela faz parte das responsabilidades do sistema? Ela captura mais do que apenas um valor de status? Se não, inclua apenas um atributo

para ela no todo. Ela proporciona uma abstração útil no tratamento com o domínio do problema?

Além disso, de uma forma similar, considere cada objeto como uma parte Para cada todo potencial, faça estas mesmas perguntas."

Identificação de conexões de ocorrência

Segundo [COA92]:

*"**Conexão de Ocorrência** é um modelo(s) do mapeamento(s) do domínio do problema que um objeto precisa ter com outros objetos, para cumprir suas responsabilidades.*

Verifique o resultado de OOAs anteriores em domínios de problemas iguais e semelhantes.

Para cada objeto, adicione linhas de conexão.

Adicione mapeamentos entre os objetos desntro do contexto em pauta, prestando atenção no posicionamento das conexões nas estruturas Gen-Spec.

Para cada objeto, defina a quantidade ou intervalo.

Limite inferior: conexão opcional? O limite inferior é igual a 0. Conexão obrigatória? O limite inferior é igual a 1 ou mais.

Limite superior: conexão única? O limite superior é igual a 1. Conexões múltiplas? O limite superior é maior que 1.

Também faça especificações adicionais para estruturas Todo-Parte (a diferença entre a estrutura Todo-Parte e a Conexão de Ocorrência é a força semântica básica)."

Identificação de assuntos

Conforme [COA92]:

*"**Assunto** é um mecanismo para orientar um leitor (analista, especialista do domínio do problema, gerente, cliente) em um modelo amplo e complexo. Os assuntos também são úteis para organização de pacotes de trabalho em projetos extensos, conforme as investigações iniciais de OOA.*

Como selecionar: transforme a classe mais superior em cada estrutura em um assunto. Depois, transforme cada classe e objeto não pertencente a uma estutura em um assunto. Além disso, verifique o resultado de OOAs anteriores em domínios de problemas iguais e semelhantes.

Como aperfeiçoar: aperfeiçoe os assuntos usando subdomínios de problemas.

Como construir: no nível de assunto, desenhe cada assunto como uma caixa retangular simples, com um nome e número de assunto dentro dela. Opcionalmente, liste também as classes incluídas no assunto.

Em outros níveis, indique os assuntos com caixas rotuladas de partição de assunto, para guiar o leitor de um assunto a outro.

Para um modelo maior, considere a impressão de um conjunto separado de diagramas para cada assunto, mostrando assuntos relacionados "quebrados" em cada conjunto de diagramas, conforme for necessário para facilitar a comunicação global.

Os assuntos podem ser encarados como quebrados, parcialmente expandidos (listando-se suas classes e objetos) ou totalmente expandidos (caixas de particionamento de assunto, colocadas juntamente com outros níveis de OOA).

Uma classe e objeto pode estar em mais de um assunto (quando isto ajudar na orientação do leitor.

Os assuntos podem conter outros assuntos, resultando em um mapa multiníveis para orientar o leitor em um modelo extenso.

Quando adicionar: faça uma adição sempre que um mapa global for necessário para guiar os vários leitores pelo modelo."

Definição de atributos

Segundo [COA92]:

*"**Atributo** é um dado (informação de estado) para cada objeto em uma classe tem seu prórpio valor.*

Identifique os atributos

Perguntas a serem feitas: Como sou descrito em geral? Como sou descrito neste domínio de problema? Como sou descrito no contexto das responsabilidades do sistema? O que preciso saber? Da quais informações de estado preciso sempre me lembrar? Quais podem ser meus estados?

Verifique o resultado de OOAs anteriores em domínios de problemas iguais e semelhantes.

Faça cada atributo capturar um "conceito atômico": um valor único ou um agrupamento de valores fortemente relacionados.

Decidir se um atributo sempre recelculável será usado é uma decisão de projeto- tempo versus memória. Na OOA, basta especificar o serviço de cálculo, e não um atributo recalculável correspondente.

(...)

Posicione os atributos

Coloque cada atributo na classe e objeto que ele descreve melhor (verifique o dom/inio do problema).

Aplique herança em estruturas Gen-Spec. Posicione os atributos mais gerais no nível mais alto. Posicione os atributos mais especializados no nível mais baixo.

(...)

Verifique casos especiais

Verifique, para cada atributo, se existe um valor não-aplicável. Verifique cada classe e objeto com apenas um atributo. Verifique os valores de repeticão para cada atributo.

(...)

Especifique os atributos

Nome: vocabulário padronizado. Reflete o domínio do problema e as responsabilidades do sistema. Legível. Procure ser mais abrangente e menos específico.

Descrição.

Especificações.

Considere que uma especificação pode simplificar o trabalho de especificação de serviço. Para usar uma especificação, leve em conta o fator custo versus benefício.

Unidade de medida, intervalo, limite, enumeração, valor default, precisão.

Especificação de criação ou acesso?

Existem especificações devido a outros valores de atributo?"

Definição de serviços

Conforme [COA92]:

*"**Serviço** é um comportamento especifico que um objeto deve exibir.*

Identifique os estados de objeto

Examine os valor potenciais para os atributos.

Determine se as responsabilidades do sistema incluem um comportamento diferente para estes valores potenciais.

Verifique resultados de OOAs anteriores em domínios de problemas iguais e semelhantes.

(...)

Identifique os serviços requeridos

Serviços algoritmicamente simples: criar - cria e inicia um novo objeto em uma classe; conectar - conecta (desconecta) um objeto a outro; acessar - obtém ou estabelece os valores dos atributos de um objeto; liberar - libera (desconecta e elimina) um objeto.

Serviços algoritmicamente complexos.

Verifique resultados de OOAs anteriores em domínios de problemas iguais e semelhantes.

Duas categorias: calcular - calcula um resultado a partir dos valores de atributo de um objeto - e monitorar - monitora um sistema ou dispositivo externo; lida com entradas e saídas externas ao sistema ou com controle e aquisição de dados de dispositivos; e pode precisar de alguns serviços associados, como iniciar ou terminar.

Pergunte: O objeto é responsável por executar quais cálculos em seus valores? O objeto é responsável por fazer qual monitoração, para detectar e responder uma alteração em um sistema ou dispositivo externo (comportamento requerido de resposta a evento)?

Use nomes de serviço específicos do domínio."

Anexo II - Código-fonte do SISTEMA ESPECIALISTA

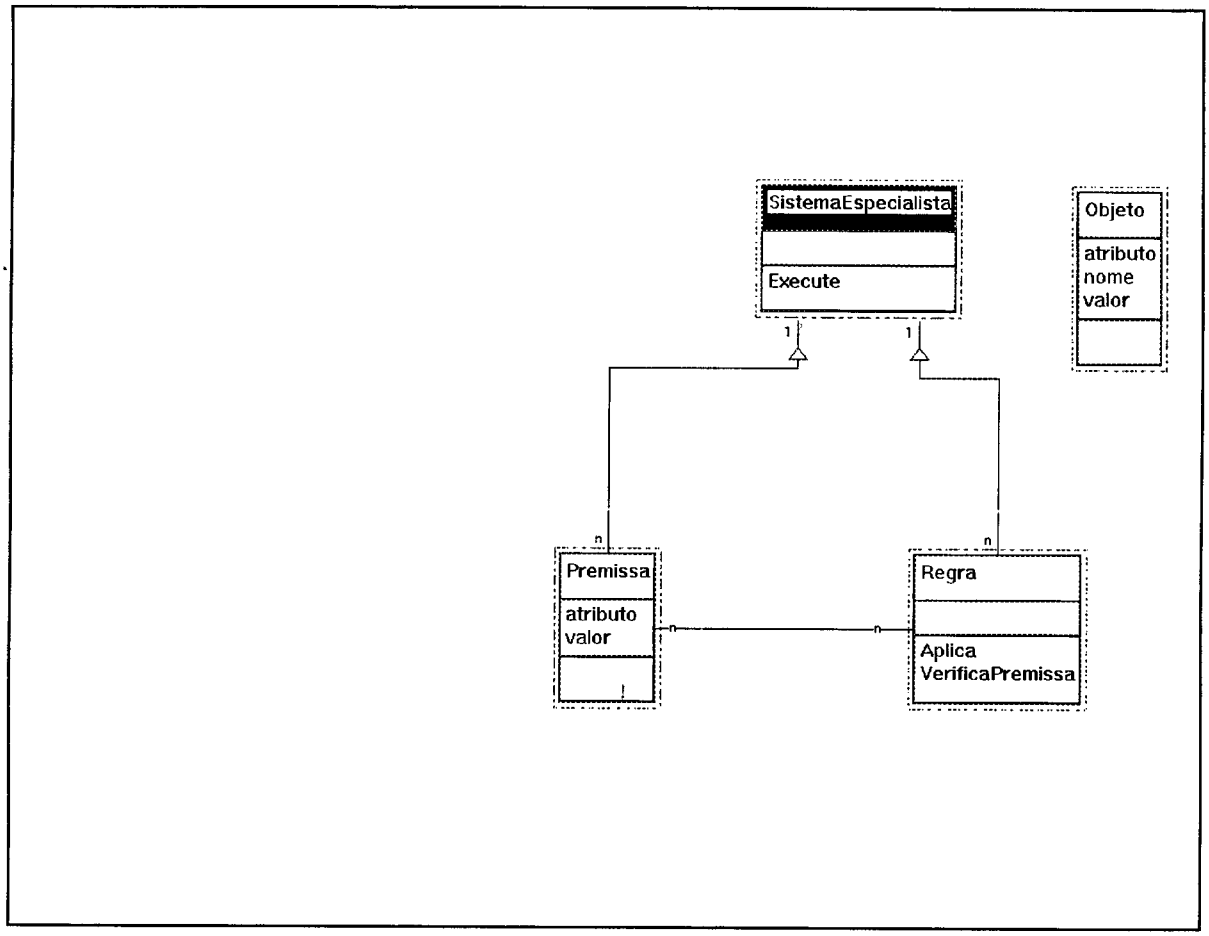

FIGURA 4.8 Modelagem SISTEMA ESPECIALISTA

Neste anexo estão contidas as listagens do código-fonte das classes implementadas no SISTEMA ESPECIALISTA: classes o SISTEMA ESPECIALISTA e classes de *interface*.

Classes do SISTEMA ESPECIALISTA

Esta seção contém as listagens do código-fonte das classes que compõem o SISTEMA ESPECIALISTA, tais como definidas na **Modelagem.**

Classe "ATRIBUTO"

Esta classe representa um atributo de uma classe .

Declaração (arquivo ATTR.H):

```
#ifndef __ATRIBUTO_H
#define __ATRIBUTO_H

#include <_defs.h>
#include <cstring.h>
#include <classlib\sets.h>

#include <char.h>

_CLASSDEF(Atributo)

class Atributo {
  public:
    Atributo(PCCHAR _nome = "",PCCHAR _valor = "");
    Atributo(RCAtributo);
    virtual ~Atributo();

    int operator==(RCAtributo) const;

    const string &Nome() const { return nome; }
    void Nome(PCCHAR _nome) { nome = _nome; }

    const string &Valor() const { return valor; }
    void Valor(PCCHAR _valor) { valor = _valor; }

  protected:

  private:
    string nome,valor;
}; // Atributo

//---------------------------------------------------------------------------------------------

typedef TISetAsVector<Atributo> AtributosSet;
typedef TISetAsVectorIterator<Atributo> AtributosSetIterator;

//---------------------------------------------------------------------------------------------

#endif __ATRIBUTO_H
```

Definição (arquivo ATTR.CPP):

```
#include "classes\atrib.h"

//---------------------------------------------------------------------------------------------

Atributo::Atributo(PCCHAR _nome,PCCHAR _valor) : nome(_nome),valor(_valor)
```

```
{
} // Atributo()

//---------------------------------------------------------------------------

Atributo::Atributo(RCAtributo outroAtributo) : nome(outroAtributo.nome),valor(
  outroAtributo.valor)
{
} // Atributo(RCAtributo)

//---------------------------------------------------------------------------

Atributo::~Atributo()
{
} // ~Atributo

//---------------------------------------------------------------------------

int Atributo::operator==(RCAtributo outroAtributo) const
{
  return nome == outroAtributo.nome && valor == outroAtributo.valor;
} // operator==
```

Classe "SISTEMA ESPECIALISTA"

Esta classe representa o sistema computacional.

Declaração (arquivoSISTESP.H):

```
#ifndef __SISTESP_H
#define __SISTESP_H

#include <_defs.h>

#include <char.h>

#include "classes\cclusao.h"
#include "classes\objeto.h"
#include "classes\premissa.h"
#include "classes\regra.h"

_CLASSDEF(SistemaEspecialista)

class SistemaEspecialista {
  public:
    SistemaEspecialista();
    SistemaEspecialista(RCSistemaEspecialista);
    virtual ~SistemaEspecialista();

    int operator==(RCSistemaEspecialista) const;

    void Executa(PCCHAR _nomeArquivo,PCHAR _arquivoSaida);
    void ReExecuta(PCHAR _arquivoSaida);

  protected:

  private:
    void AplicaAlgoritmo();
    char nomeArquivo[256];
    PCHAR arquivoSaida;

    PremissasArray premissas;
    RegrasSet regras,estoqueRegras;
    ConclusoesSet conclusoes;
    ObjetosBag objetos;

    BOOL CarregaArquivo(PCCHAR nomeArquivo);
    void EsvaziaEstruturas();
      // esvazia os conjuntos de premissas, regra e conclusoes
    BOOL MontaConjunto(WordSet &conjunto,PCHAR numeros);
      // coloca no conjunto os números contidos na string. Os números devem estar separados por
```

```
      // virgulas na string, que deve ser terminada também por virgula

    BOOL GeraConclusoes();
    void ValidaPremissa(WORD premissa);
      // percorre as regras e confirma a premissa
    void InvalidaPremissasComAtributo(RCPremissa premissa);
      // invalida todas as premissas que tenham o mesmo atributo de 'premissa' e um valor
      // diferente
    void AplicaRegrasPassiveis();
      // aplica as regras cujas premissas já tenham todas sido confirmadas e elimina-as do
      // conjunto de regras
    void EliminaRegrasDependentesDe(WORD premissa);
      // elimina do conjunto de regras as regras dependentes da premissa e põe estas regras no
      // estoque, para serem usadas depois

    BOOL GeraObjetos();
    BOOL ExisteObjeto(const string &nomeObjeto);
      // procura no conjunto de objetos um determinado objeto
    void SelecionaObjetos(const string &nomeObjeto,ObjetosQueue &selecionados);
      // coloca na lista os objetos com determinado nome
    BOOL ExisteAtributo(RObjeto,const string &nomeAtributo);
      // procura no conjunto de atributos do objeto um determinado atributo
    RAtributo SelecionaAtributo(RObjeto,const string &nomeAtributo);
      // seleciona do conjunto de atributos do objeto o atributo com determinado nome

    BOOL GeraArquivoObjetos(PCCHAR nomeArquivo);
}; // SistemaEspecialista

#endif __SISTESP_H
```

Definição (arquivo SISTESP.CPP):

```
#include "classes\sistesp.h"

#include <fstream.h>
#include <string.h>
#include <strstrea.h>
#include <classlib\queues.h>

#include <buffers.h>
#include <checks2.h>
#include <windows2.h>

//-----------------------------------------------------------------------------------------

typedef TIQueueAsVector<Regra> RegrasQueue;
typedef TIQueueAsVector<Objeto> ObjetosQueue;

//-----------------------------------------------------------------------------------------

SistemaEspecialista::SistemaEspecialista() : premissas(10,1,10)
{
  premissas.OwnsElements(TRUE);
  regras.OwnsElements(TRUE);
  estoqueRegras.OwnsElements(TRUE);
  conclusoes.OwnsElements(TRUE);
  objetos.OwnsElements(TRUE);
} // SistemaEspecialista()

//-----------------------------------------------------------------------------------------

SistemaEspecialista::SistemaEspecialista(RCSistemaEspecialista outroSistemaEspecialista) :
  premissas(outroSistemaEspecialista.premissas.UpperBound(),
  outroSistemaEspecialista.premissas.LowerBound(),10)
{
  for (WORD i = 1;i <= outroSistemaEspecialista.premissas.GetItemsInContainer();i++)
    premissas.Add(new Premissa(*(outroSistemaEspecialista.premissas[i])));

  RegrasSetIterator itRegras(outroSistemaEspecialista.regras);
  while (itRegras)
    regras.Add(new Regra(*itRegras++));

  RegrasSetIterator itEstoqueRegras(outroSistemaEspecialista.estoqueRegras);
```

```
  while (itEstoqueRegras)
    estoqueRegras.Add(new Regra(*itEstoqueRegras++));

  ConclusoesSetIterator itConclusoes(outroSistemaEspecialista.conclusoes);
  while (itConclusoes)
    conclusoes.Add(new Conclusao(*itConclusoes++));

  ObjetosBagIterator itObjetos(outroSistemaEspecialista.objetos);
  while (itObjetos)
    objetos.Add(new Objeto(*itObjetos++));

  premissas.OwnsElements(TRUE);
  regras.OwnsElements(TRUE);
  estoqueRegras.OwnsElements(TRUE);
  conclusoes.OwnsElements(TRUE);
  objetos.OwnsElements(TRUE);
} // SistemaEspecialista(RCSistemaEspecialista)

//---------------------------------------------------------------------------------------------

SistemaEspecialista::~SistemaEspecialista()
{
} // ~SistemaEspecialista

//---------------------------------------------------------------------------------------------

int SistemaEspecialista::operator==(RCSistemaEspecialista) const
{
  return FALSE;
} // operator==

//---------------------------------------------------------------------------------------------

void SistemaEspecialista::Executa(PCCHAR _nomeArquivo,PCHAR _arquivoSaida)
{
  EsvaziaEstruturas();

  strcpy(nomeArquivo,_nomeArquivo);
  arquivoSaida = _arquivoSaida;

  if (!CarregaArquivo(nomeArquivo)) {
    sBuffer.seekp(0);
    sBuffer << "Problemas na leitura do arquivo " << nomeArquivo << "." << ends;
    LoudMessage(NULL,buffer,"Erro",MB_ICONSTOP,NULL);
    return;
  }

  // coloca regras no estoque
  estoqueRegras.Flush();
  RegrasSetIterator it(regras);
  PRegra regra;
  while (it) {
    regra = it++;
    estoqueRegras.Add(new Regra(*regra));
  }

  AplicaAlgoritmo();
} // Executa

//---------------------------------------------------------------------------------------------

void SistemaEspecialista::ReExecuta(PCHAR _arquivoSaida)
{
  arquivoSaida = _arquivoSaida;

  // esvazia estruturas necessárias
  regras.Flush();
  conclusoes.Flush();
  objetos.Flush();

  // "desavalia" as premissas
  PremissasArrayIterator it(premissas);
  while (it) {
    RPremissa p = *it++;
    p.Valida(FALSE);
    p.Avaliada(FALSE);
  }
```

```
  // recupera as regras do estoque
  RegrasSetIterator itRegras(estoqueRegras);
  PRegra regra;
  while (itRegras) {
    regra = itRegras++;
    regras.Add(new Regra(*regra));
  }

  AplicaAlgoritmo();
} // ReExecuta

//---------------------------------------------------------------------------------------------

void SistemaEspecialista::AplicaAlgoritmo()
{
  if (!GeraConclusoes()) {
    LoudMessage(NULL,"Problemas na geração de conclusões.","Erro",MB_ICONSTOP,NULL);
    return;
  }

  sBuffer.seekp(0);
  sBuffer << "Geradas " << conclusoes.GetItemsInContainer() << " conclusões." << ends;
  MessageBox(NULL,buffer,"Atenção",MB_ICONINFORMATION);

  if (!conclusoes.GetItemsInContainer())
    return;

  if (!GeraObjetos()) {
    LoudMessage(NULL,"Problemas na geração dos objetos.","Erro",MB_ICONSTOP,NULL);
    return;
  }

  if (objetos.GetItemsInContainer()) {
    if (GeraArquivoObjetos(nomeArquivo)) {
      sBuffer.seekp(0);
      sBuffer << "Gerado(s) " << objetos.GetItemsInContainer() << " objeto(s). Arquivo gerado."
        << ends;
    } else {
      sBuffer.seekp(0);
      sBuffer << "Gerado(s) " << objetos.GetItemsInContainer() << " objeto(s). Problemas na "
        "geração do arquivo." << ends;
    }
  }

  MessageBox(NULL,buffer,"Atenção",MB_ICONINFORMATION);
} // AplicaAlgoritmo

//---------------------------------------------------------------------------------------------

BOOL SistemaEspecialista::CarregaArquivo(PCCHAR nomeArquivo)
{
  ifstream arquivo(nomeArquivo);
  if (!arquivo.good())
    return FALSE;

  char linha[256];

  // ler premissas
  arquivo.getline(linha,sizeof(linha)); // extrai a linha "PREMISSAS"
  arquivo.getline(linha,sizeof(linha));
  while (strcmp(strupr(linha),"REGRAS")) {
    if (linha[0]) {
      // número
      strcpy(buffer,strtok(linha,";"));
      istrstream numeroStr(buffer);
      WORD numero= 0;
      numeroStr >> (int)numero;
      if (!numeroStr.good() || !numero)
        return FALSE;

      PPremissa premissa = new Premissa;

      // atributo
      strcpy(buffer,strtok(NULL,";"));
      premissa->Atributo(buffer);
```

```
      // valor
      strcpy(buffer,strtok(NULL,"\0"));
      premissa->Valor(buffer);

      premissas.Add(premissa);
    }
    arquivo.getline(linha,sizeof(linha));
  }

  // ler regras
  arquivo.getline(linha,sizeof(linha));
  while (strcmp(strupr(linha),"FIM")) {
    if (linha[0]) {
      // premissas
      char numerosStr[128];
      strcpy(numerosStr,strtok(linha,";"));
      strcat(numerosStr,",");
      WordSet numeros;
      if (!MontaConjunto(numeros,numerosStr))
        return FALSE;

      PRegra regra = new Regra(numeros);

      // objeto
      strcpy(buffer,strtok(NULL,";"));
      regra->Objeto(buffer);

      // atributo
      strcpy(buffer,strtok(NULL,";"));
      regra->Atributo(buffer);

      // valor
      strcpy(buffer,strtok(NULL,"\0"));
      regra->Valor(buffer);

      regras.Add(regra);
    }
    arquivo.getline(linha,sizeof(linha));
  }

  return TRUE;
} // CarregaArquivo

//---------------------------------------------------------------------------

void SistemaEspecialista::EsvaziaEstruturas()
{
  premissas.Flush();
  regras.Flush();
  conclusoes.Flush();
  objetos.Flush();
} // EsvaziaEstruturas

//---------------------------------------------------------------------------

BOOL SistemaEspecialista::MontaConjunto(WordSet &conjunto,PCHAR numeros)
// esta função não deve usar strtok para extrair os números, para não "bagunçar" algum uso de
// strtok externo (como em CarregaArquivo)
{
  while (*numeros) {
    BYTE pos = 0;

    while (*numeros != ',')
      buffer[pos++] = *numeros++;
    buffer[pos] = '\0';

    istrstream numeroStr(buffer);
    WORD numero = 0;
    numeroStr >> (int)numero;
    if (numeroStr.good() && numero)
      conjunto.Add(numero);
    else
      return FALSE;

    numeros++; // consumir ','
  }
```

```
  return TRUE;
} // MontaConjunto

//---------------------------------------------------------------------------

BOOL SistemaEspecialista::GeraConclusoes()
{
  while (regras.GetItemsInContainer()) {
    // escolher uma premissa para perguntar
    RegrasSetIterator *it = new RegrasSetIterator(regras);
    WORD numeroDaPremissa = it->Current()->DependeDe();
    delete it;
    RPremissa premissa = *premissas[numeroDaPremissa];

    if (!premissa.Avaliada()) {
      // fazer a pergunta
      sBuffer.seekp(0);
      sBuffer << "Premissa #" << (int)numeroDaPremissa << ends;
      char titulo[64];
      strcpy(titulo,buffer);

      sBuffer.seekp(0);
      sBuffer << premissa.Atributo() << " = " << premissa.Valor() << "?" << ends;

      switch (MessageBox(NULL,buffer,titulo,MB_ICONQUESTION | MB_YESNOCANCEL)) {
        case IDYES:
          premissa.Valida(TRUE);
          ValidaPremissa(numeroDaPremissa);
          InvalidaPremissasComAtributo(premissa);
          AplicaRegrasPassiveis();
          break;

        case IDNO:
          premissa.Valida(FALSE);
          EliminaRegrasDependentesDe(numeroDaPremissa);
          break;

        case IDCANCEL:
          return TRUE;
      }
    } else {
      if (!premissa.Valida())
        EliminaRegrasDependentesDe(numeroDaPremissa);
    }
  }

  return TRUE;
} // GeraConclusoes

//---------------------------------------------------------------------------

void SistemaEspecialista::ValidaPremissa(WORD premissa)
{
  RegrasSetIterator it(regras);
  while (it) {
    RRegra regra = *it++;
    regra.Confirma(premissa);
  }
} // ValidaPremissa

//---------------------------------------------------------------------------

void SistemaEspecialista::InvalidaPremissasComAtributo(RCPremissa premissa)
{
  PremissasArrayIterator it(premissas);
  while (it) {
    RPremissa p = *it++;
    if (p.Atributo() == premissa.Atributo() && p.Valor() != premissa.Valor())
      p.Valida(FALSE);
  }
} // InvalidaPremissasComAtributo

//---------------------------------------------------------------------------

void SistemaEspecialista::AplicaRegrasPassiveis()
{
  RegrasQueue aplicaveis(100);
```

```
  PRegra regra;

  // selecionar regras
  RegrasSetIterator it(regras);
  while (it) {
    regra = it++;
    if (!regra->QuantPremissas())
      aplicaveis.Put(regra);
  }

  while (aplicaveis.GetItemsInContainer()) {
    regra = aplicaveis.Get();

    // aplicar a regra
    PConclusao conclusao = new Conclusao(regra->Objeto().c_str(),regra->Atributo().c_str(),
      regra->Valor().c_str());
    conclusoes.Add(conclusao);

    // eliminá-la
    regras.Detach(regra,TShouldDelete::Delete);
  }
} // AplicaRegrasPassiveis

//---------------------------------------------------------------------------

void SistemaEspecialista::EliminaRegrasDependentesDe(WORD premissa)
{
  RegrasQueue condenadas(100);
  PRegra regra;

  // selecionar regras...
  RegrasSetIterator it(regras);
  while (it) {
    regra = it++;
    if (regra->DependeDe(premissa))
      condenadas.Put(regra);
  }

  // ... e eliminá-las
  while (condenadas.GetItemsInContainer()) {
    regra = condenadas.Get();
    regras.Detach(regra,TShouldDelete::Delete);
  }
} // EliminaRegrasDependentesDe

//---------------------------------------------------------------------------

BOOL SistemaEspecialista::GeraObjetos()
{
  ConclusoesSetIterator it(conclusoes);
  while (it) {
    RCConclusao conclusao = *it++;

    ObjetosQueue objetosSelecionados;

    if (ExisteObjeto(conclusao.Objeto()))
      SelecionaObjetos(conclusao.Objeto(),objetosSelecionados);
    else {
      // inclui novo objeto no conjunto
      PObjeto novoObjeto = new Objeto(conclusao.Objeto().c_str());
      objetos.Add(novoObjeto);
      objetosSelecionados.Put(novoObjeto);
    }

    while (objetosSelecionados.GetItemsInContainer()) {
      RObjeto objeto = *(objetosSelecionados.Get());

      if (ExisteAtributo(objeto,conclusao.Atributo())) {
        if (!(SelecionaAtributo(objeto,conclusao.Atributo()).Valor() == conclusao.Valor())) {
          // conflito
          PObjeto copia = new Objeto(objeto);
          SelecionaAtributo(*copia,conclusao.Atributo()).Valor(conclusao.Valor().c_str());
          objetos.Add(copia);
        }
      } else
        objeto.Atributos().Add(new Atributo(conclusao.Atributo().c_str(),
          conclusao.Valor().c_str()));
```

```
    }
  }

  return TRUE;
} // GeraObjetos

//-----------------------------------------------------------------------------------------

BOOL SistemaEspecialista::ExisteObjeto(const string &nomeObjeto)
{
  ObjetosBagIterator it(objetos);
  while (it) {
    RObjeto objeto = *it++;
    if (objeto.Nome() == nomeObjeto)
      return TRUE;
  }
  return FALSE;
} // ExisteObjeto

//-----------------------------------------------------------------------------------------

void SistemaEspecialista::SelecionaObjetos(const string &nomeObjeto,ObjetosQueue &selecionados)
{
  ObjetosBagIterator it(objetos);
  while (it) {
    PObjeto objeto = it++;
    if (objeto->Nome() == nomeObjeto)
      selecionados.Put(objeto);
  }
} // SelecionaObjetos

//-----------------------------------------------------------------------------------------

BOOL SistemaEspecialista::ExisteAtributo(RObjeto objeto,const string &nomeAtributo)
{
  AtributosSetIterator it(objeto.Atributos());
  while (it) {
    RAtributo atributo = *it++;
    if (atributo.Nome() == nomeAtributo)
      return TRUE;
  }
  return FALSE;
} // ExisteAtributo

//-----------------------------------------------------------------------------------------

RAtributo SistemaEspecialista::SelecionaAtributo(RObjeto objeto,const string &nomeAtributo)
{
  AtributosSetIterator it(objeto.Atributos());
  while (it) {
    RAtributo atributo = *it++;
    if (atributo.Nome() == nomeAtributo)
      return atributo;
  }

  TRACE2("Atributo solicitado não encontrado: retornando um atributo vazio.");
  return *new Atributo;
} // SelecionaAtributo

//-----------------------------------------------------------------------------------------

BOOL SistemaEspecialista::GeraArquivoObjetos(PCCHAR nomeArquivo)
{
  // alterando a extensão do arquivo para .OUT
  strcpy(buffer,nomeArquivo);
  PCHAR pos = buffer;
  while (*pos != '.')
    pos++;
  *++pos = 'O';
  *++pos = 'U';
  *++pos = 'T';
  *++pos = '\0';
  strcpy(arquivoSaida,buffer);

  // criando o arquivo
  ofstream arquivo(arquivoSaida);
  if (!arquivo.good())
```

```
    return FALSE;

  // escrevendo no arquivo os objetos gerados
  WORD numeroDoObjeto = 0;
  ObjetosBagIterator it(objetos);
  while (it) {
    RObjeto objeto = *it++;
    numeroDoObjeto++;

    // cabeçalho
    sBuffer.seekp(0);
    sBuffer << "Objeto #" << (int)numeroDoObjeto << ":\n\n" << ends;
    arquivo.write(buffer,strlen(buffer));

    // nome
    sBuffer.seekp(0);
    sBuffer << "Nome: " << objeto.Nome() << "\n" << ends;
    arquivo.write(buffer,strlen(buffer));

    // atributos
    sBuffer.seekp(0);
    sBuffer << "Atributos:\n" << ends;
    arquivo.write(buffer,strlen(buffer));
    AtributosSetIterator itAtrib(objeto.Atributos());
    while (itAtrib) {
      RCAtributo atributo = *itAtrib++;

      sBuffer.seekp(0);
      sBuffer << "     " << atributo.Nome() << " = " << atributo.Valor() << "\n" << ends;
      arquivo.write(buffer,strlen(buffer));
    }

    // separador para o próximo objeto
    if (it) {
      sBuffer.seekp(0);
      sBuffer <<
        "\n\n--------------------------------------------------------------------------------\n\n"
        << ends;
      arquivo.write(buffer,strlen(buffer));
    }
  }

  return TRUE;
} // GeraArquivoObjetos
```

Classe "PREMISSA"

Declaração (arquivo PREMISSA.H):

```
#ifndef __PREMISSA_H
#define __PREMISSA_H

#include <_defs.h>
#include <cstring.h>
#include <classlib\arrays.h>

#include <char.h>

_CLASSDEF(Premissa)

class Premissa {
  public:
    Premissa(PCCHAR _atributo = "",PCCHAR _valor = "");
    Premissa(RCPremissa);
    virtual ~Premissa();

    int operator==(RCPremissa) const;

    const string &Atributo() const { return atributo; }
```

```
    void Atributo(PCCHAR _atributo) { atributo = _atributo; }

    const string &Valor() const { return valor; }
    void Valor(PCCHAR _valor) { valor = _valor; }

    BOOL Avaliada() const { return avaliada; }
    void Avaliada(BOOL _avaliada) { avaliada = _avaliada; }

    BOOL Valida() const { return valida; }
    void Valida(BOOL _valida) { valida = _valida; avaliada = TRUE; }

  protected:

  private:
    string atributo,valor;
    BOOL avaliada,valida;
}; // Premissa

//---------------------------------------------------------------------------

typedef TIArrayAsVector<Premissa> PremissasArray;
typedef TIArrayAsVectorIterator<Premissa> PremissasArrayIterator;

//---------------------------------------------------------------------------

#endif __PREMISSA_H
```

Definição (arquivo PREMISSA.CPP):

```
#include "classes\premissa.h"

//---------------------------------------------------------------------------

Premissa::Premissa(PCCHAR _atributo,PCCHAR _valor) : atributo(_atributo),valor(_valor),
  avaliada(FALSE),valida(FALSE)
{
} // Premissa()

//---------------------------------------------------------------------------

Premissa::Premissa(RCPremissa outraPremissa) : atributo(outraPremissa.atributo),valor(
  outraPremissa.valor),avaliada(outraPremissa.avaliada),valida(outraPremissa.valida)
{
} // Premissa(RCPremissa)

//---------------------------------------------------------------------------

Premissa::~Premissa()
{
} // ~Premissa

//---------------------------------------------------------------------------

int Premissa::operator==(RCPremissa outraPremissa) const
{
  return atributo == outraPremissa.atributo && valor == outraPremissa.valor &&
    avaliada == outraPremissa.avaliada && valida == outraPremissa.valida;
} // operator==
```

Classe "Regra"

Declaração (arquivo REGRA.H):

```
#ifndef __REGRA_H
#define __REGRA_H

#include <_defs.h>
#include <cstring.h>
#include <classlib\sets.h>

#include <char.h>

//-----------------------------------------------------------------------------------------

typedef TSet<WORD> WordSet;
typedef TSetIterator<WORD> WordSetIterator;

//-----------------------------------------------------------------------------------------

_CLASSDEF(Regra)

class Regra {
  public:
    Regra(WordSet &_premissas,PCCHAR _objeto = "",PCCHAR _atributo = "",PCCHAR _valor = "");
    Regra(RCRegra);
    virtual ~Regra();

    int operator==(RCRegra) const;

    void Confirma(WORD premissa) { premissas.Detach(premissa); }
      // elimina a premissa do conjunto de premissas
    WORD DependeDe() const;
      // retorna uma das premissas da regra
    BOOL DependeDe(WORD premissa) const { return premissas.HasMember(premissa); }
      // informa se a regra depende da premissa
    WORD QuantPremissas() const { return premissas.GetItemsInContainer(); }
      // quantas premissas restam

    const string &Objeto() const { return objeto; }
    void Objeto(PCCHAR _objeto) { objeto = _objeto; }

    const string &Atributo() const { return atributo; }
    void Atributo(PCCHAR _atributo) { atributo = _atributo; }

    const string &Valor() const { return valor; }
    void Valor(PCCHAR _valor) { valor = _valor; }

  protected:

  private:
    WordSet premissas;
    string objeto,atributo,valor;
}; // Regra

//-----------------------------------------------------------------------------------------

typedef TISetAsVector<Regra> RegrasSet;
typedef TISetAsVectorIterator<Regra> RegrasSetIterator;

//-----------------------------------------------------------------------------------------

#endif __REGRA_H
```

Definição (arquivo REGRA.CPP):

```
#include "classes\regra.h"

#include <checks2.h>

//---------------------------------------------------------------------------

Regra::Regra(WordSet &_premissas,PCCHAR _objeto,PCCHAR _atributo,PCCHAR _valor) :
  objeto(_objeto),atributo(_atributo),valor(_valor)
{
  WordSetIterator it(_premissas);
  while (it)
    premissas.Add(it++);
} // Regra()

//---------------------------------------------------------------------------

Regra::Regra(RCRegra outraRegra) : objeto(outraRegra.objeto),atributo(outraRegra.atributo),
  valor(outraRegra.valor)
{
  WordSetIterator it(outraRegra.premissas);
  while (it)
    premissas.Add(it++);
} // Regra(RCRegra)

//---------------------------------------------------------------------------

Regra::~Regra()
{
} // ~Regra

//---------------------------------------------------------------------------

int Regra::operator==(RCRegra outraRegra) const
{
  return objeto == outraRegra.objeto && atributo == outraRegra.atributo && valor ==
    outraRegra.valor;
} // operator==

//---------------------------------------------------------------------------

WORD Regra::DependeDe() const
{
  WordSetIterator it(premissas);
  if (it)
    return it.Current();

  TRACE2("'DependeDe' chamado para uma regra sem premissas.");
  return 0;
} // DependeDe()
```

Classe "OBJETO"

Representa a conclusão gerada pela aplicação de uma regra no Sistema Especialista.

Declaração (arquivo OBJETO.H):

```
#ifndef __OBJETO_H
#define __OBJETO_H

#include <_defs.h>
```

```
#include <cstring.h>
#include <classlib\bags.h>
#include <classlib\queues.h>

#include <char.h>

#include "classes\atrib.h"

_CLASSDEF(Objeto)

class Objeto {
  public:
    Objeto(PCCHAR _nome = "");
    Objeto(RCObjeto);
    virtual ~Objeto();

    int operator==(RCObjeto) const;

    const string &Nome() const { return nome; }
    void Nome(PCCHAR _nome) { nome = _nome; }

    AtributosSet &Atributos() { return atributos; }

  protected:

  private:
    string nome;
    AtributosSet atributos;
}; // Objeto

//---------------------------------------------------------------------------

typedef TIBagAsVector<Objeto> ObjetosBag;
typedef TIBagAsVectorIterator<Objeto> ObjetosBagIterator;

typedef TIQueueAsVector<Objeto> ObjetosQueue;

//---------------------------------------------------------------------------

#endif __OBJETO_H
```

Definição (arquivo OBJETO.CPP):

```
#include "classes\objeto.h"

//---------------------------------------------------------------------------

Objeto::Objeto(PCCHAR _nome) : nome(_nome)
{
  atributos.OwnsElements(TRUE);
} // Objeto()

//---------------------------------------------------------------------------

Objeto::Objeto(RCObjeto outroObjeto) : nome(outroObjeto.nome)
{
  AtributosSetIterator itAtributos(outroObjeto.atributos);
  while (itAtributos)
    atributos.Add(new Atributo(*itAtributos++));

  atributos.OwnsElements(TRUE);
} // Objeto(RCObjeto)

//---------------------------------------------------------------------------

Objeto::~Objeto()
{
} // ~Objeto

//---------------------------------------------------------------------------

int Objeto::operator==(RCObjeto outroObjeto) const
{
```

```
  return nome == outroObjeto.nome;
} // operator==
```

Classes de *interface*

Esta seção contém o código-fonte das classes de *interface* do SISTEMA ESPECIALISTA. Estas classes estão intimamente relacionadas à OWL[5], a biblioteca utilizada para o desenvolvimento da *interface*.

Classe "SISTEMA ESPECIALISTA"

Esta classe representa o SISTEMA ESPECIALISTA enquanto aplicativo MS-Windows. Cada cópia do SISTEMA ESPECIALISTA em execução é uma instância desta classe.

Declaração (arquivo APP.H):

```
#ifndef __APP_H
#define __APP_H

#include <_defs.h>
#include <owl\applicat.h>

_CLASSDEF(_Application)

class _Application : public TApplication {
  public:
    _Application();
    virtual ~_Application();

  protected:
    virtual void InitInstance();
    virtual void InitMainWindow();
}; // _Application

#endif __APP_H
```

Definição (arquivo APP.CPP):

```
#include "main\app.h"
```

[5] A OWL anteriormente

```
#include <bwcc.h>

#include <dir2.h>

#include "main\mainwnd.h"

//----------------------------------------------------------------------------

_Application::_Application() : TApplication("Sistema Especialista")
{
} // _Application

//----------------------------------------------------------------------------

_Application::~_Application()
{
} // ~_Application

//----------------------------------------------------------------------------

void _Application::InitInstance()
{
  TApplication::InitInstance();

  ChangeToExeDir();
  EnableCtl3d();
} // InitInstance


//----------------------------------------------------------------------------

void _Application::InitMainWindow()
{
  SetMainWindow(new _MainWindow);
} // InitMainWindow
```

Classe "RESULTW"

Janela que mostra os resultados de uma execução do Sistema Especialista

Declaração (arquivo RESULTW.H):

```
#ifndef __RESULTW_H
#define __RESULTW_H

#include <_defs.h>
#include <owl\editfile.h>
#include <owl\mdichild.h>

#include <char.h>

_CLASSDEF(ResultWindow)

class ResultWindow : public TMDIChild {
  public:
    ResultWindow(TMDIClient &parent,PCCHAR _fileName);
    virtual ~ResultWindow();

    virtual BOOL CanClose();

  protected:
    virtual char far* GetClassName() { return "ResultWindow"; }
    virtual void GetWindowClass(WNDCLASS &windowClass);
    virtual void SetupWindow();

    virtual void EvSize(UINT sizeType,TSize &size);
    void EvSysCommand(UINT,TPoint&);
```

```
  private:
    TEditFile *editFile;
    char fileName[256];

  DECLARE_RESPONSE_TABLE(ResultWindow);
}; // ResultWindow

#endif __RESULTW_H
```

Definição (arquivo RESULTW.CPP):

```
#include "main\resultw.h"

#include <stdio.h>
#include <string.h>
#include <owl\menu.h>

#include <buffers.h>

#define CM_PRINT 0xF250

//----------------------------------------------------------------------------

ResultWindow::ResultWindow(TMDIClient &parent,PCCHAR _fileName) : TMDIChild(parent)
{
  strcpy(fileName,_fileName);
} // ResultWindow

//----------------------------------------------------------------------------

ResultWindow::~ResultWindow()
{
} // ~ResultWindow

//----------------------------------------------------------------------------

BOOL ResultWindow::CanClose()
{
  return TRUE;
} // CanClose

//----------------------------------------------------------------------------

void ResultWindow::GetWindowClass(WNDCLASS &windowClass)
{
  TMDIChild::GetWindowClass(windowClass);
} // GetWindowClass

//----------------------------------------------------------------------------

void ResultWindow::SetupWindow()
{
  TRect r = GetClientRect();
  editFile = new TEditFile(this,101,"",r.left,r.top,r.right,r.bottom,fileName);

  TMDIChild::SetupWindow();

  editFile->SetReadOnly(TRUE);

  // mudar system menu
  TMenu systemMenu(GetSystemMenu());
  systemMenu.AppendMenu(MF_STRING | MF_ENABLED,CM_PRINT,"Imprimir");
  DrawMenuBar();
} // SetupWindow

//----------------------------------------------------------------------------

void ResultWindow::EvSize(UINT sizeType,TSize &size)
{
  TMDIChild::EvSize(sizeType,size);
  editFile->MoveWindow(TRect(GetClientRect()),TRUE);
} // EvSize
```

```
//----------------------------------------------------------------------------------------------

void ResultWindow::EvSysCommand(UINT cmd,TPoint &point)
{
  TMDIChild::EvSysCommand(cmd,point);

  if ((cmd & 0xFFF0) == CM_PRINT)
    // imprimir o arquivo
    for (int i = 0;i < editFile->GetNumLines();i++)
      if (editFile->GetLine(buffer,sizeof(buffer),i)) {
        strcat(buffer,"\n");
        fprintf(stdprn,buffer);
      }
} // CmPrint

//----------------------------------------------------------------------------------------------

DEFINE_RESPONSE_TABLE1(ResultWindow,TMDIChild)
  EV_WM_SIZE,
  EV_WM_SYSCOMMAND,
END_RESPONSE_TABLE;
```

Classe "MainWindow"

Esta classe representa a janela principal do SISTEMA ESPECIALISTA

Declaração (arquivo MAINWND.H):

```
#ifndef __MAINWND_H
#define __MAINWND_H

#include <_defs.h>
#include <owl\decmdifr.h>

#include "classes\sistesp.h"

_CLASSDEF(_MainWindow)

class _MainWindow : public TDecoratedMDIFrame {
  public:
    _MainWindow();
    virtual ~_MainWindow();

    virtual BOOL CanClose();

  protected:
    virtual char far* GetClassName() { return "_MainWindow"; }
    virtual void GetWindowClass(WNDCLASS &windowClass);
    virtual void SetupWindow();

    void CmExecute();
    void CmReExecute();

  private:
    SistemaEspecialista sistemaEspecialista;
    BOOL primeiraVez;

    void SetupControlBar();
    void SetupStatusBar();

  DECLARE_RESPONSE_TABLE(_MainWindow);
}; // _MainWindow

#endif __MAINWND_H
```

Definição (arquivo MAINWND.CPP):

```
#include "main\mainwnd.h"

#include <owl\buttonga.h>
#include <owl\controlb.h>
#include <owl\opensave.h>
#include <owl\statusba.h>

#include <buffers.h>
#include <windows2.h>

#include "classes\sistesp.h"
#include "main\menus.rh"
#include "main\resultw.h"

//----------------------------------------------------------------------------

_MainWindow::_MainWindow() : TDecoratedMDIFrame("Sistema Especialista","MAIN_MENU",
  *new TMDIClient,TRUE),primeiraVez(TRUE)
{
} // _MainWindow

//----------------------------------------------------------------------------

_MainWindow::~_MainWindow()
{
} // ~_MainWindow

//----------------------------------------------------------------------------

BOOL _MainWindow::CanClose()
{
  return TRUE;
} // CanClose

//----------------------------------------------------------------------------

void _MainWindow::GetWindowClass(WNDCLASS &windowClass)
{
  TDecoratedMDIFrame::GetWindowClass(windowClass);
  windowClass.hIcon = GetApplication()->LoadIcon("SISTESP_ICON");
} // GetWindowClass

//----------------------------------------------------------------------------

void _MainWindow::SetupWindow()
{
  SetupControlBar();
  SetupStatusBar();

  TDecoratedMDIFrame::SetupWindow();
} // SetupWindow

//----------------------------------------------------------------------------

void _MainWindow::SetupControlBar()
{
  TControlBar *controlBar = new TControlBar(this);

  controlBar->Insert(*new TButtonGadget(CM_EXIT,CM_EXIT));
  controlBar->Insert(*new TSeparatorGadget);
  controlBar->Insert(*new TButtonGadget(CM_EXECUTE,CM_EXECUTE));
  controlBar->Insert(*new TButtonGadget(CM_RE_EXECUTE,CM_RE_EXECUTE));
  controlBar->SetHintMode(TGadgetWindow::EnterHints);

  Insert(*controlBar,Top);
} // SetupControlBar

//----------------------------------------------------------------------------

void _MainWindow::SetupStatusBar()
{
  TStatusBar *statusBar = new TStatusBar(this,TGadget::Recessed,TStatusBar::CapsLock |
    TStatusBar::NumLock);
  Insert(*statusBar,Bottom);
} // SetupStatusBar
```

```
//----------------------------------------------------------------------------------------------

void _MainWindow::CmExecute()
{
  TOpenSaveDialog::TData dialogData(OFN_FILEMUSTEXIST | OFN_HIDEREADONLY,
    "Arquivos de premissas/regras (*.IN)|*.in|Todos os arquivos (*.*)|*.*|",0,"","IN");

  if (TFileOpenDialog(this,dialogData).Execute() == IDOK) {
    sBuffer.seekp(0);
    sBuffer << "Sistema Especialista - " << dialogData.FileName << ends;
    SetWindowText(buffer);

    char arquivoSaida[256];
    strcpy(arquivoSaida,"");
    sistemaEspecialista.Executa(dialogData.FileName,arquivoSaida);
    if (arquivoSaida[0])
      (new ResultWindow(*GetClientWindow(),arquivoSaida))->Create();
    primeiraVez = FALSE;
  }
} // CmExecute

//----------------------------------------------------------------------------------------------

void _MainWindow::CmReExecute()
{
  if (!primeiraVez) {
    char arquivoSaida[256];
    strcpy(arquivoSaida,"");
    sistemaEspecialista.ReExecuta(arquivoSaida);
    if (arquivoSaida[0])
      (new ResultWindow(*GetClientWindow(),arquivoSaida))->Create();
  } else
    LoudMessage(NULL,"Nenhum arquivo aberto.","Erro",MB_ICONSTOP,NULL);
} // CmReExecute

//----------------------------------------------------------------------------------------------

DEFINE_RESPONSE_TABLE1(_MainWindow,TDecoratedMDIFrame)
  EV_COMMAND(CM_EXECUTE,CmExecute),
  EV_COMMAND(CM_RE_EXECUTE,CmReExecute),
END_RESPONSE_TABLE;
```

Função "MAIN"

Função principal, que cria uma instância da classe "_Application".

Definição (arquivo MAIN.CPP):

```
#include "main\app.h"

//----------------------------------------------------------------------------------------------

int OwlMain(int,char*[])
{
  return _Application().Run();
} // OwlMain
```

ANEXO II - PROTOTIPAÇÃO

CICLOS DE DESENVOLVIMENTO

Um ciclo de desenvolvimento engloba todas as atividades necessárias para definir, desenvolver, testar, implantar e manter um software.

PROTOTIPAÇÃO

Uma maneira de encarar o ciclo de desenvolvimento de *software* é proposta pela prototipação. Este modelo enfatiza as fonte de solicitação do produto, focaliza os pontos de indecisão na abordagem do *software* e encoraja o uso de protótipos. As fontes de solicitação do produto são os usuários, clientes e todos aqueles direta ou indiretamente interessados na produção do *software.*

Diferentemente do que acontece no ciclo tradicional, na prototipação há um contato intenso entre o desenvolvedor, os usuários e os demais interessados; não é apenas em uma etapa (especificação do produto)

que existirá conversação entre as partes, mas é um processo contínuo de avaliação perante as necessidades da exist~encia do *software*. Há uma preocupação constante por parte de quem desenvolve de conseguir alcançar o nível de satisfação desejado pelo usuário e de todos aqueles responsáveis pela implantação do produto. A prototipação encoraja o desenvolvedor a conhecer melhor os usuários e suas necessidades, devido ao amplo debate que é propriciado pelo modelo sobre questões como : necessecidades, viabilidade, abordagem, preocupações e prioridades do *software*.

O nome prototipação origina-se de protótipo. Para a engenharia de *software* , um protótipo é também um sistema, e incorpora vários componentes do produto final. Geralmente o que diferencia um protótipo de seu produto final é a limitação de suas funcionalidades, a baixa confiabilidade e/ou a performance comprometida.

Em [BRO75], Brooks reforça a vantagem da prototipação sobre o ciclo tradicional dizendo que é impossível "acertar" um *sotware* da primeira vez. Gunther, em [GUN78], sugere que o protótipo pode ser considerado a primeira versão do *software* , e após a validação do mesmo, tem-se subsídio para se criar a primeira versão do produto final, que é a segunda versão do *software* .

ANEXO IV - Interface do SISTEMA ESPECIALISTA

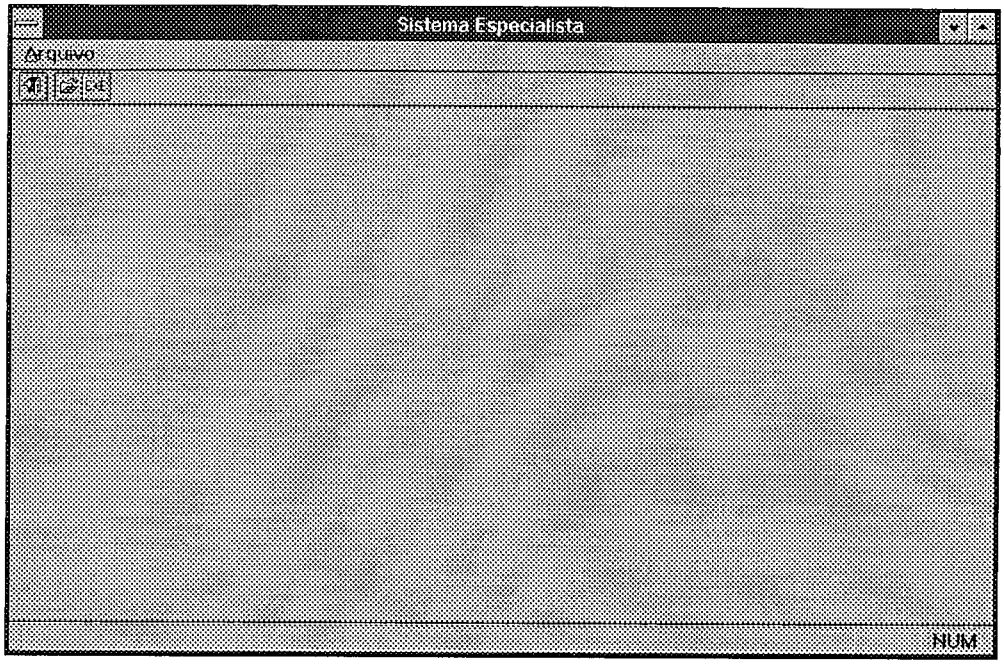

FIGURA 4.9 - Janela Principal do SISTEMA ESPECIALISTA

O SISTEMA ESPECIALISTA executado apresenta como interface uma janela com título Sistema Especialista dando a possibilidade do usuário Sair do Sistema, Carregar um arquivo de regras e premissa e executar o sistema ou Executar o último arquivo carregado que está disponível após a primeira execução.

Estas opções de inicialização do sistema estão na forma de Menu e na forma de Botões Tolbar (esse último ao ser tocado pelo mouse mostra na barra de mensagens sua função como podemos ver na FIGURA 4.11, FIGURA 4.12 e FIGURA 4.13.

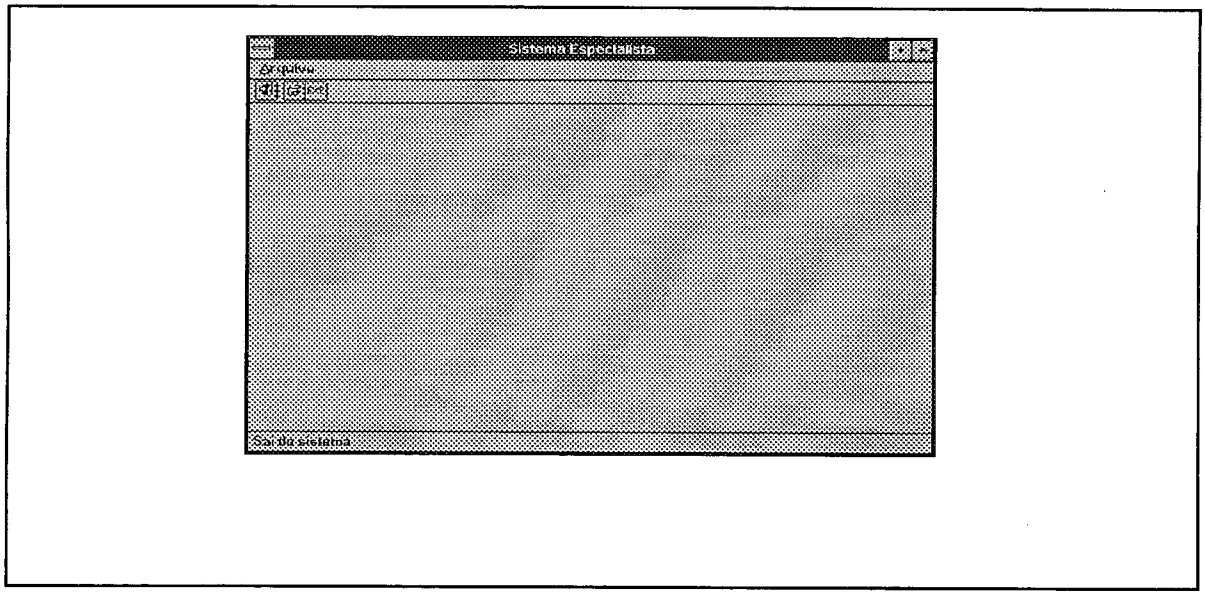

FIGURA 4.11- Sai do sistema

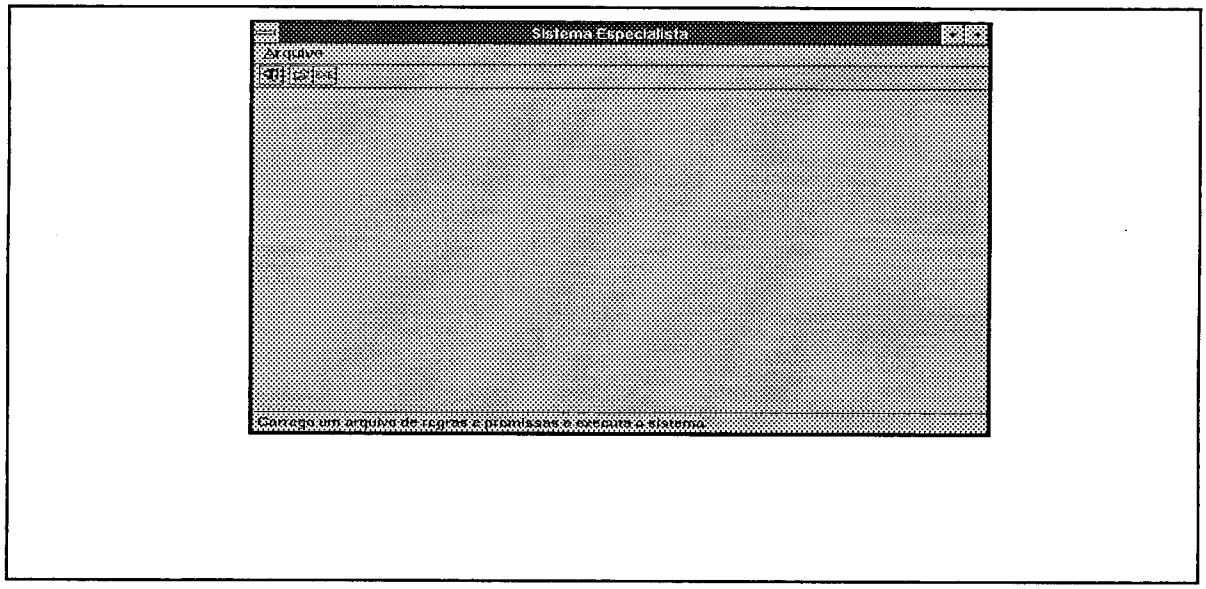

FIGURA 4.11 - Carrega um arquivo de Regras e Premissas e Executa o Sistema

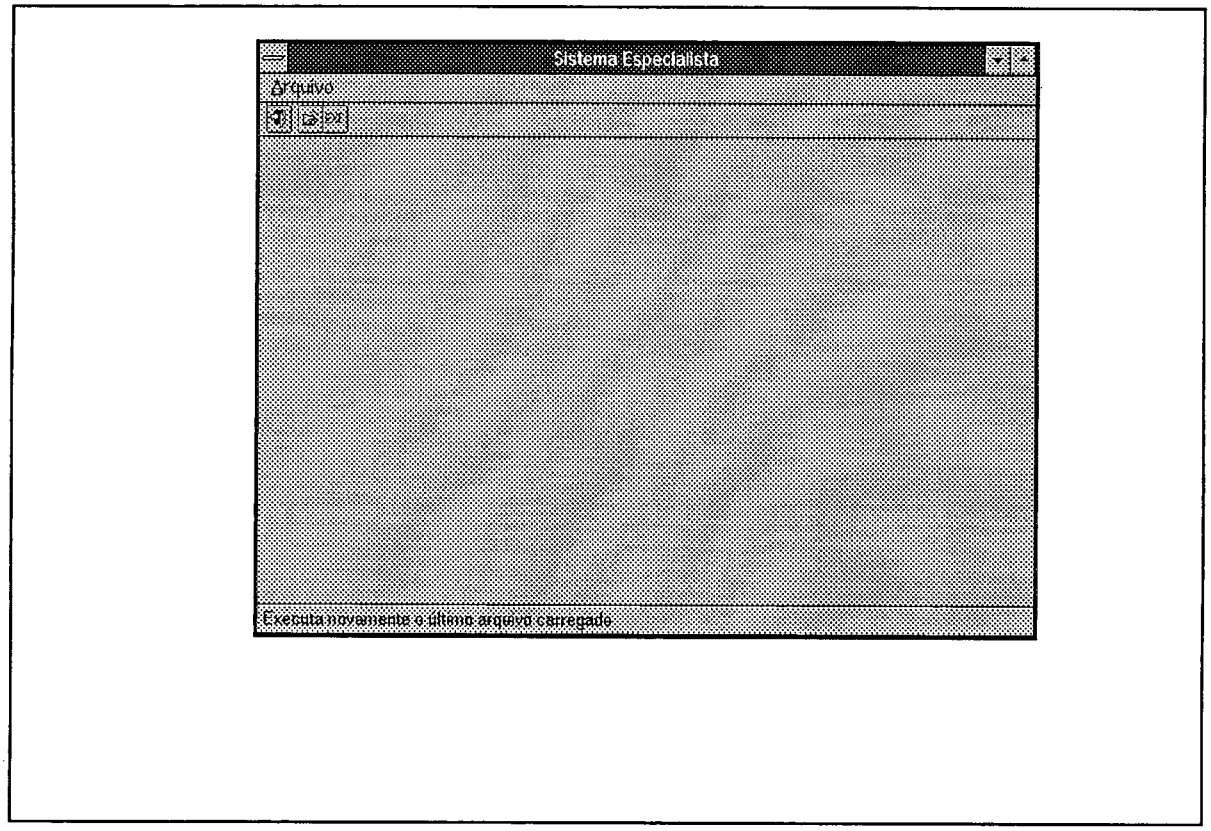

FIGURA 4.12 Executa novamente o último arquivo carregado

Após o usuário escolher a opção Carrega um arquivo de regras e premissas e executa o sistema abre-se uma janela possibilitando abrir o arquivo de premissas/regras com extensão (.in). Como podemos observar na FIGURA 4.13.

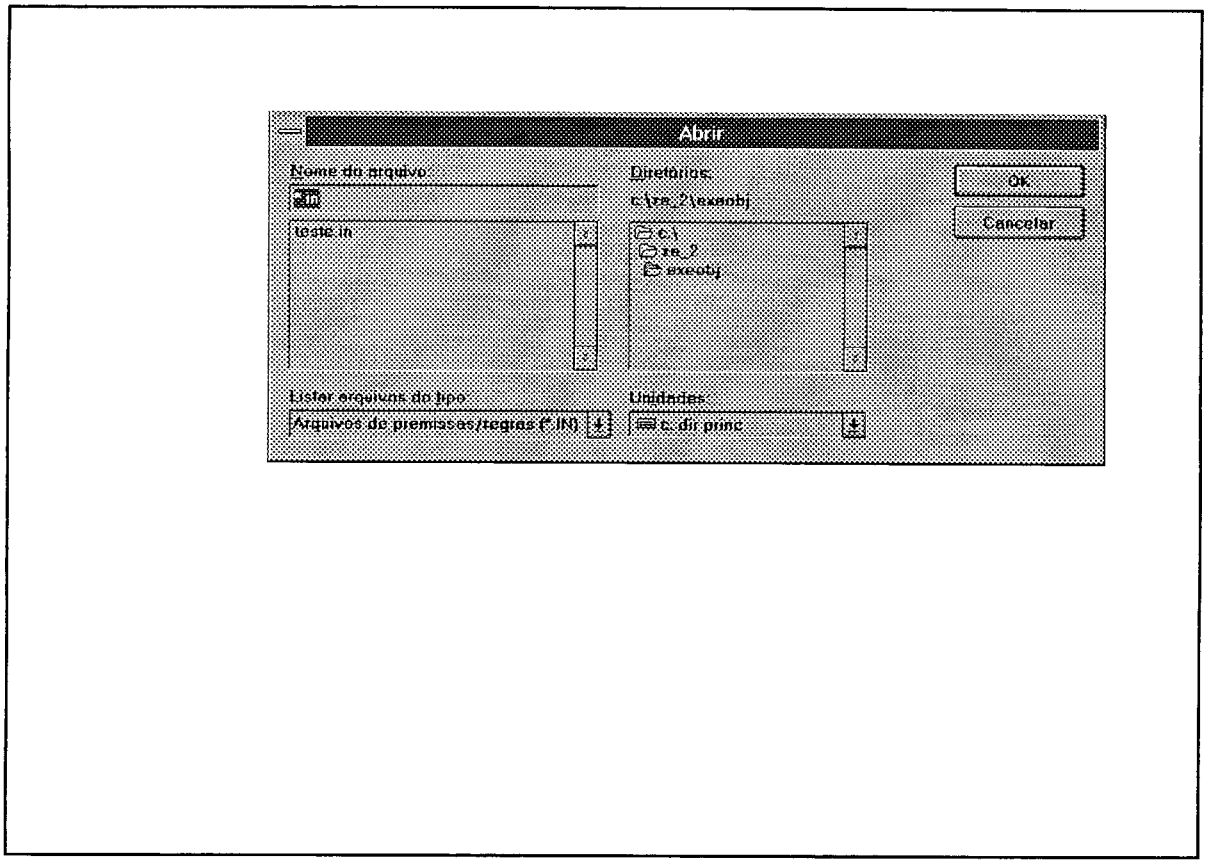

FIGURA 4.13 Abrir arquivo de premissas/regras (.in)

Escolhido o arquivo o Sistema Especialista apresenta as premissas com opção de confirmação (sim), negação (não) ou cancelamento (cancelar) conforme FIGURA 4.14 e FIGURA 4.15.

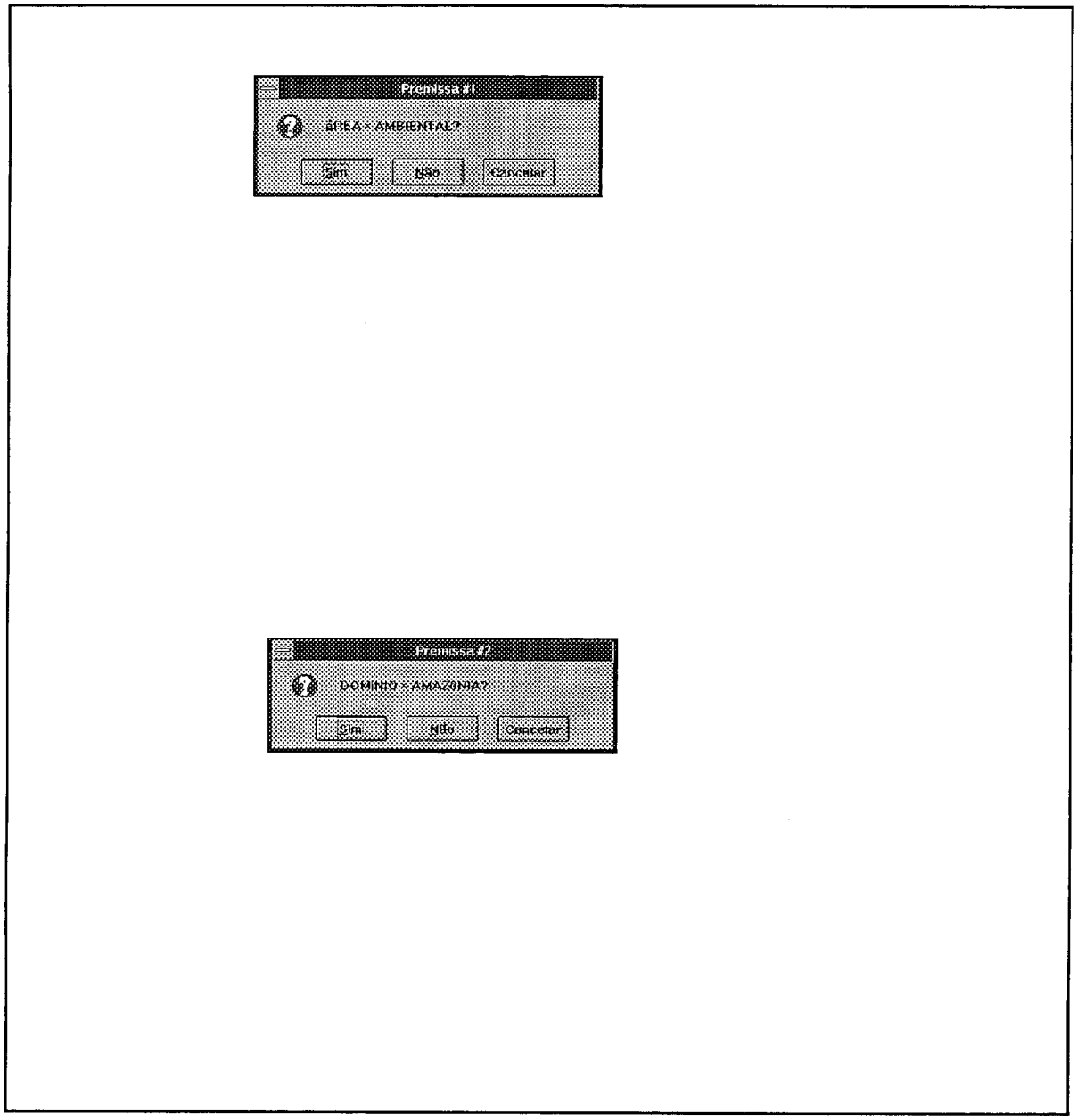

FIGURA 4.14 Diálogo escolhe premissa

Escolhida as premissas é apresentada num quadro de apresentação com o título Atenção o número de conclusões geradas, na FIGURA 4.15.

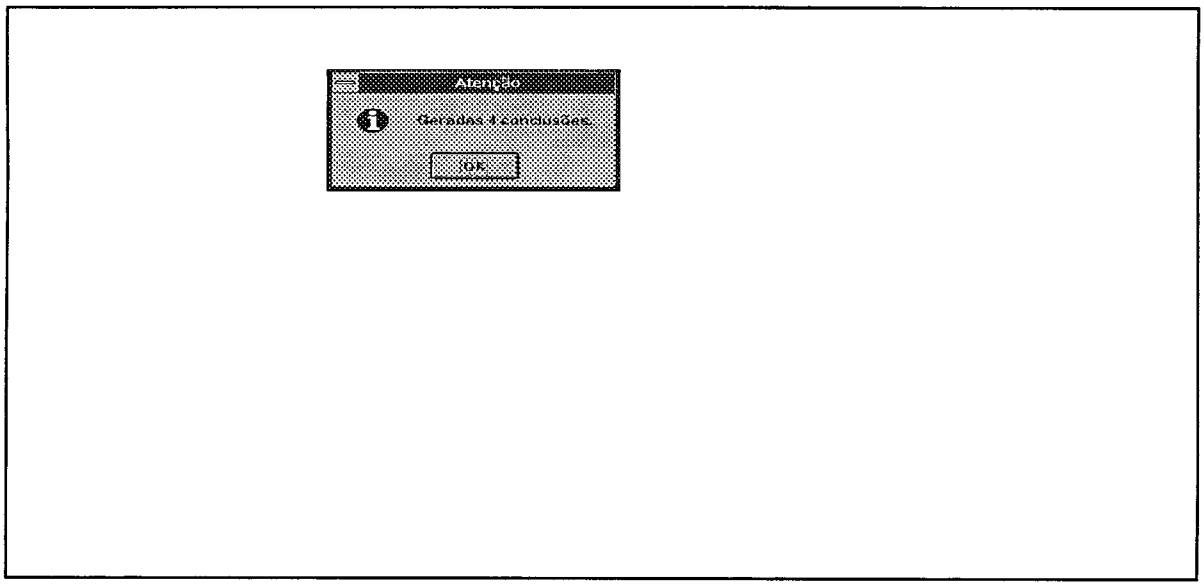

FIGURA 4.15 Conclusões geradas

Demonstrando num segundo quadro, o número de objetos gerados. E aviso que o arquivo de saída com extensão (.out) foi gerado, FIGURA 4.16.

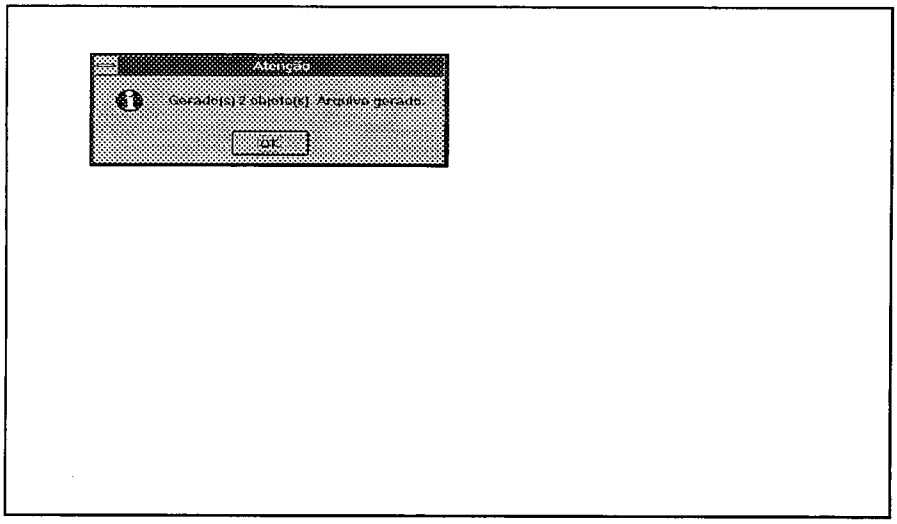

FIGURA 4.16 Quadro de atenção - número objetos gerados

Em seguida é mostrado na tela em uma janela o arquivo com extensão (.out) com o número e nome do objeto e seus atributos conforme FIGURA 4.17.

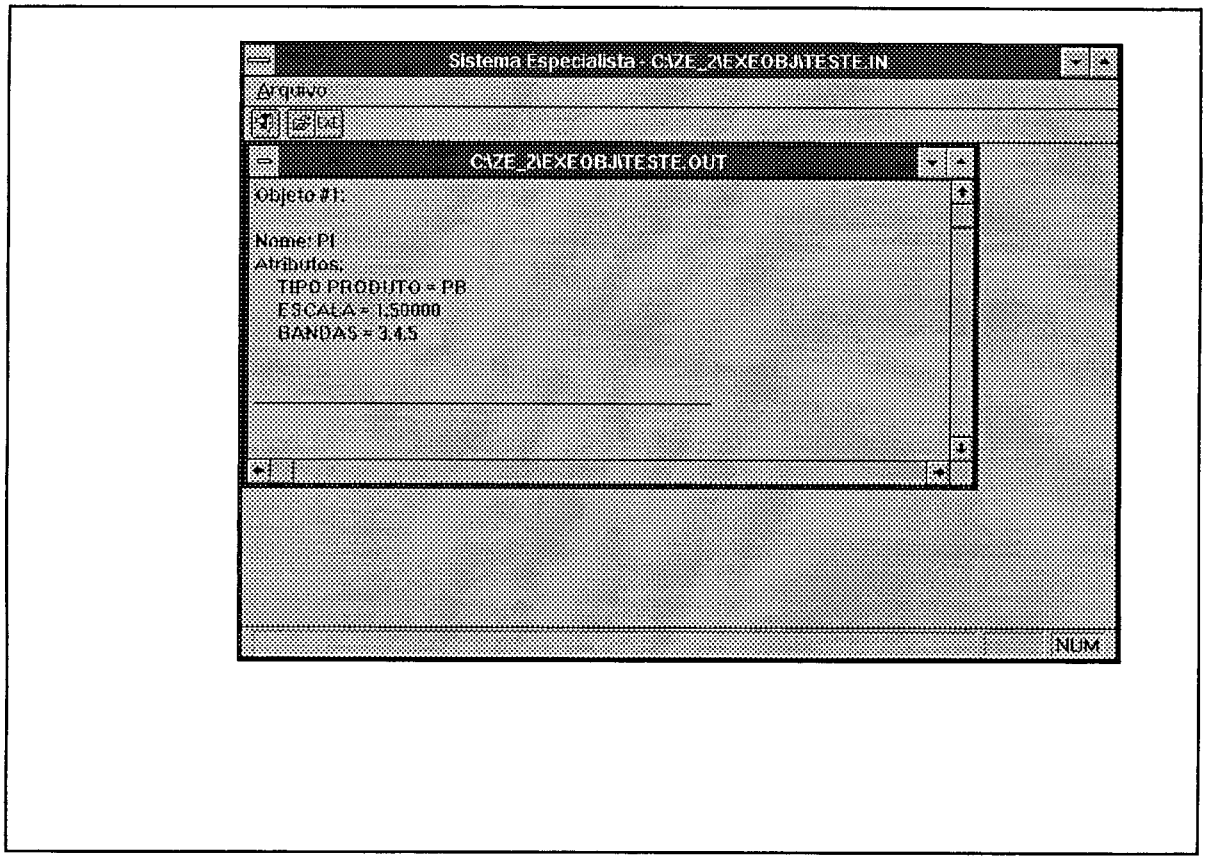

FIGURA 4.17 Arquivo gerado

Nessa janela é dado ao usuário a possibilidade de imprimir o resultado obtido no arquivo com extensão (.out), mostrado na FIGURA 4.18. No canto lateral esquerdo pode abrir-se um menu com várias opções entre elas a possibilidade de imprimir o arquivo.

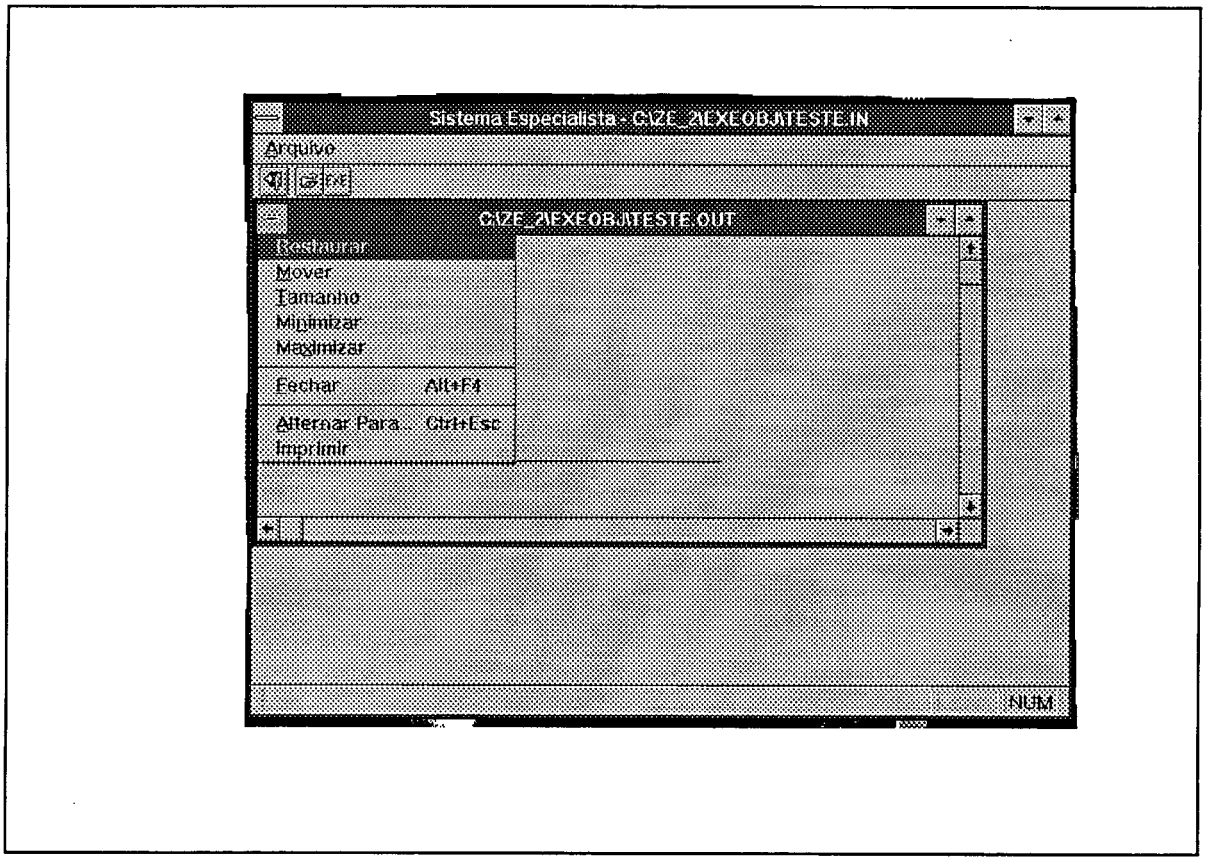

FIGURA 4.18 Opções de menu - imprimir arquivo

Uma vez realizada a operação desejada fica a critério do usuário fechar a janela do arquivo mostrado ou deixá-lo a disposição para posterior consulta e executar novamente ou sair do sistema, conforme FIGURA 4.19.

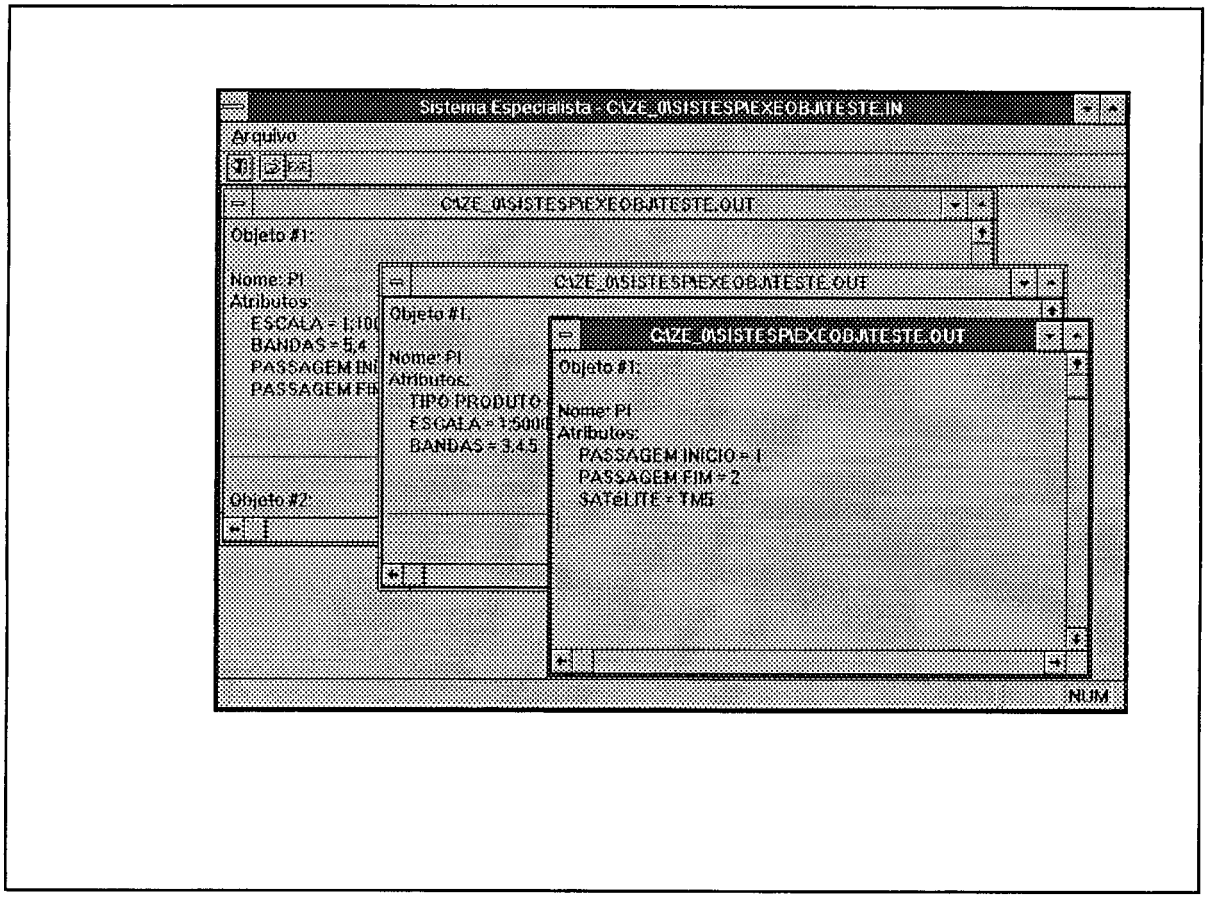

FIGURA 4.19 Janelas de arquivos gerados a partir das inferências

Para sair do sistema o usuário deve "clicar" ou selecionar a opção sair do sistema .

Anexo V - Aplica .in

```
PREMISSAS

1;Escala mapa final;1:100.000
2;Escala mapa final;1:50.000
3;produto imagem;Digital
4;produto imagem;Fotografico
5;APLICACAO;USO DA TERRA
6;APLICACAO;EXPANSAO DA AREA URBANA
7;APLICACAO;AREAS DE REFLORESTAMENTOS
8;APLICACAO;ESTUDOS DE SEDIM EM SUSP EM CORPOS AGUA
9;APLICACAO;DELIMITACAO DE CORPOS DE AGUA
10;APLICACAO;MAPEAMENTO DA REDE DE DRENAGEM
11;APLICACAO;ESTRESSES NA VEGETACAO
12;APLICACAO;IDENTIFICACAO DA REDE VIARIA
13;APLICACAO;IDENTIFICACAO DE AREAS DESMATADAS
14;APLICACAO;MONITORAMENTO DA QUALIDADE DA AGUA
15;APLICACAO;ESTUDO DE AREAS DEGRADADAS
16;APLICACAO;MONITORAMENTO DA ACAO ANTROPICA
17;APLICACAO;ESTUDO DA EROSAO
18;APLICACAO;TIPOS DE SOLO
19;APLICACAO;INVENTARIO FLORESTAL
20;APLICACAO;IDENT DE AREAS DE QUEIMADOS
21;APLICACAO;IDENT DE AREAS DE SOLO EXPOSTO
22;APLICACAO;IDENT DE AREAS DE MINERACAO
23;APLICACAO;IDENT DE AREAS IRRIGADAS
24;APLICACAO;IDENT DE AREAS OCUP PINUS E EUCALIP
25;APLICACAO;IDENT DE DUNAS
26;APLICACAO;ESTUDO DE ASSOREAMENTO

REGRAS

1,3,5;PI;Sensor;TM - 3, 4 e 5
1,3,5;PI;Sensor;HRV - XS1, XS2 e XS3
1,4,5;PI;Sensor;TM - 3, 4 e 5
1,4,5;PI;Composicao;Colorida

2,3,5;PI;Sensor;TM - 3, 4 e 5
2,3,5;PI;Sensor;HRV - pancromatica
2,4,5;PI;Sensor;HRV - pancromatica

2,3,6;PI;Sensor;TM - 3, 4 e 5
2,3,6;PI;Sensor;HRV - XS1, XS2 e XS3
2,4,6;PI;Sensor;TM - 3
2,4,6;PI;Sensor;HRV - XS2

1,3,7;PI;Sensor;TM - 3, 4 e 5
1,3,7;PI;Sensor;HRV - XS1, XS2 e XS3
1,4,7;PI;Sensor;TM - 3
1,4,7;PI;Sensor;HRV - XS2

2,3,8;PI;Sensor;TM - 1, 2 e 3
2,4,8;PI;Composicao;Colorida
2,4,8;PI;Bandas;1, 2 e 3

2,3,9;PI;Sensor;TM - 3, 4 e 5
2,4,9;PI;Sensor;TM - 4
2,4,9;PI;Sensor;HRV - XS3

2,3,10;PI;Sensor;TM - 3, 4 e 5
2,4,10;PI;Sensor;TM - 4
2,4,10;PI;Sensor;HRV - XS3

1,3,11;PI;Sensor;TM - 5
1,4,11;PI;Sensor;TM - 5
```

2,3,12;PI;Sensor;HRV - pancromatica
2,4,12;PI;Sensor;HRV - pancromatica

2,3,13;PI;Sensor;TM - 3, 4 e 5
2,3,13;PI;Sensor;HRV - XS1, XS2 e XS3
2,4,13;PI;Sensor;TM - 3
2,4,13;PI;Sensor;HRV - XS2

2,3,14;PI;Sensor;TM - 1, 2 e 4
2,3,14;PI;Sensor;HRV - XS1, XS2 e XS3

2,3,15;PI;Sensor;TM - 3, 4 e 5
2,4,15;PI;Sensor;TM - 2, 3 e 4

1,3,16;PI;Sensor;TM - 3, 4 e 5
1,3,16;PI;Sensor;HRV - XS1, XS2 e XS3
1,4,16;PI;Sensor;TM - 3 e 4

2,3,17;PI;Sensor;TM - 3, 4 e 5
2,3,17;PI;Sensor;HRV - XS1, XS2 e XS3

1,3,18;PI;Sensor;TM - 3, 4 e 7
1,4,18;PI;Sensor;TM - 7, 4 e 3

1,3,19;PI;Sensor;TM - 3, 4 e 5

2,3,20;PI;Sensor;TM - 3, 4 e 5
2,4,20;PI;Sensor;TM - 4
2,4,20;PI;Sensor;HRV - XS3

2,3,21;PI;Sensor;TM - 3, 4 e 5
2,4,21;PI;Sensor;TM - 3
2,4,21;PI;Sensor;HRV - XS2

2,3,22;PI;Sensor;TM - 3, 4 e 5
2,4,22;PI;Sensor;TM - 4
2,4,22;PI;Sensor;HRV - XS3

2,3,23;PI;Sensor;TM - 3, 4 e 5
2,4,23;PI;Sensor;TM - 4
2,4,23;PI;Sensor;HRV - XS3

1,4,24;PI;Sensor;TM - 4
1,4,24;PI;Sensor;HRV - XS3

3,2,25;PI;Sensor;TM - 1, 2 e 3
2,3,25;PI;Sensor;HRV - XS3

2,3,26;PI;Sensor;TM - 1, 2 e 3
2,4,26;PI;Sensor;TM - 3
2,4,26;PI;Sensor;HRV - XS2

FIM