UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA

DEPARTAMENTO DE CONTROLE E AUTOMAÇÃO

Este exemplar corresponda à redação final da tese defendida por Marcos Melo Schiavini e aprovada pela Comissão Julgadora em 20 / 02 / 91.

Orientador

FERRAMENTA DE AQUISIÇÃO DE CONHECIMENTO POR MODELOS EXPLÍCITOS

Por: Marcos Melo Schiavini

Orientador: Marcio Luiz de Andrade Netto

Tese de Mestrado apresentada à
Faculdade de Engenharia Elétrica da
Universidade Estadual de Campinas

Fevereiro de 1991

UNICAMP
BIBLIOTECA CENTRAL

Este trabalho contou com o apoio financeiro da CAPES; contou com o apoio técnico e logístico da Villares, do CCR/IBM e do Laboratório de Sistemas Integráveis da USP.

Dedico este trabalho aos meus pais

Antonio e Neusa Melo Schiavini

No curso deste trabalho, contamos com a participação e apoio de muitas pessoas que contribuíram para que o concluíssemos com sucesso. Dentre estas queremos aqui registrar o agradecimento:

Ao prof. MARCIO LUIZ DE ANDRADE NETTO, meu orientador de programa e de tese, pelo apoio, pela crítica constante, objetividade, otimismo, incentivo e por se ter dignado a presidir esta banca.

Ao prof. FERNANDO GOMIDE pelas idéias, críticas e sugestões em qualidade e quantidades tais que seu nome poderia até figurar como meu co-orientador.

Ao prof. MARCIO RILLO pelo apoio, pela atenção que me dedicou, pela orientação, pela acolhida fraternal no LSI e pelas inúmeras oportunidades de trabalho propiciadas.

À prof. MARIA CAROLINA por ter aceito ser membro desta banca.

Aos professores do DEPARTAMENTO DE CONTROLE E AUTOMAÇÃO, pela acolhida, apoio e orientação nas disciplinas cursadas;

Aos companheiros de equipe DAMSKI, GIORNO, MILANI, GILBERTO, ANDRE, MARCIO, pelas sugestões, críticas, cobranças e um convívio de tão agradável, quase familiar;

Aos amigos e colegas de trabalho que me proporcionaram uma convivência gratificante no tempo em que vivi em Campinas.

RESUMO

A tese apresenta uma contribuição para agilizar e organizar o processo de aquisição de conhecimento necessário ao desenvolvimento de Sistemas Especialistas. Para tanto é descrita uma ferramenta computacional de auxílio ao processo de aquisição e engenharia de conhecimento - CAKE - que emprega um modelo do domínio durante sua interação com o especialista. O modelo é elaborado e representado com o auxílio do KADS, uma metodologia de construção de sistemas baseados em conhecimento [WIELINGA 89].

Com esse trabalho visamos obter uma ferramenta de aquisição de conhecimento que não apenas apresente as vantagens de empregar um modelo como também não tenha seu uso limitado a apenas um domínio particular. Para tanto concebemos uma ferramenta que deixa explícito o modelo utilizado para guiar o processo de aquisição de conhecimento. O engenheiro do conhecimento pode alterar a ferramenta para adeqüá-la às suas necessidades.

PALAVRAS-CHAVE: Sistemas Especialistas, Aquisição de Conhecimento, Ferramenta de Aquisição de Conhecimento, Engenharia de Conhecimento, Modelagem de Conhecimento.

ABSTRACT

The theses presents a contribution to facilitate and to organize the knowledge acquisition process necessary in the development of Expert Systems. A computer aided knowledge acquisition and engineering tool - CAKE - that employs a domain model in its interaction with the expert is proposed. The model is constructed and represented with the help of KADS, a methodology to construct knowledge based systems [WIELINGA 89].

It is intended, with this work, to obtain a knowledge acquisition tool that not only has the advantages of using a model, but also does not have its applicability limited to a particular domain. For this purpose, we have conceived a tool that leaves explicit the model used in guiding the knowledge acquisition process. The knowledge engineer is able to modify the tool to make the necessary adaptations for his further needs.

KEY-WORDS: Expert Systems, Knowledge Acquisition, Knowledge Acquisition Tool, Knowledge Engineering, Knowledge Modeling.

CONTEÚDO

CAPÍTULO 1

INTRODUÇÃO GERAL

1.1 HISTÓRICO

Este trabalho foi iniciado como parte do acordo de colaboração tecnológica entre CCR/IBM e NIA/Villares cujo propósito era desenvolver técnicas e ferramentas para agilizar o processo de construção de Sistemas Especialistas. O programa de pesquisas que se originou desse acordo passou por diversas fases, entre elas uma fase inicial dedicada a estudos sobre a Aquisição de Conhecimento baseada em técnicas psicológicas e uma fase dedicada ao estudo da Aquisição de Conhecimento baseada em modelos e, mais especificamente, voltada para a metodologia KADS.

Como parte desse projeto foi desenvolvido o protótipo de uma ferramenta cujo objetivo era servir de base para a implantação da metodologia KADS na IBM e na Villares. A ferramenta que descreveremos aqui, apesar de ter se inspirado nesse protótipo e de possuir diversos conceitos da metodologia KADS, foi desenvolvida de maneira independente ao acordo de pesquisas IBM/Villares e é uma ferramenta de Aquisição de Conhecimento e não uma ferramenta KADS.

1.2 CONTEXTO DO TRABALHO

A Engenharia do Conhecimento, que é a arte de construir Sistemas Baseados em Conhecimento (SBCs), tem ganho grande impulso graças à crescente penetração comercial de tais sistemas. Os SBCs (Sistemas Baseados em Conhecimento) são sistemas que resolvem problemas usando conhecimento acerca de um domínio. Uma importante

classe de SBCs são os Sistemas Especialistas (SEs) que são SBCs que resolvem problemas da vida real que exigem uma considerável quantidade de conhecimento especializado quando resolvido por seres humanos [WIELINGA 87].

A construção de um tal artefato pressupõe a obtenção do conhecimento relevante para o exercício de uma atividade especializada através de um processo de interação com o(s) especialista(s) que possui(em) tal conhecimento, a modelagem do conhecimento através de uma representação do mesmo em algum formalismo adequado e a construção de um sistema que seja capaz de manipular essa representação do conhecimento para fins práticos.

De todas essas atividades a aquisição do conhecimento do especialista é reconhecidamente a mais difícil e demorada [BERRY 87] [OLSON 87]. Isto se deve, parte ao fato do especialista não ser capaz de expor todo seu conhecimento de uma maneira formalmente adequada e, em parte, à dificuldade que tem o Engenheiro de Conhecimento (que é o indivíduo incumbido de conceber o SE - Sistema Especialista) de converter o conhecimento fornecido pelo especialista para uma representação conveniente.

Diversas técnicas já foram concebidas para auxiliar a Aquisição de Conhecimento. Uma das estratégias que vem ganhando grande atenção é o emprego de ferramentas automáticas capazes de interagir diretamente com o especialista. Essas ferramentas permitem que um especialista, sem o auxílio de um Engenheiro de Conhecimento, construa rapidamente um SE para as suas necessidades. A própria ferramenta de Aquisição de Conhecimento solicita ao especialista as informações necessárias. Deve-se, porém, ter uma ferramenta adequada

para a área que se deseja abordar pois cada ferramenta só é adequada para uma classe de problemas [BREUKER 88]. Isto ocorre por que para determinar as informações que se deve obter do especialista e como elas devem ser estruturadas é necessário ter um conhecimento básico da área em que se está trabalhando, ou seja, um modelo de como o especialista resolve os problemas de seu domínio específico. Tal modelo é implícito a toda ferramenta de Aquisição de Conhecimento capaz de interagir diretamente com um especialista. Como um tal modelo não pode ser modificado, essas ferramentas de Aquisição de Conhecimento são inadequadas para situações aonde esse modelo não seja correto.

A metodologia KADS de desenvolvimento de SBCs [WIELINGA 89] foi concebida procurando superar as limitações da Aquisição de Conhecimento baseada em um modelo único de certa atividade. Para tanto, esta abordagem prescreve métodos para se modelar tarefas genéricas. O projeto KADS, porém, voltou-se para a concepção de um ambiente para dar suporte à sua metodologia e não especificamente à elaboração de ferramentas de Aquisição de Conhecimento que pudessem ser empregadas por pessoas não familiarizadas com a metodologia KADS e, em geral, com a área da Inteligência Artificial.

1.3. OBJETIVOS

Esse trabalho visa elaborar uma ferramenta de Aquisição de Conhecimento capaz de trabalhar com um modelo explícito de certa atividade, modelo esse construído em consonância com a metodologia

KADS. Com isso visamos obter uma ferramenta de Aquisição de Conhecimento de uso geral, uma vez que o modelo empregado pela ferramenta poderá ser alterado e adaptado a cada caso específico. Para que essa ferramenta possa ser testada será especificado um modelo para uma atividade específica (diagnose de falhas em equipamentos) e será implementado um protótipo dessa ferramenta capaz de operar com uma simplificação desse modelo de diagnose.

1.4. ORGANIZAÇÃO

Além da introdução, nossa dissertação tem mais cinco capítulos e um anexo, constando, no segundo capítulo, as principais técnicas de Aquisição de Conhecimento; no terceiro capítulo, um resumo da metodologia KADS; no quarto, o modelo que empregaremos na ferramenta; no quinto, a descrição da ferramenta de Aquisição de Conhecimento que desenvolvemos; no sexto, a síntese dos principais resultados e apresentação de algumas perspectivas de evolução; no anexo é descrita a implementação da ferramenta de Aquisição de Conhecimento.

O segundo capítulo está dividido em sete itens:

- O item 2.1. introduz o leitor em alguns conceitos básicos para o entendimento do restante do capítulo, que é a Engenharia do Conhecimento e a Aquisição de Conhecimento.

- No item 2.2. trata-se das dificuldades envolvidas no processo de Aquisição de Conhecimento e propomos uma taxonomia para facilitar o entendimento do processo.

- No item 2.3 são apresentadas as principais técnicas psicológicas de Aquisição de Conhecimento e um resumo de cada uma delas, assim como propomos uma taxonomia para classificá-las.

- No item 2.4 são apresentadas as principais técnicas de Aquisição de Conhecimento por aprendizagem.

- O item 2.5 trata das técnicas de Aquisição de Conhecimento baseadas em linguagem.

- O item 2.6 refere-se às técnicas de Aquisição de Conhecimento baseadas em ferramentas.

- O item 2.7 sintetiza os principais conceitos apresentados no capítulo e faz uma análise comparativa das abordagens citadas.

O capítulo 3 se divide em oito itens:

- O item 3.1 apresenta a metodologia KADS e um breve histórico da mesma.

- O item 3.2 apresenta os conceitos básicos implícitos a essa metodologia.

- O item 3.3 trata das etapas prescritas pela metodologia KADS.

- O item 3.4 descreve os modelos usados pela metodologia.

- O item 3.5 apresenta a linguagem de modelagem utilizada pelo KADS.

- O item 3.6 refere-se ao processo de criação de modelos dentro do KADS.

- No item 3.7 são apresentadas as ferramentas desenvolvidas dentro do projeto KADS.

- O item 3.8 resume as idéias vistas no capítulo e as perspectivas da metodologia KADS.

O capítulo 4 se divide em seis partes:

- O item 4.1 apresenta os principais conceitos abordados pelo capítulo.

- O item 4.2 trata do processo de diagnose e como modelá-lo.

- O item 4.3 refere-se à modelagem do equipamento que deverá ser diagnosticado.

- No item 4.4 aborda-se o problema da interação entre o SE e o seu usuário.

- O item 4.5 resume os itens anteriores e apresenta o modelo a ser empregado pela ferramenta.

- O item 4.6 sintetiza os conceitos apresentados no capítulo.

O capítulo 5 se divide em três partes:

- O item 5.1 apresenta as noções gerais relacionadas com a ferramenta de Aquisição de Conhecimento.

- O item 5.2 resume as características desejadas na ferramenta.

- No item 5.3 descreve-se a ferramenta de Aquisição de Conhecimento.

O anexo se divide em três partes:

- No item A.1 apresenta-se a estrutura global da ferramenta e o seu princípio de funcionamento.

- No item A.2 descreve-se a divisão do sistema em módulos e as funções de cada um.

- No item A.3 apresentam-se detalhes sobre a estruturação e uso da ferramenta.

CAPÍTULO II

AQUISIÇÃO DE CONHECIMENTO

2.1. INTRODUÇÃO

A Engenharia do Conhecimento se divide em três atividades: a obtenção e interpretação do conhecimento de certo domínio, sua representação, e a elaboração de técnicas de inferência que permitam usar esse conhecimento. As técnicas de representação e inferência são assuntos amplamente abordados em livros de Inteligência Artificial, porém a Aquisição do Conhecimento é uma área que só agora vem ganhando seu merecido destaque [WESTPHAL 89]. Aquisição de Conhecimento é a atividade combinada de extrair, interpretar e organizar o conhecimento de um especialista [MOTTA 89].

É um fato amplamente reconhecido que o gargalo da criação de SEs está na Aquisição do Conhecimento. Apesar disso, muitos autores ou não abordam esse assunto ou o fazem de maneira superficial. Isto poderia ser entendido se não existissem técnicas apropriadas para tal fim ou se a Aquisição de Conhecimento fosse uma tarefa trivial, ainda que extensa. Veremos, porém, nesse capítulo, que estão disponíveis uma série de técnicas destinadas a adquirir conhecimento sob diferentes condições e que o problema da extração de conhecimento não é de maneira nenhuma simples.

Procuraremos, em suma, nesse capítulo, identificar os problemas existentes na Aquisição de Conhecimento e, em seguida, apresentaremos as principais técnicas para adquirir o conhecimento de especialistas visando a construção de SEs de um modo mais apropriado.

2.2. CARACTERISTICAS DO CONHECIMENTO

Usualmente o Engenheiro do Conhecimento emprega uma de duas técnicas para realizar a Aquisição de Conhecimento: ou ele realiza uma intensa série de entrevistas com o especialista ou procura tornar-se ele mesmo um especialista no assunto desejado para então, através de introspecção, codificar o conhecimento numa linguagem de programação. Em quaisquer dos casos surgem dificuldades oriundas da natureza do conhecimento, pois mesmo o conhecimento relativo a um domínio restrito é composto de informações de diferentes tipos, que requerem diferentes ferramentas para serem tratadas.

O Engenheiro de Conhecimento deve ter em mente essas particularidades relacionadas com o conhecimento para poder manipulá-lo eficientemente. Algumas distinções úteis que podemos fazer acerca do conhecimento são entre conhecimento especializado e de senso comum, conhecimento acadêmico e empírico, articulado e implícito, analógico e simbólico (fig. 2.1) [SCHIAVINI 90].

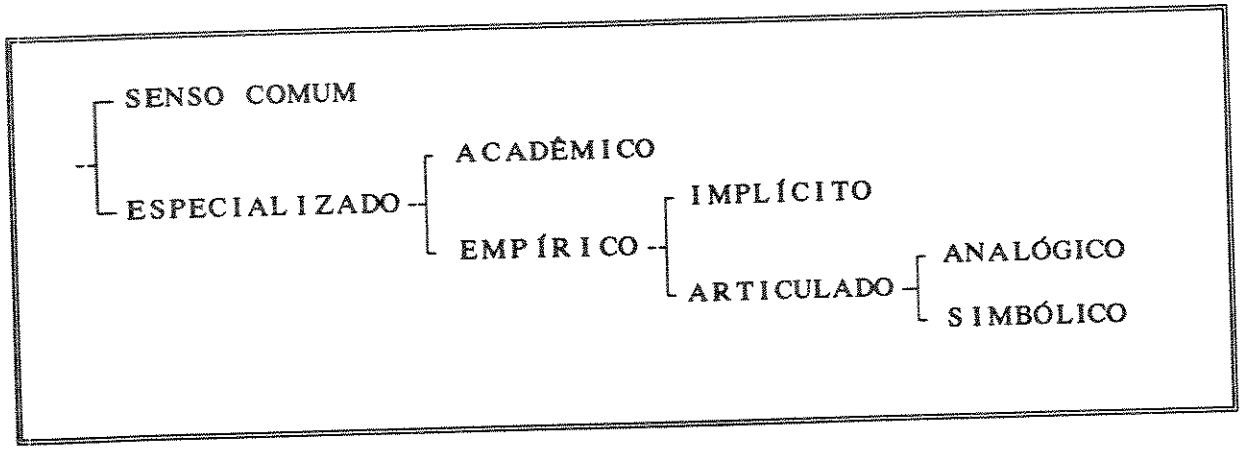


FIGURA 2.1 - TIPOS DE CONHECIMENTO.

O domínio da Inteligência Artificial restringe-se ao conhecimento especializado que pode ser acadêmico ou empírico. O domínio dos SEs é o conhecimento empírico. Esse último pode ser implícito ou articulado, podendo o conhecimento articulado ser analógico ou simbólico. Vale notar que de todos esses "tipos" de conhecimento a maioria dos SEs existentes lida apenas com o conhecimento articulado simbólico [HAYES-ROTH 84].

2.2.1. Conhecimento especializado e de senso comum

Um dos requisitos para que um conhecimento possa ser transportado para um SE é que ele não contenha senso comum, pois esse tipo de conhecimento é de difícil tratamento, tanto quanto à sua extração como quanto à sua representação. Em geral, entende-se senso comum como sendo o conhecimento que é por demais óbvio para ser explicitado, motivo que leva as pessoas a o omitir em suas explicações. O grande desafio imposto pelo senso comum é que ele não é um conhecimento delimitado, tal qual o especializado, o que tende a tornar as tentativas de seu tratamento muito extensas.

Na prática, conhecimento de senso comum é aquele que está fora da alçada do especialista. Assim, para um médico assumir que uma criança com peso abaixo do normal é filha de pais de baixa renda, e agir de acordo com tal hipótese, é um raciocínio de senso comum num país subdesenvolvido. Não é necessário que o Engenheiro de Conhecimento se preocupe com esse tipo de conhecimento a não ser que ele pretenda usar seu SE em situações com as quais o especialista não está familiarizado, aonde o seu senso comum poderá não se aplicar.

2.2.2. Conhecimento acadêmico e empírico

O conhecimento acadêmico é aquele que encontramos nos manuais e nos livros; tal conhecimento é mais ou menos formalizado e apresenta uma estrutura dedutiva. Já o empírico caracteriza-se por uma série mal estruturada de heurísticas, que se orientam no sentido do "como fazer" e não do "o que é". Os problemas não podem ser facilmente resolvidos de acordo com os cânones acadêmicos por que tal técnica, em geral, demanda um quantidade enorme de informações que não estão disponíveis. Além disso, os possíveis procedimentos a serem seguidos na solução dos problemas normalmente não são claramente explicitados no conhecimento acadêmico, de tal maneira que esse conhecimento só é adequado em situações em que existam poucos cursos de ação claramente definidos [KASSIRER 82].

Um problema que se enfrenta para se obter o conhecimento empírico desejado pelo Engenheiro de Conhecimento é que os indivíduos, ao serem questionados acerca de qualquer assunto, tendem sempre a fornecer, caso a conheçam, a visão acadêmica da questão que julgam mais correta e elegante. Um especialista poderá mesmo se envergonhar de empregar uma abordagem prática em seu trabalho por não ser capaz de justificá-la formalmente. Cabe ao Engenheiro de Conhecimento explicar ao especialista que é no seu conhecimento prático que ele está interessado e que sua habilidade é um feito e não uma falta.

2.2.3. Conhecimento articulado e implícito

As pessoas nem sempre têm consciência do que sabem: freqüentemente certo conhecimento só se manifesta quando realizamos determinada atividade e, se indagados sobre um tal comportamento, poderemos ser incapazes de explicar as razões que nos levaram a agir da forma como o fizemos. Isto é o que se chama de conhecimento implícito, em contraposição ao conhecimento articulado (o qual somos capazes de explicar).

Diversos experimentos demonstram que o conhecimento sobre como fazer é muito independente do conhecimento sobre como explicar [BERRY 87] [KINTSCH 77]. O conhecimento implícito, porém, não é meramente um conhecimento que não pode ser explicado verbalmente mas sim um conhecimento do qual não estamos cientes. Por exemplo, ainda que sejamos incapazes de ensinar uma pessoa a amarrar seu sapato por meio de palavras, ainda assim poderemos mostrar por meio de ações, ou desenhos, como fazê-lo. Isto é diferente do conhecimento realmente implícito que o especialista não é capaz de explicar por quaisquer meios. A maioria das pessoas, por exemplo, sabe como falar de maneira sintaticamente correta sem no entanto ter conhecimento completo das regras de sintaxe necessárias para tanto.

O conhecimento implícito representa uma das grandes dificuldades do Engenheiro de Conhecimento, pois de nada adianta pedir ao especialista informações sobre ele por que, ou o especialista admitirá ser incapaz de fornecê-las ou, pior, fornecerá informações enganosas. O entrevistador poderá mesmo fazer o especialista se sentir ameaçado por não conseguir fornecer respostas para questões que aparentam ser bastante razoáveis, levando-o a ver a

si mesmo como irracional e tornando-o inseguro, o que dificultará ainda mais o processo de extração de conhecimento.

2.2.4. Conhecimento simbólico e analógico

A abordagem da Inteligência Artificial é voltada para a análise de métodos computacionais e conseqüentemente, o tratamento e solução dos seus problemas é feito através do processamento simbólico. Entretanto, as evidências mostram que a resolução de problemas por especialistas depende fundamentalmente do reconhecimento de padrões e de imagens [KASSIRER 82] [KOLODNER 84]. O que caracteriza um especialista é principalmente uma capacidade de ver padrões conhecidos nos novos problemas. Os enxadristas, por exemplo, não se diferenciam dos novatos em sua capacidade de analisar as jogadas um certo número de passos à frente ou em reter informações. Eles se distinguem por terem memorizado grande quantidade de configurações de xadrez e poderem rapidamente codificar a situação corrente em padrões previamente vistos. A escolha do próximo movimento se resumirá, então, a um pequeno conjunto de movimentos sabidamente satisfatórios. Já o novato não possui essa capacidade de filtrar os maus movimentos, fazendo-o se perder em suas análises [CHASE 73].

Os especialistas, após muitos anos de prática, findam por organizar seu conhecimento de maneira a reduzir uma longa cadeia de raciocínios a uma mera associação imediatamente acessível [KUIPERS 84]. Isto faz com que tenham dificuldade em verbalizar a informação que realmente usam para resolver os problemas e leva ao chamado paradoxo da Engenharia do Conhecimento: quanto mais competente é um especialista, menos ele é capaz de descrever seu conhecimento

[CARNOTA 87]. O Engenheiro de Conhecimento deve portanto estar ciente que o especialista com maior habilidade poderá, eventualmente, não ser o mais indicado para ter seu conhecimento adquirido.

2.3. TÉCNICAS PSICOLÓGICAS DE EXTRAÇÃO DE CONHECIMENTO

Existem quatro abordagens para lidar com as dificuldades citadas: a abordagem psicológica, a abordagem por máquinas de aprendizado, a baseada em linguagens e a baseada em ferramentas [SCHIAVINI 90]. A primeira que veremos é a psicológica.

De certa maneira o Engenheiro de Conhecimento atua no processo de Aquisição de Conhecimento, como um psicólogo que deve captar modos de solução de problemas de um especialista humano. A própria interação entre ele e o especialista pode ser melhor compreendida à luz das teorias psicológicas. Sendo assim, é natural que o Engenheiro de Conhecimento adapte e aplique técnicas que foram originalmente desenvolvidas pelos psicólogos. Essas técnicas, que impõem disciplina e rigor no processo de extração, permitem obter diferentes "tipos" de conhecimento.

Recomenda-se que o emprego dessas técnicas seja precedido de uma apresentação por parte do especialista sobre sua área de atuação. O entendimento e a análise dessa apresentação fornecem ao Engenheiro de Conhecimento uma estrutura para aplicação das técnicas, uma visão geral do assunto e um primeiro contato com o vocabulário do domínio (jargão do especialista).

Descreveremos agora 18 dessas técnicas, cada uma mais adequada para certa situação, que dividimos, assim como Olson [OLSON 87], em dois grandes grupos: as técnicas diretas (voltadas para o conhecimento articulado) e as indiretas (voltadas para o conhecimento implícito). Nas técnicas diretas o especialista é diretamente solicitado a fornecer o conhecimento desejado. Nessa classe se incluem as entrevistas informais, as hipóteses terminais, a técnica adaptada de tempestade cerebral, a codificação por lista, os questionários, as curvas fechadas, os diagramas, o fluxo de inferências, a observação da realização da tarefa, a análise de descrição verbal e o fluxo de inferências.

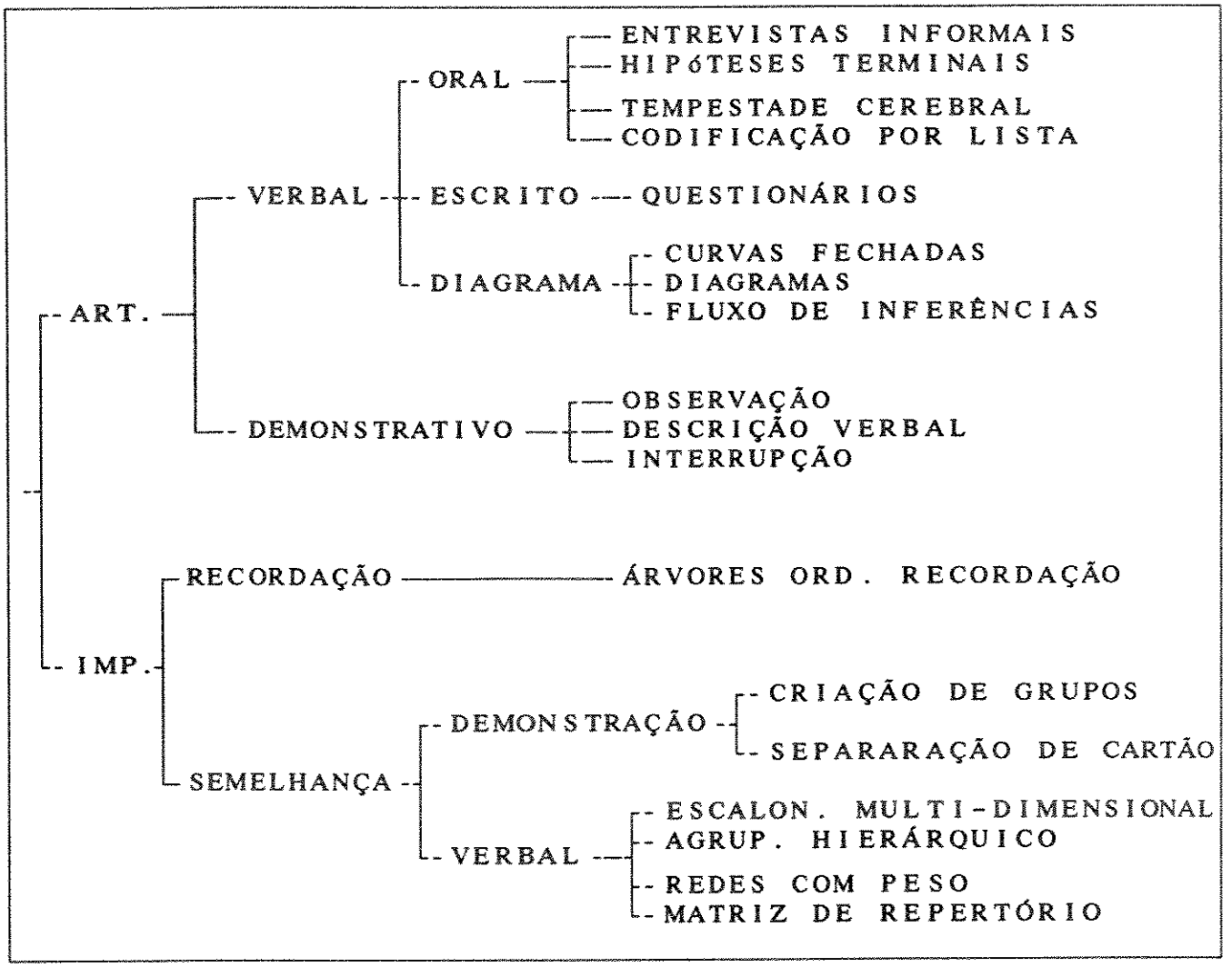


FIGURA 2.2 - TÉCNICAS DE AQUISIÇÃO DE CONHECIMENTO.

Já nas técnicas indiretas não é assumida a capacidade do especialista explicar por meios verbais ou ações, o conhecimento que possui. Entre as técnicas indiretas encontramos as árvores ordenadas por recordação, a criação de grupos, a separação de cartões, o escalonamento dimensional, o agrupamento hierárquico, as redes com pesos e a análise por matriz de repertório. Na figura 2.2 propomos uma taxonomia baseada no processo de extração básico utilizado por essas técnicas [SCHIAVINI 90].

2.3.1. Técnicas diretas (articuladas)

Vamos agora descrever sucintamente cada uma dessas técnicas. Começaremos com as que lidam com o conhecimento articulado.

2.3.1.1 Entrevistas informais

A mais conhecida e mais freqüentemente empregada técnica de Aquisição de Conhecimento é a técnica das entrevistas. Convém que as entrevistas sejam gravadas e/ou anotadas para posterior análise. Uma entrevista pode ser informal ou estruturada, dependendo do enfoque considerado [WIELLINGA 88] [GIORNO 88]. Alguns cuidados devem ser tomados para aplicar a técnica das entrevistas informais.

A primeira coisa a fazer é conseguir a cooperação do especialista. Várias são as razões que podem levar o especialista a não querer colaborar. Ele poderá temer que ao revelar suas técnicas diminua a admiração que os outros têm por ele, ou poderá não saber a forma exata como procede para resolver os problemas e não quererá admitir isto ou ainda, ele simplesmente poderá ter medo de perder seu emprego se seu conhecimento for transferido para um computador.

Convém, portanto, antes de iniciar uma série de entrevistas, ter um diálogo aberto com o especialista para descobrir se algum desses receios está presente, para então tentar esclarecer o especialista acerca do assunto, evitando assim muita perda de tempo.

Tendo conseguido essa colaboração, deve-se começar fazendo perguntas genéricas para então ir aumentando a especificidade. O primeiro objetivo que deve ser atingido é fazer um levantamento do vocabulário usado pelo especialista. Isso pode ser atingida por meio de perguntas como [OLSON 87]:

- "O que você deseja saber antes de começar a ponderar sobre o problema?";

- "Quais fatos ou hipóteses você tenta estabelecer quando pensa acerca do problema?";

- "Que fatores influenciam a forma como você raciocina sobre o problema?".

- "Que tipo de valores esse objeto pode assumir? Qual o intervalo de valores possíveis?"

- "Esse fator depende de outros fatores? Quais?"

- "Esse fator é necessário para resolver todos os problemas do domínio ou apenas alguns?"

Numa entrevista informal não se deve impor o próprio entendimento ao especialista. Deve-se deixar que ele fale mesmo que o assunto pareça tangencial ao objetivo. O Engenheiro de Conhecimento deverá apenas perguntar o que não foi entendido, mas não impor sua própria forma de ver o problema ao especialista.

O grande atrativo das entrevistas informais é que elas são muito simples de serem feitas e fornecem uma avaliação dos problemas associados ao domínio. Porém, elas tendem a ser ineficientes, pela predisposição do especialista em divagar e também pela dificuldade de conversão das informações obtidas numa representação útil.

2.3.1.2. Técnica das hipóteses terminais

Nas entrevistas estruturadas ou focalizadas busca-se aumentar a eficiência do processo, preparando-se previamente perguntas com escopos específicos. Uma das formas de estruturar a entrevista é a aplicação do Método das Hipóteses Terminais [GONÇALVES 86]. Nesse, as

regras são obtidas por meio de encadeamento para trás, ou seja, a partir da solução final dos problemas em direção aos fatos iniciais. As perguntas predefinidas geralmente são do tipo "Como você conclui isto?", cujas respostas servem novamente como "âncora" para outras perguntas, num processo repetitivo. Deve-se primeiro determinar o conjunto de possíveis soluções dos problemas pertinentes para que então o especialista explicite quais fatos levam a estas soluções (figura 2.3).

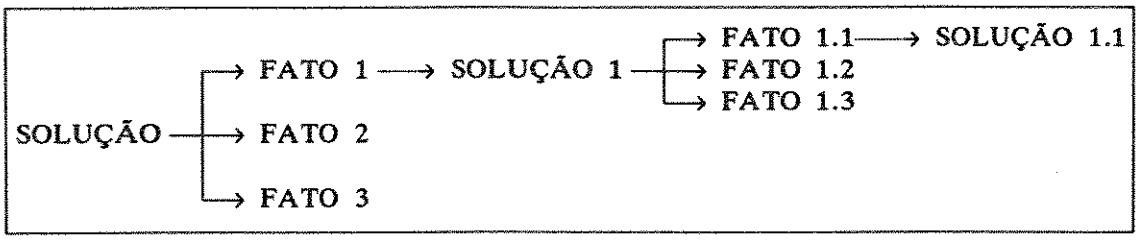


FIGURA 2.3 - EXEMPLO DE APLICAÇÃO DO MÉTODO DAS HIPÓTESES TERMINAIS.

A vantagem dessa técnica é que ela direciona a entrevista evitando divagações e abstrações desnecessárias. Obtêm-se de uma maneira direta regras de produção. A dificuldade consiste em identificar o espaço de possíveis soluções.

2.3.1.3. Técnica adaptada de tempestade cerebral

As técnicas de Aquisição de Conhecimento são normalmente dirigidas à extração do conhecimento de apenas um especialista. Existem casos, porém, em que é interessante integrar o conhecimento vindo de diferentes especialistas. Nesse aspecto é muito útil empregar o Método Adaptado de Tempestade Cerebral [GONÇALVES 86]. Esse método reúne uma equipe de especialistas que, em um processo de tempestade cerebral, geram regras de produção na forma sintática "SE

condições ENTÃO ações ". As regras geradas devem permanecer visualmente disponíveis ao grupo, de modo a facilitar refinamentos e a inspirar a geração de novas regras. O conjunto dessas regras deve passar por um posterior refinamento, feito pelo Engenheiro de Conhecimento e por um especialista, pois elas podem se apresentar de forma desordenada e conter redundâncias.

Com essa técnica geram-se regras de uma maneira simples e eficiente. Porém tais regras são pouco metódicas, exigindo um grande trabalho posterior para refiná-las.

2.3.1.4. Técnica adaptada de codificação por lista

A técnica adaptada de codificação por lista [GRECO 87] [LEÃO 87] é muito eficiente para diagnósticos. Ela consiste em obter uma lista de sinais, sintomas e testes, assim como uma lista de diagnósticos. O especialista escolhe um diagnóstico e o associa com os sinais, sintomas e testes correspondentes. Obtém-se assim uma sublista. Em seguida o especialista ordena cada um dos elementos dessa sublista de acordo com sua importância para o diagnóstico.

Esta sublista ordenada é usada para construir uma árvore orientada que começa nos elementos de maior importância e termina no diagnóstico (fig. 2.4). Durante esse processo o especialista é naturalmente levado a criar nós intermediários e a empregar conectores lógicos adequados ("E" e "OU"). Após a construção desse grafo o especialista associa pesos a cada nó de acordo com sua importância para a diagnose.

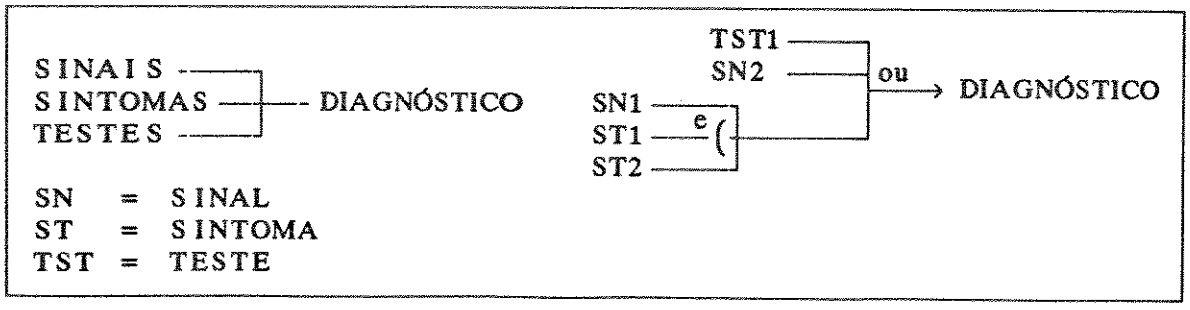


FIGURA 2.4 - EXEMPLO DE GRAFO DO MÉTODO DA CODIFICAÇÃO POR LISTA.

2.3.1.5. Questionários

As entrevistas têm a vantagem de permitir que se obtenham informações inesperadas. O especialista decide a ordem em que ele aborda os temas e o detalhe em que ele o faz. As entrevistas, porém, demoram um tempo excessivo. Já os questionários são uma forma mais eficiente de obter informações; eles permitem que o especialista forneça informações num clima mais descontraído e relaxado. Esses questionários são cartões onde estão escritas perguntas padronizadas, porém abertas, como exemplificado na figura 2.5.

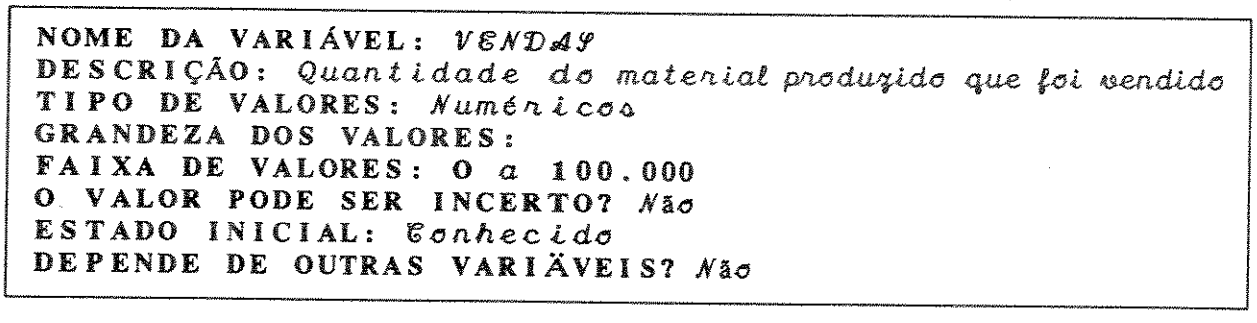


FIGURA 2.5 - CARTÃO QUESTIONÁRIO PARA ELICITAR VARIÁVEIS.

Os questionários são muito úteis em descobrir quais são os conceitos empregados pelo especialista, suas relações e em determinar possíveis incertezas nas conclusões.

2.3.1.6. Desenho de curvas fechadas

As técnicas até agora citadas destinam-se a revelar o conteúdo do pensamento do especialista durante a solução de um problema. Elas fornecem os conceitos e as relações relevantes para lidar com os objetos e as inferências realizadas. Essas técnicas não fazem qualquer hipótese acerca da estrutura do conhecimento extraído. Já a técnica do desenho de curvas fechadas destina-se a indicar as relações entre os objetos e a forma como eles se organizam dentro do espaço do problema.

O especialista é solicitado a indicar quais conceitos, relativos a um dado problema, estão associados, desenhando uma curva fechada em torno deles. Com isso pode-se determinar as relações entre os objetos e a forma como eles se organizam dentro do espaço do problema. Tal técnica é aplicável, por exemplo, à análise de fórmulas, de figuras num monitor ou à análise feita por um especialista no jogo "GO" (figura 2.6). O desenho de curvas fechadas é muito adequado para lidar com o conhecimento analógico, relacionado ao reconhecimento de padrões.

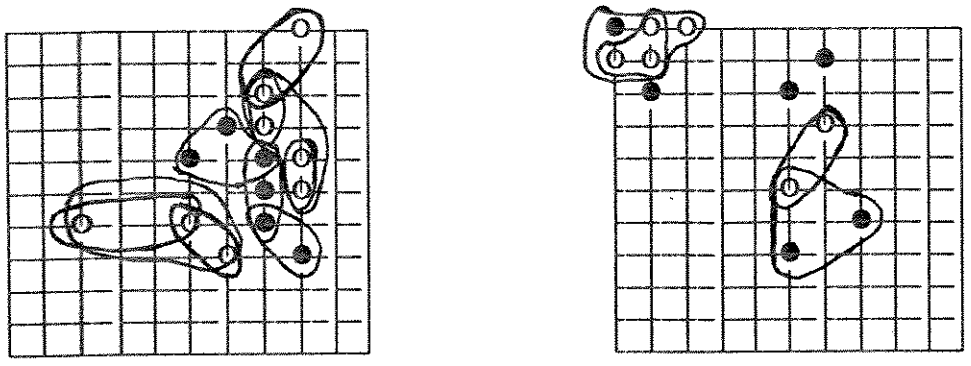


FIGURA 2.6 - CURVAS FECHADAS DESENHADAS POR UM ESPECIALISTA EM GO.

2.3.1.7. Desenho de diagramas

O objetivo aqui é obter uma representação organizada dos conceitos do domínio do especialista. Parte-se de certo elemento relacionado ao problema que é decomposto em seus subelementos constituintes, e assim sucessivamente com esses subelementos. O diagrama resultante poderá ser útil para o desenvolvimento de outras técnicas de Aquisição de Conhecimento (figura 2.7).

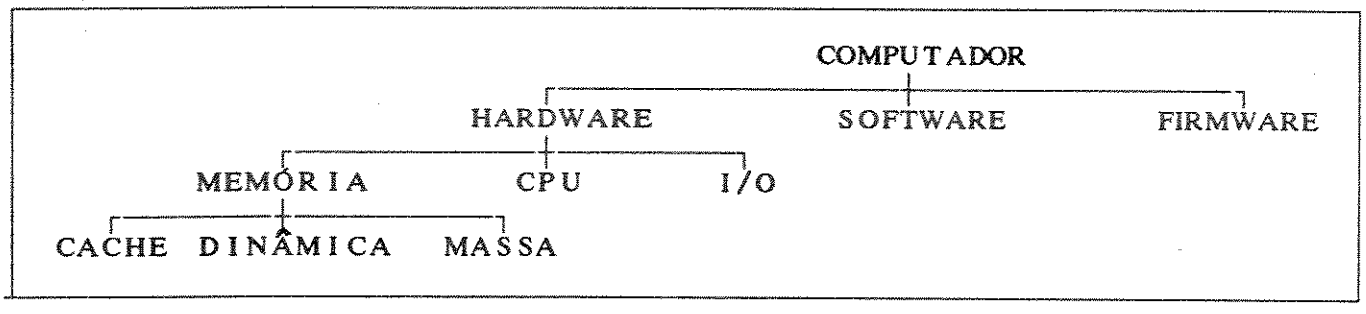


FIGURA 2.7 - DIAGRAMA DE UM COMPUTADOR

2.3.1.8. Análise por fluxo de inferências

Essa técnica busca relacionar os diversos conceitos usados pelo especialista através da construção de uma rede na qual os conceitos

estejam interligados por meio de setas associadas a pesos que vão de -1 a 1, conforme haja uma relação direta ou inversa entre os objetos. Começa-se com uma lista de conceitos relevantes, sendo que para cada par desses conceitos, o especialista é solicitado a revelar qual a relação entre os dois (figura 2.8).

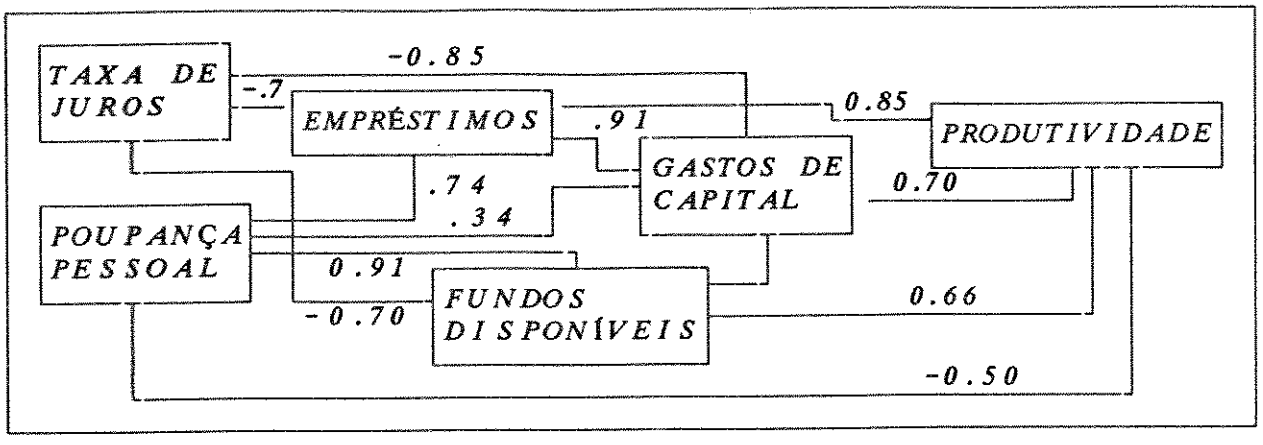


FIGURA 2.8 - EXEMPLO DE UMA ANÁLISE POR FLUXO DE INFERÊNCIAS.

Essa técnica é simples de aplicar e útil para mostrar ao especialista as características de suas habilidades que ele já esclareceu. Entretanto, a manipulação de listas de fatos extensas é problemática e poderá haver dificuldades em lidar simultaneamente com objetos de diferentes níveis de abstração.

2.3.1.9. Observação da realização da tarefa

Ao invés de questionar o especialista, podemos observar como ele resolve seus problemas e tentar inferir a partir daí as técnicas e processos empregados. Para que uma análise como esta possa ser bem sucedida, o Engenheiro de Conhecimento deve estar bem familiarizado com a área para que possa entender a tarefa do especialista, ainda que não seja capaz de realizá-la. Além de observar, deve-se tomar

notas e tentar inferir o raciocínio usado e, eventualmente, fazer um videoteipe do processo para, mais tarde, discuti-lo com o especialista. Nesse último caso devemos ter em mente que o especialista não possui uma capacidade ilimitada de recordar as razões que o levaram a adotar certo curso de ação.

Caso se use uma técnica de simulação, a solução dos problemas apresentados ao especialista deve abranger uma amostragem representativa. Caso se resolva observar o especialista atuando em campo, ao invés de empregar uma simulação, seu comportamento deve ser acompanhado por tempo suficientemente longo para cobrir uma amostra representativa de atividades, o que obviamente consumirá um tempo considerável.

Tarefas especialmente alteradas podem ser empregadas para se obter informações normalmente mascaradas. Uma forma de alterar as tarefas é limitar seu conteúdo de informação, buscando mudar a forma como o especialista resolve normalmente seus problemas. Outra forma de restringir a realização de uma tarefa é impor um limite de tempo na sua execução. Na técnica chamada de protocolo do telefone, o especialista é impedido de ver o problema que está resolvendo. Nesse caso o especialista será forçado a solucionar o problema por meio de uma série de perguntas. A desvantagem de empregar tarefas alteradas é que elas são desconfortáveis para o especialista que poderá se sentir sob pressão.

2.3.1.10. Análise de descrição verbal (Protocol Analysis)

Além de observar o especialista trabalhar podemos também, como reforço, solicitar a ele que, enquanto trabalha, pense em voz alta,

buscando esclarecer seus objetivos, suas técnicas e o que está vendo a cada momento, gravando-se as descrições obtidas. Esse procedimento é chamado de análise de descrição verbal [KUIPERS 84]. Após a solução do problema e a correspondente transcrição da descrição verbal em um texto, o Engenheiro de Conhecimento questiona o especialista quanto ao seu comportamento e ao conteúdo do texto, e efetua uma análise de texto procurando extrair regras de produção. Na figura 2.9, apresentamos um exemplo de descrição verbal, no caso do processo mental envolvido na solução do criptograma "DONALD + GERALD = ROBERT", aonde cada letra corresponde a um dígito entre 0 e 9, sem repetição, partindo-se da informação "D=5" [OLSON 87].

As técnicas de observação têm como falha serem incompletas pois embora informem sobre como o conhecimento é usado, não estabelecem os limites desse conhecimento. Se algo não é mencionado não significa que o especialista o desconheça. Além disto os especialistas não são capazes de verbalizar tão rapidamente quanto raciocinam, poderão pular passos na verbalização de seus raciocínios ou omitir coisas que julguem óbvias. O próprio Engenheiro de Conhecimento ao analisar o desempenho do especialista dificilmente conseguirá inferir todos os aspectos do conhecimento em questão, principalmente com relação às tarefas mais complexas que dificilmente serão decompostas em todas as suas componentes.

```
  DONALD
+ GERALD
--------
 ROBERT

D = 5 !
```

Cada letra tem um e apenas um valor numérico...
Há dez letras diferentes e cada uma delas tém um valor numérico
Assim eu posso, olhando para os dois Ds...
Cada D é cinco, assim T é zero.
Assim eu penso que começarei escrevendo o problema.
Escreverei cinco e cinco são zero. Agora tenho outros Ts? Não.
Mas eu tenho outro D..
Isto significa que eu tenho um cinco do outro lado.
Agora eu tenho dois As e dois Ls que estão em algum lugar e este D... três Rs...
Dois Ls é igual a um R, naturalmente estou carregando um. O que significa que R tem de ser um número impar.
Assim R pode ser um, três, não cinco, sete ou nove.
Agora G. Já que R irá ser um número impar e D é cinco, G tem de ser um número par.
Estou olhando para o lado esquerdo do problema, aqui aonde diz D+G. Oh! Mais possivelmente outro número, se eu tenho que carregar um de E + O. Eu penso que irei esquecer acerca disto em um minuto...

FIGURA 2.9 - PROTOCOLO DE RESOLUÇÃO DE UM CRIPTOGRAMA.

A vantagem dessa técnica sobre a anterior é que não há demora entre o ato de pensar e o de descrever. A técnica de descrição verbal, porém, só é aplicável em tarefas aonde a resolução do problema possa ser
acompanhada de uma verbalização. Muitas tarefas não se encaixam nessa categoria. Há aquelas em que o especialista emprega uma linguagem idiossincrática, que se verbalizada induziria a distorções e erros. Há também tarefas que não são acompanhadas de qualquer espécie de verbalização como, por exemplo, as tarefas percepto-motoras.

2.3.1.11. Análise por interrupção

A análise de descrição verbal interfere com o processo natural de resolução do problema. Uma forma de evitar isto é deixar o

especialista trabalhar pela forma natural, sem pensar em voz alta, e interrompe-lo apenas quando não mais se conseguir entender a técnica empregada. Neste momento deve-se perguntar em detalhe porque ele agiu da maneira como o fez, tentando obter no momento o seu foco de atenção e o tipo de inferências feitas.

2.3.2. **Técnicas indiretas (implícitas)**

Freqüentemente o especialista percebe relações complexas ou chega a soluções adequadas sem no entanto saber exatamente como o fez. Nem sempre os métodos empregados pelo especialista são articuláveis. Em tais casos deve-se usar técnicas indiretas de extração.

Cada uma das 7 técnicas de Aquisição indireta que apresentaremos assume uma hipótese acerca de como o conhecimento do especialista está organizado: listas, redes, tabelas, etc. Algumas entrevistas preliminares podem esclarecer qual é tal organização.

2.3.2.1. Árvores ordenadas por recordação

O elemento de partida dessa técnica são tentativas sucessivas do especialista recordar conceitos. Busca-se com isto organizar os conceitos em agrupamentos hierárquicos usando a hipótese de que os conceitos que estão associados mentalmente num mesmo agrupamento tendem a ser recordados juntos. Assim a ordem em que os conceitos são lembrados pelo especialista pode ser usada para inferir a forma como ele os estrutura (figura 2.10).

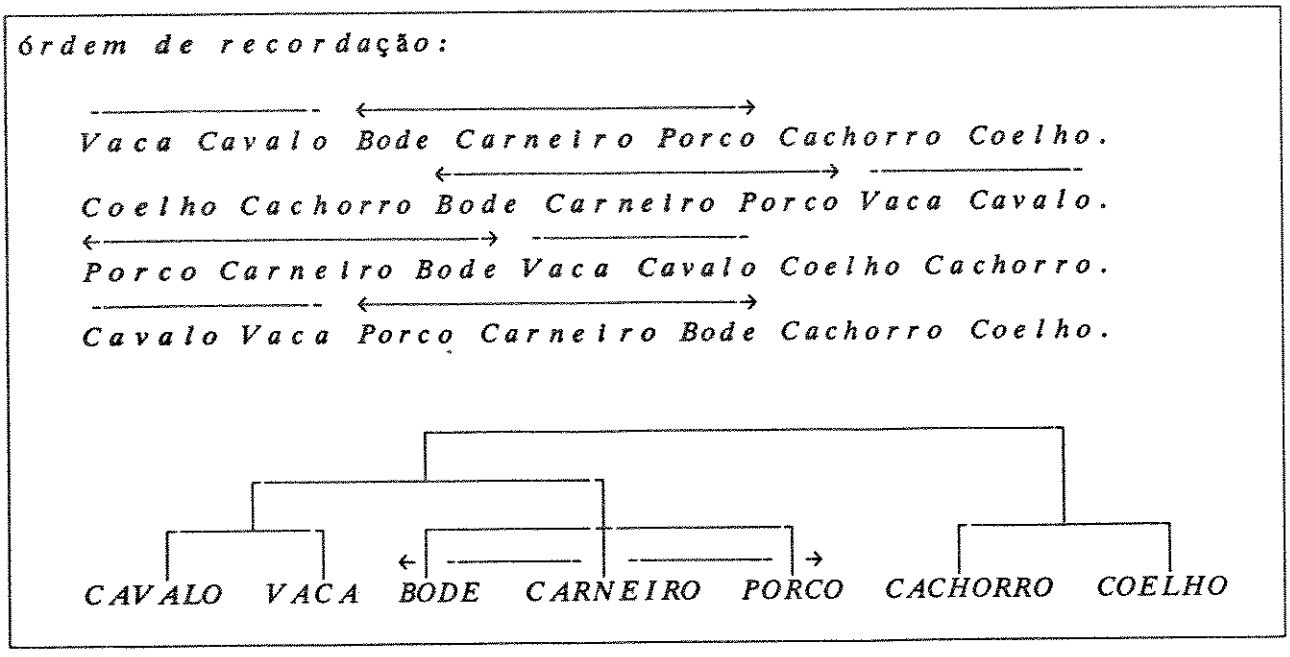


FIGURA 2.10 - EXEMPLO DE ÁRVORE ORDENADA POR RECORDAÇÃO.

2.3.2.2. Técnica de criação de grupos

Obtêm-se inicialmente o vocabulário do domínio. Transcreve-se alguns elementos escolhidos para cartões. O especialista é solicitado a juntar os cartões com características relacionadas os quais formam grupos que devem ser nomeados. Numa segunda etapa esses grupos, e os cartões que permaneceram isolados, passam novamente pelo processo de agrupamento, formando um segundo nível de agrupamento. O processo prossegue enquanto o especialista julgar útil.

2.3.2.3. Técnica de separação de cartões

Criam-se cartões com elementos do domínio do problema, da mesma forma que na técnica anterior. Esses cartões são então divididos em dois grupos pelo especialista. Em seguida, o mesmo conjunto inicial de cartões é dividido em três grupos, e após em quatro, cinco, etc.,

tantos quantos se julgar significativo. O especialista deve então organizar hierarquicamente os diversos grupos obtidos.

2.3.2.4. Escalonamento multi-dimensional

Essa técnica assume que o especialista organiza os conceitos que emprega em uma estrutura análoga ao de um espaço de n dimensões. Para descobrir como eles se posicionam nesse espaço, o especialista avalia a semelhança entre pares de conceitos. A partir dessa análise determina-se o melhor posicionamento dos conceitos num espaço com certo número de dimensões (figura 2.11), e o erro associado a tal representação.

	VACA	*CARNEIRO*	*PORCO*	*CAVALO*	*CACHORRO*	*COELHO*
BODE	*1*	*1*	*3*	*4*	*10*	*11*
VACA	*X*	*1*	*3*	*3*	*9*	*12*
CARNEIRO		*X*	*2*	*4*	*6*	*10*
PORCO			*X*	*8*	*8*	*9*
CAVALO				*X*	*6*	*6*
CACHORRO					*X*	*2*
COELHO						*X*

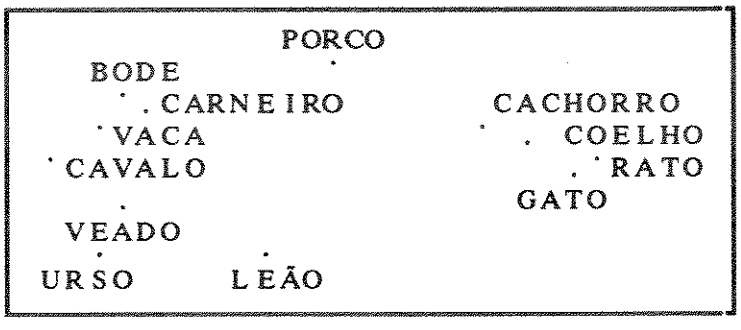


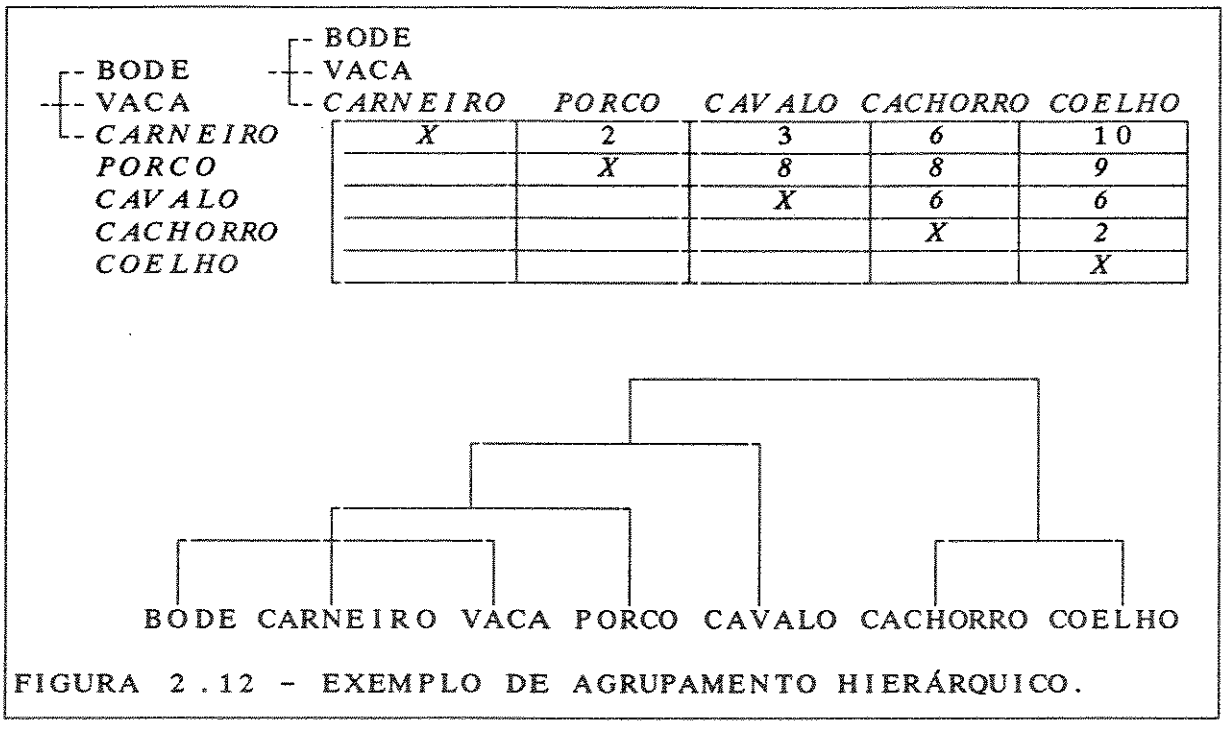
FIGURA 2.11 - EXEMPLO DE ESCALONAMENTO MULTI-DIMENSIONAL.

Uma das dificuldades dessa técnica é que é muito cansativo para o especialista julgar a semelhança entre cada par de objetos. Além

disso é difícil saber qual é o número adequado de dimensões e como analisar a representação obtida.

2.3.2.5. Agrupamento hierárquico de Johnson

Essa técnica começa da mesma forma que a anterior: coletando a avaliação das semelhanças entre cada par de objetos do domínio. A partir dessa avaliação, agrupam-se os objetos que estão mais próximos entre si. Cada agrupamento passa a ser considerado um novo objeto que substitui os seus constituintes na análise, que prossegue a partir da nova representação, até só restar um elemento. Consegue-se com isso produzir uma estrutura de agrupamentos hierarquicamente organizada (figura 2.12).



	CARNEIRO	PORCO	CAVALO	CACHORRO	COELHO
BODE	X	2	3	6	10
VACA		X	8	8	9
CARNEIRO			X	6	6
PORCO				X	2
CAVALO					X
CACHORRO					
COELHO					

FIGURA 2.12 - EXEMPLO DE AGRUPAMENTO HIERÁRQUICO.

Essa técnica é muito simples de implementar e interpretar. Porém a obtenção das avaliações entre os pares de objetos é tão cansativa quanto na técnica anterior.

2.3.2.6. Redes com peso

Assim como nas duas técnicas anteriores, o especialista fornece uma avaliação da semelhança entre cada par de objetos [OLSON 87]. A análise do resultado, porém, assume que essa avaliação resulta de uma rede de associações entre eles, tendo cada associação um peso. Começa-se estabelecendo um único caminho entre cada objeto do domínio. A partir desta rede novas associações são acrescentadas aonde a distância observada entre os conceitos é menor do que a inicialmente obtida (figura 2.13).

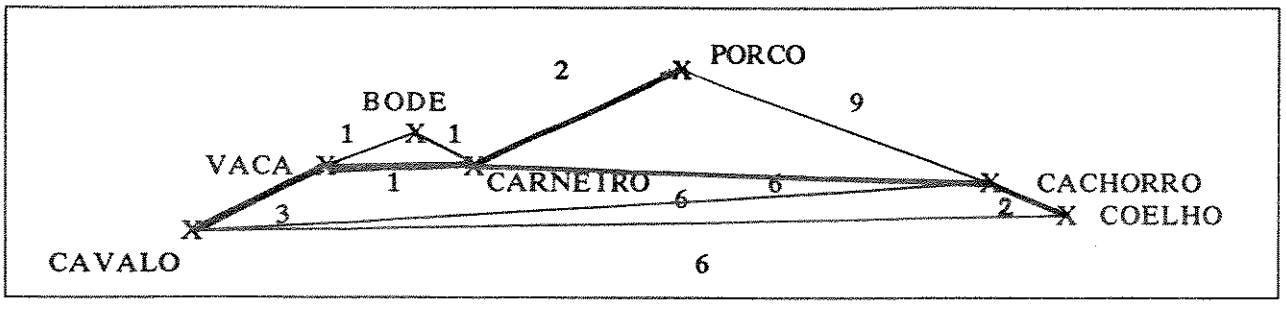


FIGURA 2.13 - EXEMPLO DE REDE COM PESOS.

Consegue-se assim identificar certas características da estrutura do conhecimento do especialista, tais como os conceitos dominantes e os cíclicos. Os conceitos dominantes são aqueles que possuem um grande número de ligações com os demais, os cíclicos são os completamente unidos entre si.

2.3.2.7. Análise por matriz de repertório

É a mais completa das técnicas de extração de conhecimento implícito. Inicialmente o especialista nomeia os objetos pertinentes ao seu campo e as características (as dimensões) em que diferem.

COMPARAÇÃO DOS ESTUDANTES DE UM CURSO

		E1	E2	E3	E4	E5	E6	E7
C1	IMATURO	3	1	3	1	1	1	1
C2	MÁS QUALIFICAÇõES	1	2	1	2	3	2	2
C3	INSEGURO	3	1	1	2	3	1	3
C4	FAZ PERGUNTAS	1	3	2	2	1	3	2
C5	BOA FREQUÊNCIA	1	3	3	1	2	2	3
C6	FAZ AS LIÇõES	1	3	2	1	1	3	1
C7	SOCIÁVEL	1	3	3	2	2	3	1
C8	DESORGANIZADO	3	1	1	2	3	2	1
C9	BRIGÃO	3	1	1	2	3	3	1

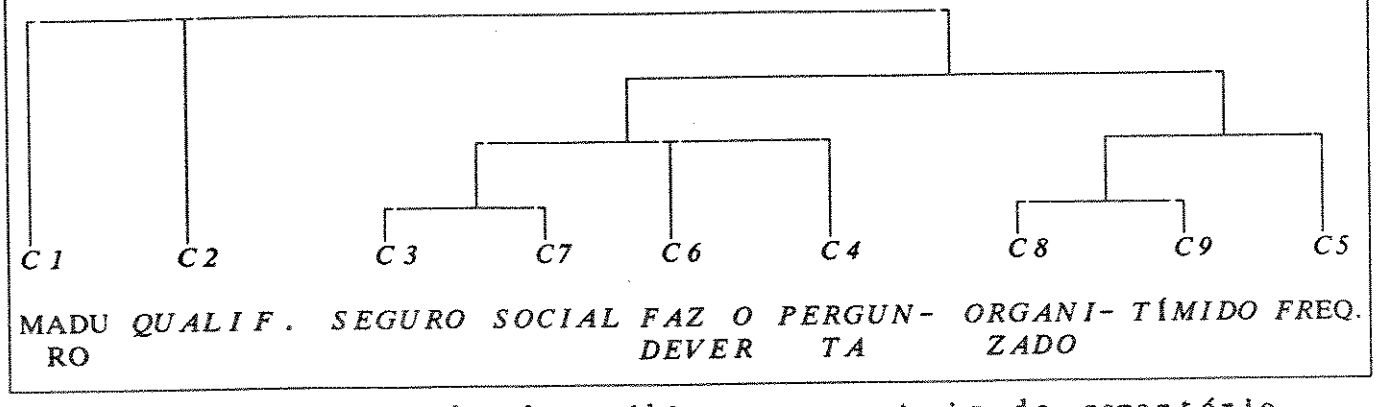


FIGURA 2.14 - Exemplo de análise por matriz de repertório.

As dimensões obtidas e os conceitos formam uma matriz de análise. O especialista então avalia todos os objetos de acordo com cada uma das dimensões desta matriz. Os valores obtidos são submetidos a uma análise pela técnica de agrupamento hierárquico de Johnson que revela a forma como se agrupam os objetos entre si. Além disso, as próprias dimensões levantadas são analisadas, para se descobrir como elas se organizam e se agrupam (figura 2.14). A partir desta análise, determina-se quais são as dimensões altamente correlacionadas, quais implicam umas nas outras e aquelas que são hierarquicamente superiores às demais. Pode-se com isso gerar regras que refletem correlações entre as dimensões, e a organização hierárquica delas, por exemplo, no exemplo da figura 2.14, podemos

concluir que: "se alguém é seguro então ele é sociável". Protótipos de SEs podem ser diretamente obtidos por essa técnica.

Uma das vantagens da análise por matriz de repertório é que ela permite avaliar a similaridade entre os objetos do domínio sem ser necessário passar pelo tedioso processo de avaliar par a par as semelhanças.

2.4. AQUISIÇÃO DE CONHECIMENTO POR APRENDIZAGEM

Se fosse possível construir programas capazes de aprender, então o gargalo da construção de SEs estaria definitivamente superado. Existem muitos debates acerca da viabilidade da construção de tais programas. Apesar disso, esse é um campo de estudo muito ativo, que tem apresentado resultados práticos na forma de sistemas com capacidade de aprendizado.

Entre as técnicas mais comuns empregadas para construir máquinas capazes de aprender, temos: aprendizagem por roteamento, por ajuste de parâmetros, aprendizagem pelo que foi dito, aprendizagem por analogias, redes neuronais, aprendizado por indução, e outras. Elas empregam uma das seguintes estratégias básicas:

- memorização: o aprendizado é feito pela memorização de resultados obtidos na solução de problemas anteriores;

- instrução ou aconselhamento: o conhecimento é fornecido numa forma diretamente reconhecível pela máquina;

- dedução: o conhecimento já disponível é reformulado para gerar informações previamente ignoradas;

- analogia: novas informações são obtidas a partir da transferência de características de um conceito para outro;

- algoritmos genéticos: emprega uma abordagem baseada em mutações aleatórias de conceitos seguida por posterior seleção por critérios apropriados (sobrevivência do mais adequado);

- indução: é a abordagem mais estudada para aprendizagem simbólica. Utiliza-se de um grande conjunto de exemplos e contra-exemplos particulares para criar regras gerais.

A aprendizagem de roteamento [RICH 83] é a mais elementar das formas de aprendizado. Ela consiste meramente em armazenar resultados obtidos a partir da avaliação de uma árvore de decisão para uso em casos posteriores. Por exemplo, num jogo de xadrez a determinação do próximo movimento pode ser feita por meio de uma função heurística que associa um valor a cada possível posição subseqüente. A aprendizagem por roteamento consiste em armazenar esses valores de tal sorte que num próximo jogo eles não precisem ser recalculados.

Outra forma de aprendizado é o ajuste de parâmetros. No caso previamente citado do jogo de xadrez existe uma função heurística que combina uma série de fatores associados ao posicionamento das peças no tabuleiro de maneira a obter um valor para tal posicionamento. Esses fatores são combinados por meio dos parâmetros de uma certa função heurística. É muito difícil, porém, saber a princípio qual o valor correto desses parâmetros. Uma forma de lidar com essa dificuldade é começar apenas com uma estimativa e deixar o próprio programa modificar os valores dos parâmetros com base na experiência.

Isto é o que se denomina aprendizagem por ajuste de parâmetros [RICH 83].

Na aprendizagem pelo que foi dito o conhecimento é apresentado numa forma que a máquina é capaz de reconhecer. A máquina converte os conselhos fornecidos em um conjunto de afirmações que se relacionam diretamente ao que já é conhecido.

Na aprendizagem por analogias, o conhecimento fornecido à máquina não é diretamente relevante: o computador precisa levantar casos semelhantes que ajudem na solução do problema em questão.

As redes neuronais possuem uma abordagem coneccionista onde a aprendizagem é feita a partir da mudança dos padrões de conectividade entre as unidades processadoras, que são fortemente paralelas, por meio de processos baseados em teorias de otimização.

A técnica de indução é muito útil quando há exemplos documentados ou facilmente obteníveis. Essa técnica pressupõe poucas hipóteses a respeito da natureza dos dados sobre os quais opera, pode ser reaplicada, permite analisar casos extensos (muitos exemplos), evita falsas hipóteses e gera regras que o próprio especialista pode não conhecer.

Por outro lado, o processo de aprendizado por indução, normalmente fornece resultados sem explicações e não distingue informações necessárias daquelas apenas confirmatórias. A técnica assume que a base de exemplos é correta e completa e não há garantias que os resultados da análise sejam aplicáveis a dados diferentes da base de exemplos.

Há muitos algoritmos que implementam o aprendizado por indução. Um dos mais usados em programas comerciais é o "Iterative

Dichotomizer 3" - ID3 [HART 87]. Há mais de uma implementação do ID3, mas o princípio é construir uma árvore de decisão a partir de um conjunto de exemplos (training set) que permita discriminar classes de objetos do domínio. Os nós da árvore correspondem a atributos dos objetos tais como cor, tamanho, etc. Os arcos estão relacionados com os possíveis valores desses atributos. As folhas da árvore representam conjuntos de objetos com idêntica classificação. A determinação de quais atributos são apropriados para montar esta árvore de classificação é feita com base em características estatísticas do conjunto de exemplos usando, por exemplo, a entropia obtida pela teoria da informação ou por meio do cálculo do chi-quadrado.

Outro exemplo de algoritmo de indução é o AQ11 [MICHALSKI 83] que trabalha a partir de um processo de generalização-especialização de conceitos. O algoritmo começa com a hipótese mais geral possível e segue para descrições mais específicas. Quando uma hipótese falha em cobrir uma ocorrência mais específica ela é descartada por ser específica demais e então uma hipótese mais geral é formulada. Se uma hipótese cobre uma hipótese negativa ela é considerada geral demais e é eliminada, uma hipótese mais específica é então proposta. As hipóteses que não cobrem todos os exemplos são eliminadas por serem específicas demais. Esse tipo de indução é de natureza intrinsecamente heurística e é feita a partir de regras de generalização/especialização em oposição a estratégias estatísticas como a do ID3.

Ambos os algoritmos podem ser empregados para gerar diretamente regras de produção. As regras obtidas pelo AQ11 tendem, porém, a ser

mais simples e inteligíveis que as do ID3. O AQ11 exibe uma complexidade de cálculo exponencial com relação ao tamanho do problema enquanto o ID3 tem uma complexidade que cresce apenas linearmente, isto torna o ID3 mais rápido que o AQ11. O algoritmo do ID3 é conceitualmente mais simples, o que o torna mais fácil de ser compreendido e alterado, enquanto o AQ11 emprega regras ad hoc. Por outro lado, o AQ11 permite que o especialista forneça informações para guiar o processo de indução, o que não ocorre com o ID3 [RUBERG 89].

Várias são as aplicações possíveis para a aprendizagem de máquina. Podemos usá-la para a construção automática de bases de conhecimento, para o refinamento dessas bases, como auxílio à detecção de padrões, para a revelação de estruturas subjacentes a um conjunto de observações, e ainda em áreas como a visão e compreensão da fala, no ensino apoiado por computador, etc.

2.5. AQUISIÇÃO DE CONHECIMENTO BASEADA EM LINGUAGEM

A idéia básica é que o Engenheiro de Conhecimento formule um modelo conceitual que represente a estrutura do domínio, contendo seus objetos relevantes, com suas propriedades, e as relações existentes entre esses objetos. O uso de um tal modelo fornece apoio ao processo de Aquisição de Conhecimento por meio da formulação prévia de uma representação intermediária entre os dados obtidos do especialista e a sua codificação em máquina. Esse modelo se caracteriza por uma descrição em alto nível do conhecimento empregado

pelo especialista que é útil para estruturar o processo de Aquisição de Conhecimento. É uma forma ideal de documentar o processo de desenvolvimento do sistema. É particularmente adequado à construção de SEs de segunda geração que são aqueles que trabalham com conhecimento profundo. Conhecimento profundo (deep knowledge) é o conhecimento básico relacionado com os COMOs e os PORQUÊS de certo domínio, fornecendo as informações de causa e efeito necessárias. Já o conhecimento superficial (shallow knowledge) é o conhecimento baseado na experiência adquirida pelo especialista. Ele fornece atalhos através do problema, permitindo ao especialista solucioná-lo mais rápida, eficiente e precisa que um novato [FINK 85] [FINK 87].

Tais modelos são construídos por meio de linguagens apropriadas que fornecem um vocabulário no qual o conhecimento especializado pode ser representado de uma forma coerente. O uso dessas linguagens facilita a identificação de objetos e processos do domínio.

A função principal do modelo é atuar como uma ponte entre a aquisição e a representação do conhecimento, influenciando ambos os processos, tornando-os mais sistemáticos e eficientes. Pode-se, por essa abordagem, inferir de antemão os "tipos" de conhecimento que devem ser adquiridos, ter uma visão do problema e de sua resolução, evitando os transtornos da excessiva dependência do especialista, que tipicamente tem dificuldade em verbalizar o próprio conhecimento. A Aquisição de Conhecimento corresponderá, então, à obtenção dos elementos que instanciam o modelo com o emprego de técnicas de extração baseadas na psicologia (item 2.3).

Existem vários trabalhos feitos no sentido de definir linguagens de modelagem adequadas. Algumas dessas tentativas são

muito próximas ao formalismo de implementação [BYLANDER 86], outras são significativamente independentes desse formalismo [JOHNSON 87] [KERAVNOV 86] [ALEXANDER 86] [FREILING 85] [BREUKER 87b].

Johnson, por exemplo, fornece uma linguagem para a análise de descrições verbais (protocols) utilizando como primitivas bolhas (associadas ao contexto do problema), setas (que ligam as bolhas e indicam o encaminhamento a ser seguido para resolver um problema), triângulos (associados às setas e que explicitam o conjunto de habilidades necessárias para ir de um contexto a outro), nuvens (associados aos objetivos) e caixas (que indicam os elementos que disparam os objetivos). Trata-se de uma linguagem com pouco comprometimento com a implementação, totalmente voltada para o processo de análise das descrições verbais e que não se preocupa com uma modelagem mais profunda do conhecimento envolvido.

Já a análise ontológica de Alexander é voltada principalmente para a descrição das estrutura dos elementos do domínio. Essa descrição é feita em três níveis: num primeiro nível (ontologia estática) temos os objetos elementares, suas propriedades e relações, num segundo nível (ontologia dinâmica) é definido o espaço do problema e as ações que transformam o problema de um estado para outro e no terceiro nível (ontologia epistêmica) são definidas as restrições e métodos que controlam o uso do conhecimento dos outros níveis. Não há porém como descrever operadores ou o comportamento dos objetos, só suas estruturas. Não se trata, portanto, de uma linguagem capaz de lidar com o meta-conhecimento (conhecimento sobre o conhecimento), sendo fundamentalmente voltada para o conhecimento declarativo do domínio.

Uma proposta mais ampla e genérica é a metodologia KADS [BREUKER 87b]. O KADS fornece não apenas uma linguagem descritiva para a construção de modelos conceituais que é a linguagem KADS de modelagem conceitual (KCML) como também toda uma metodologia para a concepção como um todo de um SBC, baseada numa separação entre o processo de análise do problema e o processo de projeto do sistema. Além disso o KADS propõe uma gama de modelos básicos (a partir da identificação de padrões repetitivos nas tarefas especializadas) que podem ser empregados conforme a necessidade específica na modelagem do problema.

Para tanto o KADS conceitua e formula os Modelos de Interpretação [BREUKER 87b], que modelam tarefas elementares, chamadas tarefas genéricas, como diagnose, classificação, projeto e planejamento, entre outras. A adequada combinação de Modelos de Interpretação, relacionados com o escopo do SE a ser construído, forma, nesse enfoque, a base para o processo de Aquisição de Conhecimento relativa a construção do SE em consideração. O KADS, deve-se ressaltar, não é apenas uma linguagem de Aquisição de Conhecimento, mas algo bem mais amplo: uma metodologia de construção de SBCs que, em particular, prescreve uma linguagem de auxílio à Aquisição de Conhecimento.

Além do KADS, outras linguagens também identificam esses padrões repetitivos representados nos Modelos de Interpretação. Como exemplos temos os trabalhos de Bylander & Chandrasekaran [BYLANDER 86], Bylander & Mittal [BYLANDER 86], Gruber & Cohen [GRUBER 87]. O Modelo de Interpretação do KADS muito se aproxima da idéia de tarefas genéricas do grupo de Chandrasekaran da Universidade do Estado de

Ohio. Porém, ao contrário do Modelo de Interpretação que é uma representação do conhecimento especializado em si e independente de como ele será usado para construir um SE, as tarefas genéricas de Chandrasekaran são combinações primitivas entre essa representação e a sua implementação através de um mecanismo de controle. A Aquisição de Conhecimento é feita em função dessas tarefas genéricas que já contém em si um mecanismo de implementação.

O grande problema dessa abordagem é que ela torna difícil entender como tais tarefas genéricas, que se pressupõe serem primitivas, podem ser agrupadas de maneira a obter sistemas reais. Principalmente se forem criadas linguagens diferentes, como a CSRL [BYLANDER 86], para descrever cada tarefa genérica. Tal problema decorre do fato de não se distinguir entre a análise do conhecimento e o projeto do sistema real.

2.6. AQUISIÇÃO DE CONHECIMENTO POR FERRAMENTAS

Várias ferramentas foram desenvolvidas para facilitar o processo de Aquisição de Conhecimento, empregando técnicas de extração e de aprendizagem. Estas ferramentas são capazes de conduzir entrevistas, auxiliar a análise de descrições verbais, aprender por analogia, por indução, etc.

As ferramentas de aquisição existentes propiciam dois tipos de assistência [MARCUS 85]:

. aumento de facilidades que permitam ao especialista comunicar sua perícia à máquina;

. organização do conhecimento comunicado de forma a permitir obter o conhecimento relevante para a situação em foco.

Elas distinguem-se com relação aos seus graus de generalidade, ao tipo de problema a que são aplicáveis, ao nível de abstração em que trabalham, etc. [BOOSE 89].

Há os editores inteligentes de conhecimento, que apenas auxiliam o Engenheiro de Conhecimento a construir grandes bases de conhecimento como, por exemplo, o CYC, o KPT (KADS Power Tool) e o KREME [BOOSE 89]. Esses editores não têm um modelo previamente estabelecido, o que faz com que sejam menos potentes e, em contrapartida, de uso mais geral. As ferramentas fornecem recursos como editores de descrição verbal (protocol editor), editores gráficos e browsers para conhecimento e estruturas de raciocínio, que são muito úteis para auxiliar a fase de análise. Esses editores podem ser usados em combinação com outras ferramentas como, por exemplo, ferramentas de implementação, ou mesmo ferramentas de modelagem ou máquinas de aprendizagem.

Numa outra abordagem, temos as ferramentas de aquisição de conhecimento específicas (TEIRESIAS, OPAL, KITTEN, AQUINAS, etc. [BOOSE 89]) que se destinam apenas a certas classes de problemas, tornando disponível em máquina, algumas técnicas de aquisição de conhecimento. Elas são voltadas para problemas de domínios específicos e empregam um método especializado com muito conhecimento do domínio, ou são voltadas para problemas genéricos usando, nesse caso, um método geral com pouco conhecimento do domínio.

Nas ferramentas de aquisição de conhecimento específicas o diálogo entre a ferramenta e o especialista é feito por meio de uma

linguagem semi-natural ou é parte de um ambiente de edição restrito. O diálogo é controlado pela ferramenta e é voltado para o refinamento da base de conhecimento [BREUKER 88]. A seguir apresentaremos, como ilustração, um trecho de um diálogo entre o SE ROGET e um especialista humano [BENNET 1985]:

ROGET: Quais são os observáveis de um teste de cultura bacteriológica?

ESPECIALISTA: identidade do organismo encontrado na cultura informada pelo laboratório.

ROGET: Qual é a lista de valores esperados para o valor da identidade do organismo encontrado na cultura informada pelo laboratório?

Por favor entre cada termo ou frase em uma linha separada e finalize com uma linha em branco ou um DONE.

ESPECIALISTA: E. COLI

ESPECIALISTA: PROTEUS-MIRABOLIS

ESPECIALISTA: PSEUCOMAS

ESPECIALISTA:

Esse diálogo ou, de maneira geral, a Aquisição de Conhecimento, é dirigida por um modelo implícito na arquitetura do shell e na sua interface com o usuário. Esse modelo permite também que os dados obtidos do especialista sejam diretamente convertidos para um formalismo de implementação. Na maioria das ferramentas de Aquisição de Conhecimento existentes esse modelo é implícito e fixo o que faz com que tais ferramentas tenham seu uso restrito aos domínios aonde esse modelo seja aplicável. Essas ferramentas funcionam como

interface de sistemas já existentes, auxiliando a manutenção dos mesmos como, por exemplo, o TEIRESIAS [DAVIS 82] para o MYCIN e o OPAL [MUSEN 87] para o ONCOCIN.

Algumas ferramentas, porém, foram desenvolvidas se propondo a superar tal limitação. Estas ferramentas, embora tenham surgido do estudo de técnicas de aquisição para certo problema, foram generalizadas e se tornaram aplicáveis a uma classe maior de problemas semelhantes (ROGET, MOLE, MORE, KNACK, SALT, etc. [BOOSE 89]). Uma tal ferramenta se caracteriza por fornecer meios para descrever não apenas o conhecimento declarativo de um domínio, como também o meta-conhecimento acerca de como esse conhecimento pode ser usado na solução dos problemas.

Vamos agora examinar algumas das principais tentativas de construir ferramentas de Aquisição de Conhecimento gerais.

O ROGET [BENNET 85], desenvolvido a partir do EMYCIN (shell do MYCIN), foi uma das primeiras tentativas feitas nesse sentido. Para tanto ROGET conta com uma base de "estruturas conceituais" que são descrições abstratas da estrutura de SEs, composta por tipos de dados, tipos de inferências, etc., comparáveis aos Modelos de Interpretação do KADS. A generalidade do ROGET foi no entanto restringida em função dos limites impostos pela estrutura do EMYCIN, que é seu shell.

ROGET auxilia o usuário a selecionar uma estrutura conceitual adequada de sua biblioteca por meio de perguntas acerca do problema em questão e através de descrições das estruturas disponíveis. Numa segunda fase essas estruturas são editadas para comportar a base de dados do SE que será construído. Além disso, ROGET fornece conselhos

práticos acerca, por exemplo, da viabilidade do SE que se deseja construir. A base de dados é então compilada por ROGET para a forma de um SE.

A versão atual do ROGET é capaz de trabalhar apenas com conceitos e regras simples em domínios muitos semelhantes ao do MYCIN. Ao contrário do KADS, ROGET se concentra apenas na estrutura dos objetos com que trabalha e ignora os métodos de inferência e as estratégias. ROGET, é um sistema de modelo único, apesar de todo esforço que foi feito no sentido de torná-lo de uso geral.

MOLE [ESHELMAN 88] é um sistema desenvolvido a partir do SE homônimo. É um sistema adequado para lidar com problemas que envolvam exclusivamente classificação heurística, ou seja, a discriminação entre um conjunto de hipóteses a partir de associações com um conjunto de sintomas. Como tal técnica pode ser aplicada a uma grande classe de problemas, MOLE possui, em princípio, uma aplicabilidade ampla.

A contrapartida a essa aplicabilidade geral, oriunda de uma técnica fraca de solução de problemas é, naturalmente, ter pequeno poder de representação de conhecimento, o que restringe os domínios adequados ao MOLE. O conhecimento com que MOLE lida é um conhecimento acerca de associações entre conceitos, não há lugar para algo mais complexo tal como um modelo causal. Em tal esquema não é possível representar estados ou objetos [BREUKER 87b].

Um sistema semelhante ao MOLE é o **TKAW** [**BOOSE** 89] que é também um sistema voltado para diagnose. Porém trata-se de um sistema mais específico, destinado à diagnose de equipamentos nas quais as falhas podem ser representadas na forma de hierarquias abstratas. Em vista

dessa especificidade o TKAW possuiu um maior poder representativo que o MOLE.

STUDENT [GALE 86] é um sistema voltado para a análise estatística de dados, que é uma técnica muito específica e poderosa, ao contrário da classificação heurística do MOLE e TKAW. O sistema se baseia em uma "estrutura conceitual do domínio" que é equivalente conceitualmente à idéia de Modelos de Interpretação do KADS. Porém não é proposto um meio sistemático para construir tal estrutura conceitual ou uma linguagem em que ela possa ser representada. Sugere-se que essa estrutura conceitual seja derivada de algum SE já existente para o domínio o que, porém, é certamente um método difícil e demorado.

KNACK [KLINKER 87] é uma ferramenta de Aquisição de Conhecimento capaz de construir SEs voltados para a avaliação e aperfeiçoamento de projetos de sistemas eletro-mecânicos. O sistema emprega explicita, porém informalmente, os conceitos de métodos de solução de problemas e de finalidade do conhecimento que são muito semelhantes aos conceitos de estrutura de tarefa e de meta-classe da metodologia KADS, e que poderiam com um pouco mais de formalismo, levar à linguagem de modelagem conceitual do KADS (KCML).

Num extremo oposto ao KNACK, temos o **BLIP** [MORIK 86]. Enquanto o KNACK fornece uma linguagem descritiva poderosa que, no entanto, é usada em um domínio especifico, o BLIP é uma ferramenta que não se destina a nenhum domínio em particular. Em compensação a linguagem empregada pelo BLIP para construir seu modelo é a lógica, através do cálculo de predicados. BLIP possui recursos para refinar o modelo

inicial fornecido através de um sistema de aprendizado voltado para o refinamento do conhecimento envolvido nesse modelo porém, ainda assim, a lógica é uma linguagem descritiva pouco adequada para modelar o conhecimento especializado.

KRITON [LINSTER 89] também é uma ferramenta que em muitos aspectos aproxima-se da proposta da metodologia KADS pois, ao contrário das ferramentas anteriores, é uma abordagem baseada numa linguagem descritiva. O sistema auxilia o processo de Aquisição de Conhecimento por meio de um processo de análise de protocolo e da conversão desses dados para uma base de dados por meio de uma representação intermediária. Essa representação é feita por meio de uma linguagem descritiva de duas camadas, uma que descreve o conhecimento declarativo do domínio e outra que descreve o meta-conhecimento. Duas diferenças do KRITON para nossa abordagem é que a construção do modelo por meio da linguagem é dificultada pelo fato do KRITON, ao contrário do KADS, não adotar o conceito de modelos genéricos e, além disso, não há uma distinção clara entre o modelo do conhecimento especializado e o modelo de como esse conhecimento será usado na construção do SE (no KADS, Modelo Conceitual e Modelo de Projeto).

2.7. CONCLUSÕES

As técnicas psicológicas e o aprendizado por máquina são as que servem de base a qualquer processo de extração de conhecimento, mesmo que baseada numa linguagem ou em uma ferramenta. É provável que o

aprendizado por máquina venha a ter uma importância crescente no futuro pois ele automatiza totalmente o processo de Aquisição de Conhecimento, superando dessa maneira o gargalo da Aquisição de Conhecimento. Nesse sentido abordagens como a coneccionista [OBERMEIER 89] são muito promissoras. Atualmente, porém, as limitações do aprendizado por máquina são grandes e as aplicações práticas são restritas basicamente à construção de protótipos, à pesquisa e a outras situações particulares.

As técnicas de Aquisição de Conhecimento baseadas em técnicas psicológicas são sem dúvida as mais importantes e utilizadas. Apesar das técnicas psicológicas diretas serem as mais usualmente empregadas, as indiretas também fornecem importantes informações acerca do conhecimento do especialista. Deve-se ter em mente que cada uma tem um escopo limitado de aplicação. Assim devemos analisar a natureza do conhecimento a ser adquirido e a forma como ele está estruturado para termos condições de saber quais técnicas são mais adequadas.

Os especialistas utilizam diferentes representações de objetos e relações, assim como regras de inferência. Nem todas as técnicas psicológicas de Aquisição de Conhecimento que vimos são capazes de revelar todos esses tipos de Conhecimento. Na tabela 2.1 temos uma classificação dessas técnicas de acordo com sua capacidade de tratar tais elementos. O desenho de curvas fechadas e o desenho de diagramas destinam-se a esclarecer a relação entre os objetos no espaço do problema. A análise por fluxo de inferência, como indica o nome, fornece as cadeias de inferências empregadas pelo especialista. As demais técnicas diretas de extração de conhecimento, com exceção da

análise do fluxo de inferências, por possuírem uma forma livre, permitem que todo tipo de informação tenha chance de ser obtida. Elas não se destinam a um tipo especial de conhecimento.

	OBJETOS	*RELAÇÕES*	*REGRAS*
CURVAS FECHADAS		X	
DIAGRAMAS		X	
FLUXO *DE INFERÊNCIAS*			X
OUTROS MÉTODOS DIRETOS	X	X	X
MATRIZ DE REPERTÓRIO	X	X	X
OUTROS MÉTODOS INDIRETOS		X	

TABELA 2.1 - COMPARAÇÃO DOS MÉTODOS DE AQUISIÇÃO QUANTO AO TIPO DE CONHECIMENTO ELICITÁVEL.

Todas as técnicas diretas têm como deficiência o fato de suporem que o especialista é capaz de dizer ou mostrar aquilo que sabe. Já as técnicas indiretas, embora não dependam disso, são mais limitadas pois, com exceção da análise por matriz de repertório, fornecem apenas relações entre os objetos. Quando se emprega técnicas indiretas deve-se ter em mente que elas fazem hipóteses acerca da forma como o especialista organiza seu conhecimento, como vemos na tabela 2.2.

	LISTAS	TABELA	HIERARQUIA	FLUXOS	REDES	ESPAÇO FÍSICO	MODELO FÍSICO
Arvore Orden.	+		X				
Criação de gr			X				
Separ.de gr.			X				
Esc. multi-d.	+	+				X	
Agrup. hierar			X				
Redes com pes	+	+	+		X		+
Matriz de rep			X	+			
Curvas fechad					X	+	
Diagramas					X	+	
Fluxo de inf.				X	+		

A estrutura básica tratada é indicada com "X", e as formas que também podem naturalmente ser reveladas são marcadas com "+".

TABELA 2.2 - COMPARAÇÃO DOS MÉTODOS INDIRETOS QUANTO ÀS ESTRUTURAS ELICITÁVEIS.

Nenhuma das técnicas apresentadas, porém, resolve totalmente o complexo problema de Aquisição de Conhecimento. Mas, por meio de suas características, elas auxiliam em diversas etapas, facilitando o processo global de aquisição. Não existe, tampouco, nenhuma forma segura de saber quais delas aplicar. O Engenheiro de Conhecimento deve saber avaliar a adequação de cada uma para o problema que esteja tratando. Para tanto o processo de Aquisição de Conhecimento deve começar com um estudo da área em questão, seguida por uma série de entrevistas informais gravadas, transcritas e analisadas. A partir daí deverá ser avaliada a conveniência de empregar outras técnicas o que, na pior das hipóteses, ajudarão a melhorar o entendimento do problema.

O uso dessas diversas técnicas deve ser feita combinadamente pois adquirem diferentes tipos de conhecimento, conforme apresentado. Além disto, os resultados de uma técnica podem ser utilizados como auxílio à aplicação de outras técnicas. A forma como combiná-las, porém, permanece uma questão de intuição pessoal.

Já as técnicas de Aquisição de Conhecimento baseadas em linguagem e por ferramentas são um campo novo e que vem ganhando destaque crescente. O uso de uma linguagem pré-definida permite que se estruture de uma forma mais organizada o trabalho do Engenheiro do Conhecimento não só no que tange à questão da representação do conhecimento como também do processo de Aquisição do conhecimento e também é útil na escolha das técnicas de aquisição de conhecimento mais apropriadas. O uso de modelos é muito útil na construção de SEs de segunda geração, aonde é necessário um conhecimento mais profundo acerca do domínio.

As ferramentas de Aquisição de Conhecimento são uma outra tendência que já conta com um número enorme de produtos com finalidade de pesquisa e, também, com muitos produtos comerciais. Existem tanto as ferramentas de uso geral, os editores inteligentes de conhecimento, que são indicadas para auxiliar o trabalho de um Engenheiro de Conhecimento, como também ferramentas específicas para determinadas aplicações, que se destinam a especialistas que desejem construir seus sistemas sem o auxílio de um Engenheiro de Conhecimento.

Para casos práticos, a escolha da ferramenta ideal depende da tarefa considerada. Para problemas de configuração foi desenvolvido, por exemplo, o sistema SALT, para problemas de classificação os sistemas MOLE e AQUINAS, etc. Entre os sistemas de uso geral, que pressupõem no entanto como usuário um Engenheiro de Conhecimento, temos o KPT e o KRITON. O primeiro auxilia o Engenheiro de Conhecimento por meio de um sofisticado editor e de modelos

epistemológicos, enquanto o segundo emprega entrevista automática e análise de texto e descrição verbal.

Diversos pesquisadores têm trabalhado no sentido de atingir uma metodologia geral para o desenvolvimento de SEs [VAUDET 90] [ALBERT 90] [BREUKER 87b] [MOTTA 89]. Essas pesquisas têm apresentado, como subprodutos, uma gama de técnicas e ferramentas que devem ser conhecidas pelos Engenheiros de Conhecimento.

Uma abordagem muito interessante é a metodologia KADS que não só concebe uma metodologia geral voltada exclusivamente para o desenvolvimento de SBCs, como representa uma síntese entre a abordagem por linguagem e o uso de ferramentas de Aquisição de Conhecimento voltadas para domínios específicos. A biblioteca de Modelos de Interpretação, que é uma característica encontrada apenas no KADS, fornece modelos que servem de guia ao processo de Aquisição de Conhecimento e assim une as vantagens de se empregar uma ferramenta de Aquisição de Conhecimento com um modelo implícito que direciona o processo de extração do conhecimento com a abordagem mais geral baseada em linguagem.

CAPÍTULO III

A METODOLOGIA KADS

3.1. INTRODUÇÃO

O projeto KADS faz parte do programa de pesquisa da comunidade européia ESPRIT (European Strategic Programme for Research and Development in Information Technology). O coordenador do grupo de pesquisa é Bob J. Wielinga do Departamento de Ciência Social de Informática da Universidade de Amsterdam. O objetivo do projeto é criar uma metodologia inteligível e comercialmente viável que permita desenvolver SBCs de uma maneira mais estruturada do que atualmente é feito.

O termo KADS tem tido vários significados durante os seis anos de vida do projeto. Por exemplo: "Knowledge Acquisition Development System", "Knowledge Acquisition anD Structuring" e "Knowledge Acquisition, Design and Structuring".

Atualmente o nome completo do projeto ESPRIT KADS é "uma metodologia para o desenvolvimento de Sistemas Baseados em Conhecimento". Tal projeto se iniciou em setembro de 1985 e terminou oficialmente em fevereiro de 1990. Pesquisas ainda estão em andamento e está prevista uma continuação do projeto que se denominará "KADS II" que será orientada ao refinamento da metodologia KADS. Dois pequenos projetos pilotos preliminares precederam o atual projeto com duração total de 1 ano e meio. No total estima-se que o projeto KADS como um todo consumiu 120 homens/ano.

3.2. NOÇÕES GERAIS

O projeto KADS foi concebido em 1984 numa época em que não havia grande interesse metodológico dentro da comunidade de Inteligência Artificial. O paradigma básico para construir SBCs era a prototipagem rápida usando hardware e software específicos, tais como máquinas LISP, shells, etc. Desde então muitas organizações perceberam que o desenvolvimento de SBCs de um ponto de vista organizacional não difere fundamentalmente de outros sistemas destinados a lidar com informação. Certas características encontradas no desenvolvimento de SBCs tais como a análise de informação, seleção de aplicações, gerenciamento do projeto, documentação de requisitos do projeto, projeto modular, reutilização, etc., são similares aos encontrados no desenvolvimento de sistemas convencionais.

3.2.1. Metodologia

Há pouco consenso acerca do que é uma metodologia. Há um uso indistinto de termos como método, abordagem, técnica que estão de alguma maneira relacionados com o conceito de metodologia, mas que são usados diferentemente por vários autores. Dentro da cultura científica européia, da qual o KADS se originou, metodologia significa a ciência dos métodos, enquanto método significa um conjunto de regras sobre como desenvolver um sistema. O KADS adota o termo metodologia pois considera que sua abordagem abrange mais do que uma mera coleção de métodos, pois também apresenta uma teoria que embasa estes métodos.

Os paradigmas seguidos pela metodologia KADS são [WIELLINGA 89]:

a) Uso de uma linguagem intermediária para captar a essência dos modelos tanto na fase de análise quanto na fase de projeto.

b) Emprego da noção de modelos genéricos de domínios, tarefas e métodos de computação genéricos para solução de problemas.

c) A separação entre diferentes tipos de conhecimento de acordo com o emprego dado a este conhecimento.

Os princípios que o KADS assume são [BREUKER 87b]:

a) O conhecimento e o domínio devem ser analisados antes que o projeto e a implementação do SBC comece. Grande esforço deve ser concentrado na Aquisição de Conhecimento antes que um formalismo de implementação seja escolhido.

b) Tal análise deve ser feita de uma maneira dirigida por um modelo tão cedo quanto possível. Um modelo da estratégia de solução do problema em questão não apenas facilita a análise dos dados como também ajuda a identificação de um formalismo de implementação adequado.

c) A transição dos dados verbais do domínio para o formalismo rígido da implementação deve ser feita passando por um modelo expresso num nível epistemológico.

d) A análise deve incluir não apenas o conhecimento do especialista como também a funcionalidade do sistema desejado. Dados acerca do ambiente e do usuário devem ser obtidos e analisados.

e) Novos dados só devem ser obtidos do especialista quando a análise de dados previamente obtidos mostre a necessidade de

esclarecimentos e complementações. Conseqüentemente extração e análise irão se alternar.

f) Os dados obtidos e as interpretações devem ser documentados. Documentação contínua é uma condição necessária para o trabalho em equipe.

3.2.2. A abordagem KADS

A Engenharia de Conhecimento é um processo que se subdivide em três atividades: extração, interpretação e organização do conhecimento especializado. Vários autores se referem à Aquisição de Conhecimento como sendo o gargalo da construção de SEs. Do ponto de vista do KADS o maior empecilho dessa tarefa está não na obtenção de conhecimento, ou seja na extração, mas sim na conversão deste conhecimento para formas úteis, ou seja está na modelagem do conhecimento.

Para solucionar esse problema o KADS concebe duas fases distintas na construção de um SBC: uma fase de abstração, onde se obtém uma descrição de alto nível do conhecimento relevante, e uma fase de projeto, onde essa descrição é formalizada na forma de um SBC.

Na metodologia KADS um SBC é visto como um modelo de um sistema do mundo real e não como um recipiente de conhecimento. Assim a construção de um tal sistema deve ser feita por um processo gradual de modelagem, a partir do mapeamento de uma compreensão de certo comportamento exibido por um sistema externo para a sua descrição na forma de um artefato.

Esse mapeamento deve ser feito por etapas. A partir de uma descrição verbal de um domínio obtida de um especialista, há uma abstração desse conhecimento dando origem a um modelo esquemático, com certa tendência cognitiva, que é chamado Modelo Conceitual. A descrição abstrata do conhecimento contida nesse modelo deve então ser melhor definida, levando em conta as especificações, com o intuito de criar um modelo que sirva de base para o projeto do SBC. Tal modelo, que tem o mesmo nível de abstração do Modelo Conceitual, é chamado Modelo de Projeto. Esse modelo é concebido por um processo de formalização que visa obter uma sintaxe explícita e não ambígua para os termos usados, garantindo uma consistência e transparência sem a qual o processo de construção de um SBC é inviável. Pode-se atingir, assim, uma descrição detalhada do SBC, passível de implementação, que é o Modelo de Projeto Detalhado. O SBC é obtido por um mapeamento direto desse modelo para uma linguagem de programação.

3.3. O PROCESSO DE INTERPRETAÇÃO

No KADS a construção de um SBC passa por diferentes formas de representação do conhecimento especialista, o que é feito gradualmente a partir das seguintes etapas: identificação do conhecimento, conceituação, análise epistemológica, análise lógica e implementação. Essas etapas constituem o processo de interpretação do KADS. Cada uma dessas fases apresenta o conhecimento do domínio em um diferente nível de abstração e/ou formalização.

Esses 5 níveis representam uma síntese entre as 4 quatro formas de analisar um conhecimento de Sloman [SLOMAN 80] (nível individual, conceitual, ao nível de um particular formalismo e ao nível de mecanismos que implementam um formalismo) e os níveis de representação de primitivas de tipos de conhecimento de Branchman (1978), que são: nível lingüístico, conceitual, epistemológico, lógico e de implementação.

3.3.1. Identificação do conhecimento

Nesse nível o conhecimento do domínio está representado em linguagem natural obtida de um especialista ou de livros e manuais. Na fase de identificação o conhecimento se apresenta numa forma verbal, como vemos na figura 3.1.

"MIKE: Por que, quero dizer, a principal coisa que quero fazer é olhar para, bem, fazer 2 coisas: uma é olhar as células que estão mudando quando os animais estão aprendendo algo de novo, fazer perguntas sobre como os padrões de conecções dendríticas variam quando o animal aprende, para ver como a dimensão espacial dos processos dendríticos estão mudando, ver como sua orientação, seu campo ..."

FIGURA 3.1 - Trecho de uma descrição verbal [MOTTA 89].

Nesse ponto, um mesmo conhecimento pode ser representado de maneiras diferentes pelos especialistas, seja por que eles empregam diferentes terminologias seja por que eles não estruturam seu

UNICAMP
BIBLIOTECA CENTRAL

conhecimento da mesma maneira. A fase de identificação visa identificar certos conceitos do domínio com o intuito de organizar a vasta massa de dados obtida do especialista, preparando assim uma fase posterior de interpretação do conhecimento.

Durante a fase de identificação as transcrições verbais são organizadas de maneira a localizar os conceitos relevantes do domínio. Para tanto o conhecimento extraído do especialista passa por um processo de remoção de informações irrelevantes (ruído), e de localização e extração de grupos atômicos de conhecimento representados por conceitos, afirmações, definições e relações. A análise dos dados depende fortemente das técnicas de Aquisição de Conhecimento empregadas: quando se usam técnicas estruturadas de Aquisição de Conhecimento esta é uma tarefa muito mais simples do que quando se aplicam técnicas não estruturadas [MOTTA 89].

A fase de identificação pode ser facilitada pelo emprego de ferramentas automáticas, tais como browsers e editores gráficos. No KNACK [KLINKER 87] as descrições são analisadas pela identificação e isolamento de palavras chaves e conceitos no texto, no KRITON [LINSTER 89] o texto passa por uma análise estatística. Uma forma de ver a fase de identificação é como sendo um processo no qual se busca desenvolver uma enciclopédia conceitual caracterizada pela associação de conceitos com fragmentos do texto sob análise. Um exemplo do resultado desse processo de análise é apresentado na figura 3.2.

OBJETOS
    AXÔNIOS
    CÉLULAS
        TIPOS DE CÉLULAS
        ATIVIDADE DA CÉLULA
    DENDRITOS
    TRANSMISSOR
    SINAPSE
MUDANÇA
    ANÁLISE ESTATÍSTICA
MUDANÇAS NA MEMBRANA
    MUITO RÁPIDO

FIGURA 3.2 - Exemplo do resultado do processo de análise [MOTTA 89].

3.3.2. Conceituação do conhecimento

Durante a fase de conceituação o Engenheiro de Conhecimento tenta impor uma estrutura global sobre os dados coletados. Os conceitos identificados na fase anterior servem de base para a fase de conceituação que visa integrar tais conceitos de acordo com primitivas conceituais ou tipos. Os tipos são classes de elementos estruturados. A primitiva conceitual relação, por exemplo, pode ser de tipos como parte_de, é_um, causa, etc. Assim podemos integrar os conceitos "casa" e "telhado", através do tipo de relação "parte_de", da seguinte forma: "telhado é parte da casa". Através do uso destes tipos é possível organizar coerentemente os conceitos, embora os princípios de estruturação dessa organização ainda não estejam explícitos.

Na fase de análise obtém-se o que é chamado MODELO DE INTERPRETAÇÃO que é uma descrição esquemática do processo de resolução do problema empregado pelo especialista e que guia o processo de análise. O conceito de Modelo de Interpretação é

fundamental dentro da metodologia KADS e será melhor apresentado posteriormente, no item 3.4.2.

Obtém-se a partir da análise conceitual uma representação abstrata do domínio na forma de tabelas, diagramas de fluxo, taxonomias hierárquicas, redes causais, ou outra forma de organização que se julgar conveniente. No caso da análise de transcrições verbais no domínio da diagnose, por exemplo, deve-se isolar conceitos de sintomas, reclamações, resultados de testes, etc. Os dados verbais obtidos devem então ser mapeados no Modelo de Interpretação de maneira a organizar as informações fornecidas pelo especialista.

3.3.3. **Análise epistemológica.**

Durante a análise epistemológica determinam-se propriedades estruturais do conhecimento especializado objetivando organizar o conhecimento do domínio a partir de uma formalização epistemológica. Cada elemento pode possuir uma estrutura própria levando assim a sub níveis estruturais. Essa organização é feita por meio das primitivas da linguagem de modelagem conceitual KADS (KCML), que posteriormente analisaremos. Essa fase finda por conceber um MODELO CONCEITUAL do conhecimento especializado que é uma descrição em alto nível, sem vínculos com a implementação, do conhecimento do domínio e que serve de base para dar início à fase de projeto.

3.3.4. **Análise lógica.**

Durante a análise lógica as estruturas obtidas na fase anterior são mapeadas para um formalismo que representa o conhecimento e as inferências feitas. A análise lógica parte do Modelo Conceitual

obtido na fase anterior e dos requerimentos desejados e se obtém uma descrição parcial do SBC (a análise lógica faz parte do processo geral de projeto do sistema).

3.3.5. **Análise de implementação**

Na análise de implementação o modelo é convertido para mecanismos que sejam convenientes para descrever a implementação do SBC. As primitivas empregadas para tal fim são, por exemplo, preenchimento de slots, matching, testes, etc.

3.3.6. **O ciclo de vida**

A construção de um SBC é um processo de sucessivas modelagens que visa ir de dados sobre um sistema até a implementação de um artefato que reproduza tal comportamento. Essa modelagem é feita através de uma fase inicial de análise do problema, que gera um modelo altamente abstrato, seguida de uma fase de projeto que produz o SBC. Durante esse processo são obtidos três modelos intermediários: o Modelo Conceitual ou M1, o Modelo de Projeto ou M2 e o Modelo de Projeto Detalhado ou M3, conforme vemos na figura 3.3.

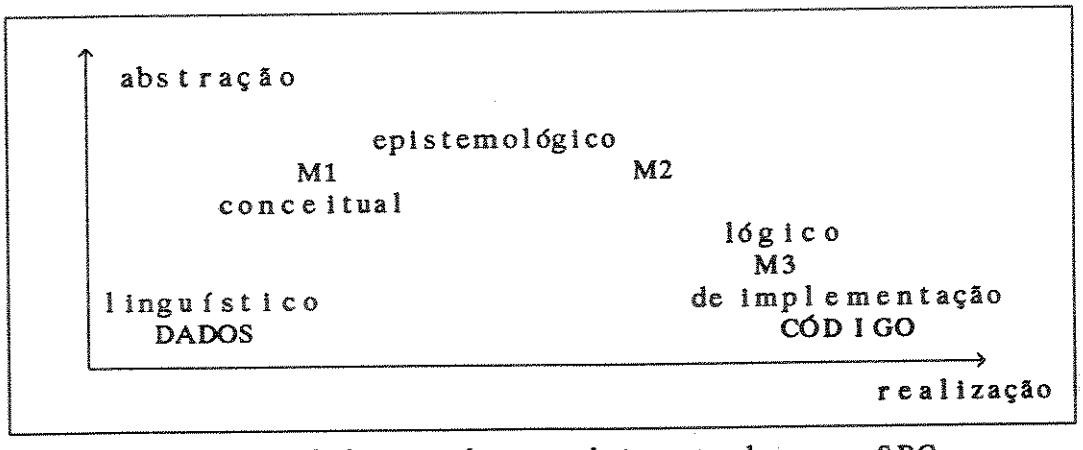


FIGURA 3.3 - Modelos no desenvolvimento de um SBC.

DADOS refere-se às informações obtidos pelo Engenheiro de Conhecimento a partir de um especialista ou de livros e manuais. Os dados possuem um nível de abstração muito baixo. Num nível superior de abstração temos o modelo M1 que é o Modelo Conceitual. Trata-se de uma descrição abstrata do conhecimento do especialista, independente de qualquer formalismo de implementação. O M1, conjuntamente com os requisitos desejados para o SBC, dá origem ao Modelo de Projeto (M2) que é um modelo num mesmo nível de abstração que o Modelo Conceitual, porém já voltado para uma possível implementação. Entre o Modelo de Projeto e o artefato final há o Modelo de Projeto Detalhado ou Técnico (M3), que pode ser convertido num SBC por meio de um mapeamento direto.

3.4. MODELOS NO KADS

A metodologia KADS conceitua uma série de modelos envolvidos na concepção de um SBC. Além dos já mencionados MODELO CONCEITUAL, MODELO DE PROJETO e MODELO DE PROJETO DETALHADO, temos também o MODELO DE INTERPRETAÇÃO e o MODELO DE COOPERAÇÃO.

3.4.1. O Modelo Conceitual

O **MODELO CONCEITUAL** é uma descrição de alto nível do conhecimento especializado requerido para realizar certa atividade, é

DADOS refere-se às informações obtidos pelo Engenheiro de Conhecimento a partir de um especialista ou de livros e manuais. Os dados possuem um nível de abstração muito baixo. Num nível superior de abstração temos o modelo M1 que é o Modelo Conceitual. Trata-se de uma descrição abstrata do conhecimento do especialista, independente de qualquer formalismo de implementação. O M1, conjuntamente com os requisitos desejados para o SBC, dá origem ao Modelo de Projeto (M2) que é um modelo num mesmo nível de abstração que o Modelo Conceitual, porém já voltado para uma possível implementação. Entre o Modelo de Projeto e o artefato final há o Modelo de Projeto Detalhado ou Técnico (M3), que pode ser convertido num SBC por meio de um mapeamento direto.

3.4. MODELOS NO KADS

A metodologia KADS conceitua uma série de modelos envolvidos na concepção de um SBC. Além dos já mencionados MODELO CONCEITUAL, MODELO DE PROJETO e MODELO DE PROJETO DETALHADO, temos também o MODELO DE INTERPRETAÇãO e o MODELO DE COOPERAÇãO.

3.4.1. O Modelo Conceitual

O **MODELO CONCEITUAL** é uma descrição de alto nível do conhecimento especializado requerido para realizar certa atividade, é

um modelo completo de certa atividade que futuramente será implementada em um SBC, porém sem qualquer vinculação com como isto será feito.

A descrição do Modelo Conceitual é feita a partir de uma linguagem denominada KCML (KADS Conceptual Modeling Language). A base dessa linguagem é a divisão do conhecimento especializado em quatro tipos ou camadas: conhecimento estático ou do domínio, conhecimento relativo a inferências, conhecimento a nível da tarefa e conhecimento estratégico. O conhecimento do domínio refere-se à descrição dos conceitos, relações e estruturas necessárias à caracterização do domínio. O conhecimento das inferências descreve as inferências que podem ser feitas sobre os conhecimentos do domínio, por exemplo, generalizações, abstrações, classificações, etc. O conhecimento de tarefa descreve a forma como as inferências são organizadas e controladas para que um certo objetivo seja atingido. O conhecimento de estratégias refere-se à forma de se utilizar as tarefas dentro de um planejamento amplo, possivelmente com uma supervisão e controle da execução dessas tarefas, detectar e resolver impasses, para que o problema em questão como um todo possa ser resolvido.

O Modelo Conceitual possui uma descrição completa desses quatro tipos de conhecimento especializado. O nível de representação desse modelo é abstrato porém pode vir a ser executável. No último caso a fase de projeto consistirá em transformar a especificação executável do Modelo Conceitual em um sistema eficiente e utilizável. O Modelo Conceitual serve de base para o projeto do SBC.

3.4.2. O Modelo de Interpretação

O **MODELO DE INTERPRETAÇÃO** é uma descrição do conhecimento necessário para realizar uma certa classe de tarefas ao nível dos conhecimentos referentes às inferências realizadas, das tarefas executadas e das estratégias empregadas. Um Modelo de Interpretação pode ser visto como um Modelo Conceitual abstraído de suas características específicas a um domínio e, portanto, mantém alguma semelhança com o conceito de shell que é um SE abstraído de suas características do domínio. O Modelo de Interpretação, assim como o Modelo Conceitual, é descrito com o auxílio da KCML.

O Modelo de Interpretação é construído com o auxílio de uma biblioteca fornecida pelo KADS, que fornece Modelos de Interpretação de tarefas básicas (tarefas genéricas), de tal sorte que Modelos de Interpretação de tarefas da vida real possam ser construídos a partir da composição de Modelos de Interpretação dessas tarefas elementares.

3.4.3. O Modelo de Cooperação

O **MODELO DE COOPERAÇÃO** é uma representação da forma como o SBC deve interagir com o usuário. Os usuários e o sistema podem ser vistos como cooperando para realizar alguma tarefa mais ampla do que a que o SBC realiza. A análise da maneira como o SBC e o usuário dividem as sub-tarefas que devem ser realizadas e como eles trocam informações é descrita no Modelo de Cooperação. Esse modelo especifica as diferentes modalidades (maneiras) nas quais o usuário e o sistema juntos podem realizar a tarefa. Como exemplos de modalidades temos o sistema solucionador autônomo de problemas, o professor, o instrutor e o conselheiro. A modalidade determina em

grande medida como o SBC deve se comunicar com o usuário e que partes do seu processamento devem ser transparentes para ele.

O Modelo de Cooperação é composto por:

a) O modelo da tarefa. É a decomposição hierárquica da tarefa em sub-tarefas.

b) A distribuição das sub tarefas para agentes que, por exemplo, pode se resumir a determinar quais tarefas serão executadas pelo SBC e quais o serão pelo usuário.

c) Os recursos fornecidos por cada usuário para cada tarefa (objetos, habilidades e conhecimentos).

d) As tarefas de comunicação implícitas à distribuição de tarefas e a propriedade dos recursos.

e) As tarefas de comunicação necessárias para recursos adicionais.

f) A iniciativa pelas tarefas de comunicação.

O Modelo de Cooperação é um aspecto recente e ainda não totalmente desenvolvido e testado da metodologia KADS. Em geral a análise da cooperação pode ser ignorada adotando-se uma forma fixa de interação entre o sistema e o usuário, ou seja uma modalidade fixa. Nos casos mais simples a comunicação se resume a obter informações do usuário e apresentar uma solução.

3.4.4. O Modelo de Projeto

O **MODELO DE PROJETO** é um Modelo Conceitual modificado de maneira a levar em conta o desempenho que se deseja no SBC. Enquanto o Modelo Conceitual é uma descrição apenas do conhecimento

especializado, o Modelo de Projeto leva em conta o que se deseja ao construir o SBC.

O Modelo de Projeto se compõe de três elementos ou camadas:

a) Camada funcional. Consiste na descrição funcional do sistema, ou seja, do comportamento esperado do SBC. O modelo funcional é obtido pela conversão do Modelo Conceitual em um conjunto de blocos funcionais através de uma integração do Modelo Conceitual com os requerimentos do sistema.

b) Camada dos métodos ou do comportamento. Consiste na descrição dos métodos que serão empregados para realizar as funções que o SBC executará. Tais métodos são os usualmente usados dentro da Inteligência Artificial: algoritmos de busca, classificações por meio de relações é-um, etc. O métodos são compostos de elementos de projeto que são as partes físicas destes métodos. Por exemplo, o método da classificação hierárquica é composto por procedimentos de classificação, subsumption relations, definições de classe e pares atributo-valor [WIELINGA 88].

c) Camada de estrutura. Os elementos de projeto dos métodos escolhidos são decompostos em módulos, fornecendo assim a base para o projeto detalhado e para a implementação. A estrutura dos módulos representa a arquitetura do SBC. Na especificação dessa camada deverá ser levado em conta o ambiente em que o SBC deverá ser implementado. Como exemplos de ambientes temos PROLOG, KEE, EMYCIN, cada um deles possui seu conjunto de métodos e elementos de projeto que pode ou não ser ampliado.

3.4.5. O Modelo de Projeto Detalhado

O **MODELO DE PROJETO DETALHADO** . Trata-se da descrição do SBC em termos de mecanismos que sejam convenientes para a implementação do SBC. As primitivas empregadas para tal fim são, por exemplo, preenchimento de slots, matching, testes, etc.

Na figura 3.4 apresentamos esquematicamente a seqüência em que esses modelos são obtidos durante o processo de concepção de um SBC.

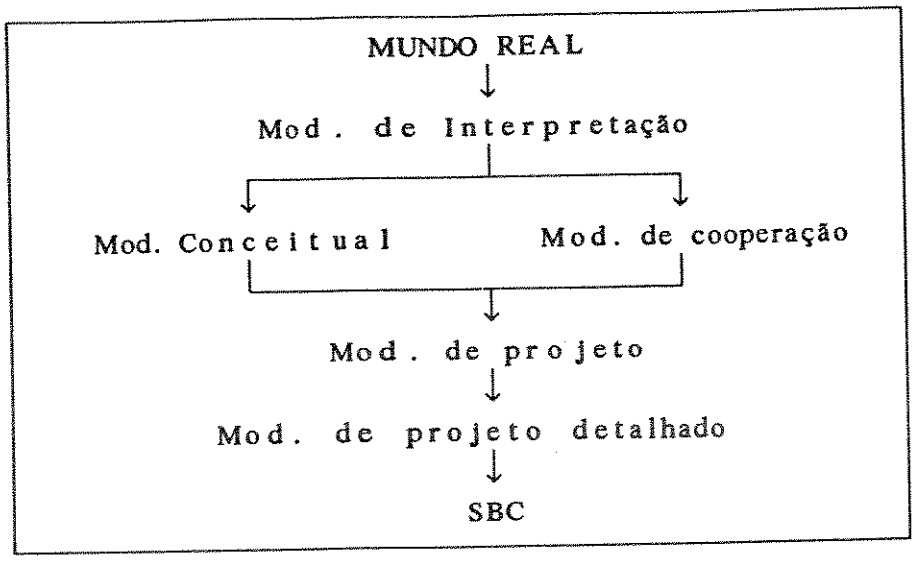


FIGURA 3.4 - Os modelos no KADS.

3.5. A LINGUAGEM KADS DE MODELAGEM CONCEITUAL

A linguagem KADS de modelagem conceitual (KCML) destina-se a fornecer recursos para representar o conhecimento especializado. Essa linguagem tem como base uma teoria acerca da forma como os especialistas resolvem seus problemas que é a teoria das quatro camadas [WIELINGA 86]. Essa teoria se baseia em duas hipóteses:

- é possível e útil distinguir entre diversos tipos de conhecimento;

- esses tipos de conhecimento podem ser organizados em camadas, que têm interação limitada.

A KCML está fortemente associada ao modelo de quatro camadas do KADS, que é a divisão do conhecimento especializado em conhecimento do domínio, de inferências, de tarefas e de estratégia. Para cada um desses tipos de conhecimento o KADS fornece termos específicos para descrever o conhecimento, figura 3.5.

CAMADA	OBJETOS	ORGANIZAÇÃO
ESTRATÉGIA	PLANOS, META-REGRAS REPAROS, IMPASSES	ESTRUTURA DO PROCESSO
TAREFA	OBJETIVOS, TAREFAS	ESTRUTURA DA TAREFA
INFERÊNCIA	METACLASSES FONTES DE CONHECIMENTO	ESTRUTURA DE INFERÊNCIA
DOMÍNIO	CONCEITOS, RELAÇÕES ESTRUTURAS, REGRAS	ESTRUTURA AXIOMÁTICA

FIGURA 3.5 - Camadas de descrição do conhecimento especializado

3.5.1. Camada de domínio

Na camada de domínio temos uma descrição declarativa do conhecimento especializado. A distinção dessa camada é um primeiro passo para se obter uma descrição flexível e reutilizável do conhecimento especializado. O conhecimento nessa camada está numa forma que é independente da tarefa que é realizada, ou seja, a

representação não depende do uso a que se destina. Assim, esse conhecimento do domínio pode ser usado tanto para fins de solução de problemas, como para apresentação de explicações, para ensino, etc.

O KCML não prescreve termos específicos para descrever o conhecimento contido nessa camada porque o KADS entende que o Engenheiro de Conhecimento deve estar livre para empregar qualquer formalismo ou representação de conhecimento que julgar adequada. Seja qual for o formalismo de representação empregado pelo Engenheiro de Conhecimento ele deverá representar os elementos constituintes do domínio que são conceitos, relações, estruturas e regras do domínio.

- Conceitos. Deve haver uma descrição estruturada dos conceitos relevantes do domínio com seus atributos. Por exemplo:

"Nome do conceito:
descrição:
origem:
transcrição:
sinônimo:
atributo-1 <restrição de valor>
...
atributo-n <restrição de valor>"

- Relações. Deve haver também uma descrição das relações relevantes ao domínio. Exemplos de relações são: É-UM, PARTE-DE, etc.

- Estruturas. As estruturas são entidades que resultam de agrupamentos de conceitos através de relações. Exemplos de estruturas são a estrutura física de um equipamento, o "lay-out" de um sistema, etc.

- Regras do domínio. São descrições em uma linguagem natural estruturada de regras "SE-ENTÃO" na qual se pode expressar relações do domínio. Essas relações podem ser de vários tipos: relações

causais, heurísticas, relações temporais, etc. Um exemplo de uma regra do domínio pode ser [WIELINGA 89]:

"Se
a tarefa principal é interromper
então
o rotor é uma turbina axial.

Se
a tarefa principal é dispersão
então
o rotor é uma turbina radial."

A presença de regras na camada do domínio deve ser vista com cautela, pois trata-se de um assunto em que há divergências dentro do projeto KADS [BREUKER 87b] [WIELINGA 89].

3.5.2. **A camada de inferência**

Nesse nível está descrita a competência necessária para realizar inferências sobre os elementos da camada do domínio, o que se consegue por meio de um meta-conhecimento do domínio. A camada de inferência especifica quais inferências podem ser feitas, **não como ou quando elas são feitas**. O como depende das estruturas do domínio que permitem que métodos sejam aplicados. O quando é especificado na próxima camada que será descrita mais à frente.

O KCML fornece dois elementos básicos para descrever o conhecimento desse nível: as Fontes de Conhecimento e as Metaclasses dos quais resulta uma estrutura descritiva que é a Estrutura de Inferência,

As **FONTES DE CONHECIMENTO** são descrições de tipos de inferências elementares, tais como abstração, especificação, construção, etc. Deve-se salientar que o termo "Fonte de Conhecimento" não tem o mesmo sentido das fontes de conhecimento das arquiteturas de blackboard, tal como no HEARSAY [LESSER 77].

A Fontes de Conhecimento são descrições funcionais que possuem parâmetros de entrada e de saída. Uma fonte de conhecimento apenas descreve o que acontece com seus parâmetros sem no entanto dizer como isto ocorre. Assim, por exemplo, a fonte de conhecimento "ABSTRAI" indica que seu parâmetro de saída é mais abstrato, ou seja possui menos atributos, que seu parâmetro de entrada.

Uma Fonte de Conhecimento é a descrição de uma manipulação sobre conceitos. As operações possíveis com conceitos são: transformação do conceito num novo conceito, geração de um novo conceito a partir de um conceito previamente definido e geração de um novo conceito a partir do relacionamento entre dois conceitos. Além dessas três operações, temos também as operações com estruturas. Na figura 3.6 apresentamos uma tipologia de fontes de conhecimento sob esse prisma.

MUDA CONCEITO
- ATRIBUI VALOR
- CALCULA

GERA CONCEITO
- INSTANCIA
- CLASSIFICA (IDENTIFICA)
- GENERALIZA
- ABSTRAI
- ESPECIFICA

DIFERENCIA ENTRE CONCEITOS
- COMPARA
- CASA

MANIPULAÇÃO DE ESTRUTURA
- MONTA
- DECOMPÕE
- TRANSFORMA

FIGURA 3.6 - Tipologia das Fontes de Conhecimento.

Uma Fonte de Conhecimento é descrita por meio de uma referência a seus parâmetros de entrada e saída, ao método empregado para realizar a inferência e ao conhecimento do domínio necessário para realizar a inferência. Por exemplo, apresentamos a seguir a descrição da fonte de conhecimento "ESPECIFICA" cuja função é determinar o valor normal de saída de uma função:

FONTE DE CONHECIMENTO: especifica. Esta fonte de conhecimento especifica a saída normal, ou seja, o valor correto esperado de saída ou uma função dado algum valor ou função de entrada, do componente aonde é assumido que esteja a falha.
ENTRADA: modelo do sistema.
SAÍDA: norma.
MÉTODO: associação direta.
CONHECIMENTO DO DOMÍNIO: conhecimento acerca do comportamento.

Mais recentemente a metodologia KADS foi alterada e as Fontes de Conhecimento passaram a ser descritas apenas pelo nome da inferência que realizam e, eventualmente, por uma estrutura de inferência que a detalhe.

As **METACLASSES** representam as funções que uma classe de conceitos do domínio pode ter durante certo processo de solução de problemas, como por exemplo, "hipótese", "evidência" e "conclusão". Uma Metaclasse indica como um conjunto de conceitos pode ser usado. Por exemplo, na diagnose médica, o conceito "gripe" pertence à Metaclasse "conclusão" enquanto o conceito "febre" pertence às Metaclasses "evidência" e "hipótese".

As Metaclasses são os argumentos das Fontes de Conhecimento, elas não possuem estruturas sendo meros slots que podem ser preenchidos com um conceito do domínio a que estejam associadas. Na figura 3.7 apresentamos sua tipologia.

```
PROBLEMA
      QUESTÃO
DADO
      ESTRUTURA_DE_DADOS
            DESCRIÇÃO_DO_CASO
            DESCRIÇÃO_DO_SISTEMA
      DADO_INDIVIDUAL
            RESTRIÇÃO
            VARIÁVEL
            SINTOMA
                  RECLAMAÇÃO
      PAPEL INTERMEDIÁRIO DO DADO/PROBLEMA
            PARÂMETRO
            FATOR
            DESCOBERTA
                  EVIDÊNCIA
      PAPEL INTERMEDIÁRIO DO CONHECIMENTO DO DOMÍNIO
            MODELO DO SISTEMA
            HIPÓTESE
            NORMA
            TERMO
      SOLUÇÃO
            DIAGNOSE
            CLASSE_DE_DECISÃO
            PLANO
            PROJETO
```

FIGURA 3.7 - Tipologia de Metaclasses.

A ESTRUTURA DE INFERÊNCIA é uma representação gráfica do fluxo de informações do processo completo de inferência através das Fontes de Conhecimento e das Metaclasses. Na figura 3.8 temos, como exemplo, a Estrutura de Inferência da diagnose sistemática. A interpretação que deve ser dada ao diagrama é a seguinte. A partir de uma reclamação do usuário referente a um defeito em certo equipamento, seleciona-se a representação adequada para o equipamento (o modelo do sistema que será usado na diagnose). Esse modelo é decomposto em sub partes, passando cada sub parte a representar uma hipótese de onde está a falha. Cada hipótese é testada. Caso o teste determine que a falha está em uma certa sub parte, esta é novamente decomposta. Se, porém, não for possível decompor tal sub parte então obteve-se uma conclusão, ou seja, a causa da falha foi encontrada.

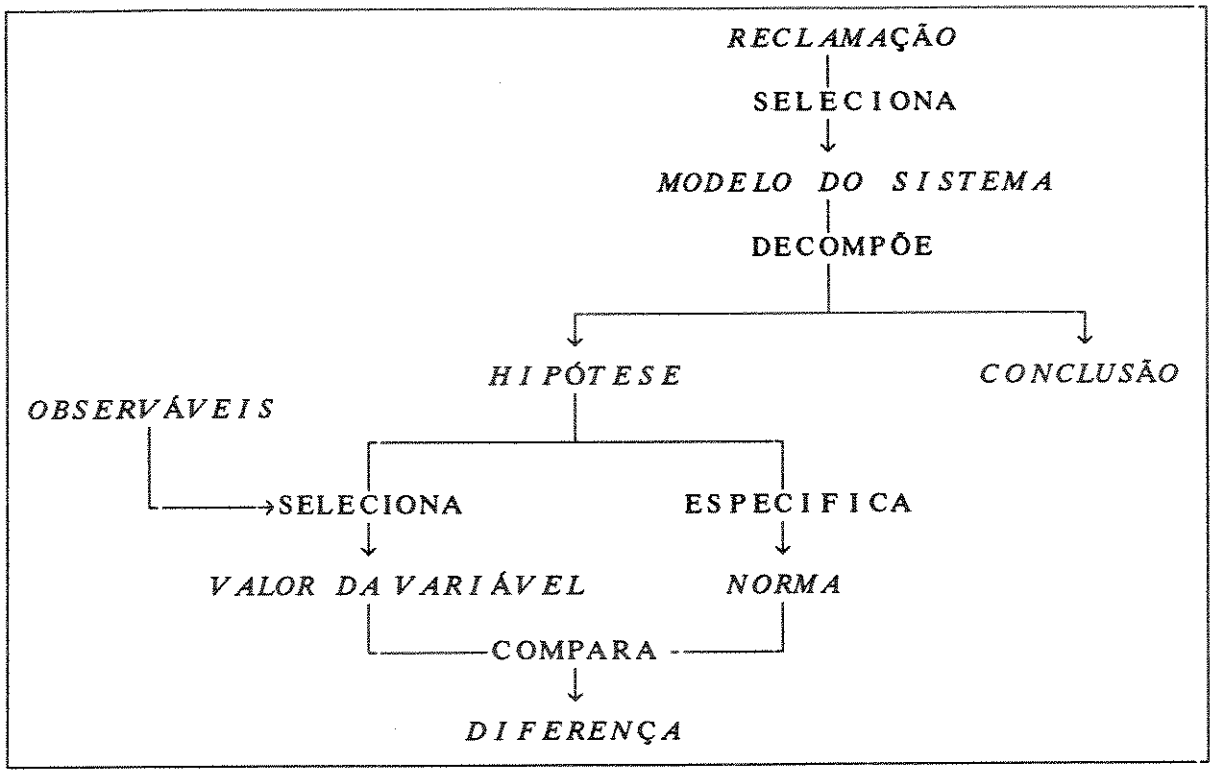


FIGURA 3.8 - Estrutura de Inferência da diagnose sistemática.

3.5.3. **A camada de tarefa**

Na camada de tarefa descreve-se o conhecimento necessário para organizar as inferência de maneira que certa meta seja atingida. Os objetos constituintes dessa camada são os elementos de controle e os objetivos. Os objetivos podem ser decompostos em sub-objetivos, que são estados desejados durante o processo de solução de um problema. As Metaclasses são exemplos de tais estados e podem desta maneira servir de ponte entre a camada de inferência e a de tarefa. A Metaclasse "conclusão", por exemplo, pode servir para indicar um possível estado desejado de um processo de solução de problema.

Na figura 3.9 apresentamos um exemplo de uma estrutura de tarefa. Nessa figura, as fontes de conhecimento servem para especificar os objetivos, por exemplo, "seleciona(modelo do sistema)" indica que o objetivo aqui é selecionar um modelo de sistema apropriado. O termo "ENQUANTO" indica um elemento de controle, cujo significado é o convencional.

```
ache(diagnóstico)
  seleciona(modelo do sistema)
    ENQUANTO (não houver conclusões)
      seleciona(valor da variável)
      especifica(norma)
      compara(valor da variável com a norma)
```

FIGURA 3.9 - Estrutura da tarefa da diagnose sistemática.

3.5.4. **A camada de estratégia**

Nem sempre a forma como um problema deve ser resolvido, descrita na camada de tarefa, é fixa. Por vezes é necessário adaptar o processo de solução a circunstâncias tais como a disponibilidade de

informações e recursos, e outros fatores tais como a competência e intensões do cliente/usuário. Essa adaptabilidade do comportamento caracteriza um conhecimento estratégico e deve ser representado na camada de estratégia.

As primitivas da KCML para essa camada são os planos, as meta-regras, os reparos e os impasses. Deve ser ressaltado que a camada de estratégia dificilmente precisa ser empregada pois os SBC normalmente possuem uma estratégia fixa de resolução de problemas. A presença da camada de estratégia na KCML visa principalmente garantir a completeza dessa linguagem.

3.6. O MODELO DE INTERPRETAÇÃO E AS TAREFAS GENÉRICAS

O Modelo de Interpretação é uma descrição em alto nível do conhecimento especializado. Ele é um modelo inicial obtido no começo da fase de análise que serve de guia para o processo de análise dos dados. O Modelo de Interpretação não faz referências ao conhecimento do domínio pois essa camada é preenchida durante a fase de análise. Um Modelo de Interpretação contém uma descrição KCML dos aspectos invariantes de tipos de tarefas comuns.

3.6.1. As tarefas genéricas

Para tornar modular a concepção de um Modelo de Interpretação é conveniente introduzir o conceito de tarefas genéricas que são tarefas associadas a problemas elementares. As tarefas genéricas devem possuir como entrada um problema e fornecem como saída uma

solução, ainda que essa solução possa ser empregada em outras tarefas. É uma das premissas do KADS que essas tarefas são limitadas em número. Outra premissa básica é que as tarefas da vida real podem ser descritas como uma composição de tarefas genéricas. Assim sendo, um Modelo de Interpretação de qualquer tarefa real pode ser obtido pela composição dos Modelos de Interpretação de tarefas genéricas.

Os Modelos de Interpretação das tarefas genéricas são os elementos básicos para o uso de Modelos de Interpretação no KADS. Embora o conceito de tarefas genéricas possa ser encontrado também em outros trabalhos [BYLANDER 86], [MOTTA 89], o mais completo conjunto de modelos de tarefas genéricas já proposto, tanto em abrangência quanto em detalhe é fornecida pelo KADS no que é chamado de biblioteca de modelos de interpretação [BREUKER 87b].

Para que essa composição possa ser feita é necessário inicialmente determinar todas as possíveis tarefas genéricas. Para tanto o KADS propõe a taxonomia de tarefas genéricas vista na figura 3.10.

```
análise de sistemas
      identificação
            classificação
                  classificação simples
                  diagnose
                        diagnose de falha única
                              classificação heurística
                               diagnose sistemática
                                     busca causal
                                     localização
                        diagnose de múltiplas falhas
                  avaliação
            supervisão
      predição
            predição de um comportamento
            predição de valores
modificação de um sistema
      reparo
      remediar
      controle
            manutenção
síntese de sistemas
      transformação
      projeto
            projeto por transformação
            projeto por refinamento
                  projeto por refinamento de caminho único
                  projeto por refinamento de caminhos múltiplos
            configuração
      planejamento
      modelagem
```

FIGURA 3.10 - Taxonomia de tarefas genéricas.

A divisão principal das tarefas é entre tarefas analíticas, sintéticas e tarefas de modificação. As tarefas analíticas visam identificar propriedades ou comportamentos desconhecidos de um sistema. Uma propriedade desconhecida pode, por exemplo, ser um componente defeituoso de um dispositivo. Um comportamento desconhecido pode ser a predição do estado de um sistema, a partir de um conjunto de valores iniciais dos parâmetros. As tarefas analíticas não alteram a estrutura do sistema.

Já as tarefas sintéticas visam determinar uma descrição estrutural do sistema em termos de um conjunto dado de elementos (vocabulário), formalismos ou estruturas parciais. Como exemplos de tarefas sintéticas temos projeto e planejamento. Entre as tarefas sintéticas e analíticas, temos as tarefas de modificação que possuem um pouco desses dois tipos de tarefa.

3.6.2. A construção de um Modelo de Interpretação

A seleção ou construção de um Modelo de Interpretação não é considerada pelo KADS uma atividade específica da metodologia pois o Modelo de Interpretação é uma ferramenta auxiliar e não um fim em si mesmo. A escolha do Modelo de Interpretação é basicamente uma atividade heurística que ocorre a partir de pistas durante a fase de análise.

Essa construção é feita inicialmente por meio de uma identificação das tarefas existentes do problema, em seguida seleciona-se um modelo para cada uma dessas tarefas e por fim é feita uma avaliação da adequação do modelo como um todo. As principais atividades envolvidas na seleção de um Modelo de Interpretação são apresentadas a seguir:

1) IDENTIFICAR TAREFAS

a. Decompor a tarefa, se possível, em sub tarefas.

b. Determinar as tarefas prioritárias e possíveis seqüências implícitas.

c. Verificar se a composição de tarefas é invariante com relação aos problemas típicos.

2) SELECIONAR MODELOS

d. Determinar as características das soluções e dos resultados intermediários.

e. Escolher possíveis Modelos de Interpretação genéricos aplicáveis.

f. Verificar se há algum modelo da vida real que incorpora a mesma combinação de tarefas genéricas. Se não houver, e se houver mais de uma tarefa genérica, combina-se os Modelos de Interpretação.

3) AVALIAR A ADEQUAÇÃO DO MODELO

g. Observar se os dados contém descrições dos passos de inferência.

h. Verificar se os dados são confiáveis e suficientemente gerais/abstratos para caracterizar um passo de inferência.

i. Confirmar se esses passos de inferência refletem as fontes de conhecimento da Estrutura de Inferência do Modelo de Interpretação. Em caso afirmativo a construção do modelo detalhado e da base de conhecimento pode começar.

j. Caso contrário verificar se o modelo pode ser modificado.

3.6.3. **Diagnose sistemática por localização**

Vamos agora apresentar um exemplo de um Modelo de Interpretação da biblioteca KADS. O exemplo é a tarefa genérica de diagnose sistemática por localização [BREUKER 87b]. A diagnose sistemática é um tipo de diagnose que se caracteriza por possuir uma estrutura de raciocínio estruturada. A diagnose sistemática pode ser feita por localização ou por busca causal. Em ambos os casos a estrutura de inferência é a mesma e pode ser vista na figura 3.8.

Na diagnose por localização, a partir de uma reclamação do usuário referente a um defeito em certo equipamento, seleciona-se uma representação adequada para o equipamento, um modelo do sistema que será usado. Esse modelo do equipamento é decomposto em sub partes, passando cada sub parte a representar uma hipótese de onde está a falha. Se não for possível decompor o modelo então obteve-se uma conclusão, ou seja, a causa da falha foi encontrada. Caso contrário, dada uma hipótese de falha, especifica-se o comportamento do equipamento do sistema que se esperaria observar caso essa hipótese fosse incorreta e, ao mesmo tempo, seleciona-se no equipamento sintomas que possam ser úteis para confirmar/refutar a hipótese. Compara-se a predição feita com o observado no equipamento. Se houver diferença, a hipótese está correta e o processo de diagnose prossegue procurando localizar a falha ao nível de um componente reparável.

As Metaclasses necessárias para a diagnose sistemática por localização são:

```
MODELO DO SISTEMA       : decomposição hierárquica parte-de.
RECLAMAÇÃO              : sistema falho.
UNIVERSO DE OBSERVÁVEIS: variáveis de saída observáveis.
HIPÓTESES               : (sub)sistema contendo um componente falho.
VALOR DE VARIÁVEL       : valor de saída observado.
NORMA                   : especificação da saída.
DIFERENÇA               : componente falho.
CONCLUSÃO               : componente falho.
```

As Fontes de Conhecimento necessárias para a diagnose sistemática por localização, são:

SELECIONE UM MODELO DE SISTEMA
FONTE DE CONHECIMENTO: **seleciona**. A diagnose começa com a seleção de um modelo de sistema.
ENTRADA: **reclamação**. A reclamação pode ser acerca de um sistema com pelo menos um componente falho.
SAÍDA: **modelo do sistema**. O modelo do sistema consiste de uma hierarquia de componentes. Uma vez que o modelo pode ser facilmente decomposto a seleção poderá consistir de um sub-modelo relevante do modelo completo do sistema.
MÉTODO: associação direta.
CONHECIMENTO DO DOMÍNIO: conhecimento acerca do comportamento e sobre a estrutura do sistema.

DECOMPONHA O MODELO DO SISTEMA
FONTE DE CONHECIMENTO: **decompõe**. O modelo é decomposto em sub-modelos no próximo nível da hierarquia da decomposição do equipamento.
ENTRADA: **modelo do sistema**.
SAÍDA: **hipótese** (um componente falho localizado em alguma parte do sistema) ou **conclusão** (um componente que não pode ser mais decomposto).
MÉTODO: descer a hierarquia da decomposição do equipamento.
CONHECIMENTO DO DOMÍNIO: decomposição hierárquica do equipamento.

SELECIONE UM VALOR DE VARIÁVEL
FONTE DE CONHECIMENTO: **seleciona**. Um valor de variável é selecionado do universo de observáveis de saídas de componentes.
ENTRADA: **hipótese** e **universo de observáveis**.
SAÍDA: **hipótese** (um componente falho localizado em alguma parte do sistema) e **conclusão** (elemento básico da decomposição do equipamento).
MÉTODO: descer a hierarquia da decomposição funcional do equipamento.
CONHECIMENTO DO DOMÍNIO: decomposição funcional do equipamento.

DETERMINE A NORMA
FONTE DE CONHECIMENTO: **especifica**. Essa fonte de conhecimento especifica a saída normal, ou seja, o valor correto esperado de saída ou uma função dado algum valor ou função de entrada, do componente aonde é assumido que esteja a falha.
ENTRADA: **modelo do sistema**.
SAÍDA: **norma**.
MÉTODO: associação direta.
CONHECIMENTO DO DOMÍNIO: conhecimento acerca do comportamento.

COMPARA O VALOR DA NORMA COM O VALOR DA VARIÁVEL
FONTE DE CONHECIMENTO: **compare.**
ENTRADA: **valor da variável e norma.**
SAÍDA: **diferença.** Diferença entre o valor de saída esperado e observado que é usado para reiniciar o processo de localização. A diferença é um componente cujo comportamento não é o esperado.
MÉTODO: comparação de valores ou funções.
CONHECIMENTO DO DOMÍNIO: relevância de graus de diferenças (por exemplo, intervalos de confiabilidade, etc.).

A Estrutura da Tarefa para a diagnose sistemática por localização é apresentada a seguir:

```
ache(diagnóstico)
  seleciona(modelo do sistema)
    ENQUANTO (não houver conclusões)
      seleciona(valor da variável)
      especifica(norma)
      compara(valor da variável com a norma)
```

"ENQUANTO (não houver conclusões)" é um elemento de controle, as demais primitivas são objetivos. A estratégia de diagnose é fixa.

3.7. AS FERRAMENTAS DO KADS

A primeira ferramenta desenvolvida pelo KADS para dar suporte à sua metodologia foi o sistema **KADS-3**. Esse sistema foi concebido para guiar o usuário através das etapas prescritas pela metodologia KADS e permitir o acesso a ferramentas de auxílio a cada uma delas. O KADS-3 foi considerado muito restritivo, impondo uma tutela excessiva para usuários mais experientes. Adotando uma abordagem oposta, foi então construído o **KPT** (KADS POWER TOOL) que consiste num sistema que

agrupa um certo conjunto de ferramentas que auxiliam a construção do SBC sem no entanto fornecer aconselhamentos mais gerais.

O KADS Power Tool é composto de um editor de descrições verbais (protocols) - cujo núcleo é um sistema de hipertexto que permite editar as informações colhidas oralmente de um especialista, um editor de conceitos e uma biblioteca de Modelos de Interpretação. O KPT foi extensivamente testado em uma série de casos reais e em termos gerais aprovado. Porém algumas falhas foram notadas, tais como uma falta de integração entre as diversas ferramentas, falta de um aconselhamento e auxílio geral e uma arquitetura um tanto rígida que dificultava a inclusão de outras ferramentas [WIELLINGA 89].

Procurando superar essas dificuldades um sistema mais completo foi concebido: o **SHELLEY**, que não é propriamente uma ferramenta, mas um ambiente, ou seja um conjunto de ferramentas que operam sobre uma mesma base de dados. Essas ferramentas são denominadas Activity Support Tools (AST), e se subdividem nos seguintes tipos:

- ASTs de auxílio à Aquisição de Conhecimento através da análise de protocolos, definição de léxicos, etc.

- ASTs de auxílio à modelagem através de um editor conceitual, da manipulação de estruturas de inferência, etc.

- ASTs de modalidade pela distribuição das tarefas, pela avaliação dos modelos, etc.

- ASTs para gerenciamento do projeto através do modelo do ciclo de vida KADS.

Essas ferramentas operam sobre uma base de dados comum que é chamada de base de objetos persistentes ou base de modelos. Entre as facilidades fornecidas pelo SHELLEY encontramos a possibilidade de

definir centralmente a interface com o usuário, browsers para relações e atributos, geração de documentação do projeto, aconselhamento, etc.

No SHELLEY procurou-se fornecer um auxílio e aconselhamento geral e buscou-se integrar completamente as diversas ferramentas que o compõem, pontos fracos do KPT. Esse novo sistema de apoio ao uso da metodologia KADS é recente e ainda não se encontrava concluído quando do término do projeto ESPRIT KADS em fevereiro de 1990. O SHELLEY foi concebido para operar em estações de trabalho da SUN, sistema operacional UNIX.

3.8. CONCLUSÃO

O projeto KADS é a mais completa e ambiciosa tentativa que podemos encontrar na literatura de analisar as atividades realizadas pelo especialista ao resolver um problema, o que é feito por meio dos modelos de interpretação. O KADS é uma metodologia completa desenvolvida especificamente para facilitar e organizar a construção de SBCs que fornece elementos únicos na literatura tal como a sua biblioteca de modelos de interpretação.

A metodologia KADS aborda todos os aspectos da Engenharia do Conhecimento, desde o processo de Aquisição de Conhecimento, passando pelos aspectos relativos à representação (a linguagem KADS de modelagem conceitual) e de interação com o usuário, até a implementação propriamente dita do sistema.

A grande falha do projeto KADS é o de não fornecer uma descrição acessível e compreensível de sua metodologia. Tanto é assim

que a maior parte dos artigos referenciados nesse texto foram obtidos diretamente da Universidade de Amsterdam. O KADS falha também em não se apresentar numa forma fortemente integrada e direcionada. Outro ponto fraco do KADS é uma negligência à representação do conhecimento: não há apoio ou modelos concernentes à camada de domínio [GREEF 87].

A metodologia KADS é um padrão Europeu emergente em Engenharia do Conhecimento que já possui penetração no mercado norte-americano. O KADS serviu de base para a metodologia SKE (Structured Knowledge Engineering) que atualmente está sendo comercializada pela Boolesian na Europa e Estados Unidos.

CAPÍTULO IV

O MODELO DE DIAGNOSE

4.1. INTRODUÇÃO

Nos capítulos anteriores apresentamos as noções gerais relativas ao problema da Aquisição de Conhecimento, em particular algumas ferramentas de Aquisição de Conhecimento, e abordamos a metodologia KADS, que utilizamos na construção dos modelos da nossa ferramenta. Nesse capítulo trataremos da diagnose, domínio ao qual se destinará tal ferramenta. Com isso visamos, primeiro compreender melhor essa área para, em seguida, desenvolver um modelo de diagnose para a ferramenta.

Diagnose é a atividade que visa determinar a explicação da falha de um equipamento. Falha é a não realização de uma função que o equipamento deveria realizar ou a realização de alguma função inesperada. Diagnóstico é o resultado final da diagnose, ou seja, diagnose é o processo de obtenção do diagnóstico. Teste é um procedimento que visa determinar a ocorrência de algum sintoma. Sintomas são condições observáveis associadas a falhas, a uma função específica, a estados normais ou anormais ou a explicações [STEELS 89].

Para que se possa diagnosticar e reparar um equipamento devemos ter em mente alguma estratégia geral, precisamos conhecer com alguma profundidade o equipamento e devemos estar aptos a realizar testes e reparos de uma maneira eficiente. Esses conhecimentos não são necessariamente separados: é possível representar todos essas informações através de regras que forneçam conclusões e as ações que devam ser realizadas de uma maneira direta. Outra forma de construir um sistema de diagnose é com o uso de modelos aonde temos uma

representação explícita do equipamento, do processo geral de diagnose e reparo e, eventualmente, também da forma como o sistema deve interagir com o usuário (figura 4.1).

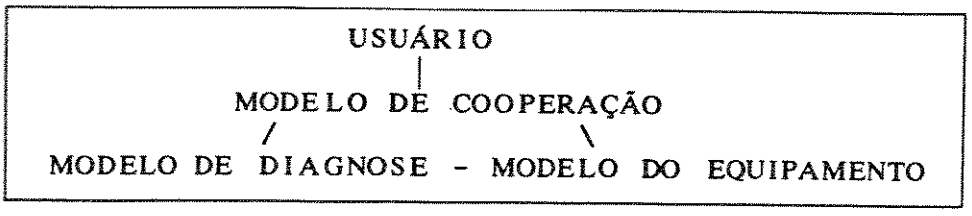


FIGURA 4.1 - Os modelos no problema da diagnose

O modelo de diagnose explicita a forma como a diagnose é feita. O modelo de cooperação representa o conhecimento acerca da interação entre o usuário e o sistema e o modelo do equipamento contém informações sobre o equipamento em si. Esses são os três "tipos" de conhecimento necessários para construir um sistema de diagnose, e é sobre eles que a partir de agora passaremos a tratar.

4.2. O MODELO DA DIAGNOSE

Existem duas técnicas básicas para realizar a diagnose: diagnose baseada em sintomas e diagnose baseada em modelo. Essa última se subdivide em diagnose por modelo de falhas e diagnose por modelo da estrutura e comportamento do sistema projetado [HAMSCHER

89]. A diagnose por comportamento do sistema projetado, por sua vez, pode ser feita por localização ou por busca causal [BREUKER 87]. A figura 4.2 resume essa classificação:

* Diagnose baseada em sintomas (diagnose heurística)
* Diagnose baseada em modelo
  - modelo do comportamento do sistema falho (modelos de falhas)
  - modelo do comportamento e estrutura do sistema projetado
    . localização
    . busca causal

FIGURA 4.2 - As técnicas para a diagnose.

Atualmente a técnica dominante para automatizar a diagnose é a baseada em sintomas que se utiliza de um conjunto de regras heurísticas. Tais sistemas, porém, possuem as graves deficiências de todo sistema especialista de primeira geração: é difícil saber se eles cobrem todas as falhas possíveis, é complexo fazê-los trabalhar com informações inconsistentes vindas, por exemplo, de sensores falhos e, principalmente, tais sistemas precisam ser totalmente refeitos quando ocorrem alterações no equipamento que eles diagnosticam [FULTON 90].

Essas deficiências levaram ao surgimento dos sistemas especialistas de segunda geração que possuem um conhecimento do domínio no qual operam. Esse conhecimento é chamado conhecimento profundo e se contrapõe ao conhecimento superficial das regras ad

hoc. O conhecimento profundo habilita o sistema especialista a trabalhar com casos não previstos inicialmente e a lidar com equipamentos diferentes.

Fink [FINK 85] [FINK 87] define conhecimento profundo (deep knowledge) como sendo o conhecimento básico relacionado com o COMO e o PORQUÊ o dispositivo trabalha de tal forma e procedimentos para corrigir seus defeitos, fornecendo as informações de causa e efeito necessárias quando se diagnostica uma falha. Já o conhecimento superficial (shallow knowledge) é o conhecimento baseado na experiência adquirida diagnosticando o problema. Ele fornece atalhos através do problema, permitindo ao especialista solucioná-lo mais rápida, eficiente e precisamente que um novato.

Conforme já mencionado a diagnose baseada em modelo se divide em diagnose por modelo do sistema falho e por modelo do sistema projetado. A diagnose por modelo do sistema falho emprega um conhecimento profundo acerca das falhas do equipamento, ou seja, o conhecimento acerca de como e porque o sistema falha. Para Steels [STEELS 89], um modelo de falhas envolve informações acerca das dependências funcionais entre os módulos dos equipamentos, por exemplo, o circuito de vídeo de um televisor, para funcionar, depende da fonte de alimentação do sistema, e informações acerca das possíveis causas dos defeitos e as explicações para as mesmas. Um módulo é um conjunto de componentes que executam a mesma função.

Para Steels, causa é um estado anormal de um componente que o leva a provocar a falha no equipamento, por exemplo, uma causa possível para que um carro não queira funcionar é a falta de combustível no tanque. Já explicação é uma condição interna ou

externa que levou à ocorrência da causa, por exemplo, uma explicação para a falta de combustível é um furo no tanque ou o esquecimento do proprietário do veículo de abastecê-lo com combustível.

A diagnose por modelo do sistema projetado emprega um conhecimento profundo sobre como o sistema deve operar em situações normais. Duas formas de se realizar diagnose com esse tipo de conhecimento são a estratégia de localização e a de busca causal, que na verdade são muito semelhantes entre si, tanto assim que a metodologia KADS propõe a mesma estrutura de tarefa para ambas [BREUKER 87b]. A localização segue um processo de isolamento do problema em estruturas funcionais cada vez mais elementares, como quando, para diagnosticar um defeito em um carro, procuramos inicialmente determinar se o problema se encontra na parte elétrica ou na parte mecânica. A estratégia de busca causal trabalha com a descrição das dependências funcionais entre os componentes do equipamento (rede causal) procurando delimitar o problema dividindo a rede causal em sub-redes cada vez menores. Parte-se, nesse caso, da manifestação do defeito e busca-se determinar quais elementos do equipamento poderiam causar tal falha.

É interessante salientar a distinção entre a diagnose por localização e baseada em modelo de falhas. Enquanto nessa última temos uma hierarquia de falhas baseada em relações de herança de propriedades, na primeira temos relações de superclasse de características. Por exemplo, na diagnose por modelo de falhas podemos ir da hipótese "doença pulmonar" para a hipótese "gripe" aonde a gripe herda todas as propriedades de doença pulmonar, ou seja a gripe pertence ao conjunto das doenças pulmonares. Já numa diagnose

por localização podemos ir da hipótese "defeito no módulo de som" para a hipótese "defeito no amplificador". O módulo amplificador é um sub-módulo do módulo de som, conseqüentemente tem todas as funções desse, o amplificador está contido no módulo de som. Uma disfunção mais específica tem todas as propriedades de uma mais geral, ao contrário de uma função mais específica que tem apenas parte das características de uma menos específica.

4.2.1. A tarefa de diagnose

De maneira relativamente independente do modelo que se empregue para diagnosticar o equipamento, existe uma infinidade de maneiras diferentes de organizar a busca do defeito em um equipamento, Freiling e Jacobson [FREILING 89] citam as seguintes:

a) Busca cega. Consiste em selecionar aleatoriamente um teste de um componente e aplicá-lo, até ser possível obter um diagnóstico;

b) seleção de módulo. Consiste em selecionar um módulo do equipamento e aplicar exaustivamente (por seleção aleatória) todos os testes associados a esse módulo;

c) localização "top-down". Começa testando um módulo que não seja sub-módulo de nenhum outro, se esse se mostrar defeituoso seleciona-se para teste um sub-módulo do módulo em questão. Se os testes desse sub-módulo não indicarem a presença de uma falha, retorna-se para o módulo que contém esse sub-módulo e seleciona-se outro sub-módulo para ser testado.

Um problema da localização top-down é que ela é muito trabalhosa, exigindo vários passos de decomposição hierárquica do dispositivo para localizar a falha. A experiência demonstra, porém,

que os especialistas são freqüentemente capazes de ir diretamente ao âmago da questão e diagnosticar o equipamento com apenas alguns testes;

d) seleção heurística de módulo. Seleciona-se como foco de atenção o equipamento como um todo. Quando um foco de atenção é escolhido, primeiro usa-se as regras de seleção para esse módulo para ver se o foco de atenção precisa ser mudado para outro módulo. Se não for o caso, faz-se o teste funcional do módulo. Prossegue-se por meio de uma localização top-down;

e) minimização de ajustes. Inicialmente aplicam-se os testes que tenham menor custo, que não precisem de ajuste para serem feitos. Prossegue-se com testes de custo cada vez maior. Sempre que um ajuste é feito faz-se todos os testes que necessitem desse ajuste. Pode-se também combinar seleção de módulo com minimização de ajustes, nesse caso primeiro seleciona-se um módulo e em seguida prossegue-se pela minimização dos ajustes;

f) minimização de ajustes dentro da seleção de módulo. Usa seleção heurística de módulo como estratégia inicial mas, sempre que um ajuste é feito, realiza-se todos os teste que requeiram o ajuste, independente do módulo a que pertençam;

g) seleção oportunista de módulo. Cada teste acrescenta uma crença sobre o estado de outros módulos. Seleciona-se o módulo que está hierarquicamente mais próximo às partes que podem ser substituídas e tenham maior grau de crença em sua falha.

Como pode ser observado existem diversas maneiras de se realizar a diagnose de um equipamento. O nosso objetivo é propor um

modelo suficientemente maleável para permitir que diversas delas possam ser abrangidas pelo modelo apenas com pequenas alterações. Esse modelo que descreveremos engloba elementos de uma estratégia de localização e de um modelo de falhas. A localização é a estratégia básica para determinar a causa da falha, porém, para gerar as hipóteses de falha mais prováveis a cada momento, utilizamos um modelo de falhas, que contém a heurística da diagnose.

Tal modelo engloba uma diagnose guiada por uma função heurística que determina qual o próximo módulo a ser testado. O uso de uma função heurística adequada permite que se obtenha diferentes versões das políticas de diagnose previamente citadas. Nessa abordagem todo módulo possui uma probabilidade de falha que é a crença que temos que ele esteja defeituoso, sendo essa probabilidade alterada sempre que um teste é feito, essa alteração depende do resultado do teste e das relações entre os módulos do dispositivo.

Devemos distinguir claramente a probabilidade de falha de um módulo da probabilidade dele não operar. A probabilidade de falha de um módulo reflete a crença que depositamos em que ele esteja com defeito. A não operação de um módulo, porém, pode ser devida não só a ele estar com defeito como também a um dos módulos de que ele dependa estar falho. Assim sendo, embora apenas um dos módulos possa estar com defeito, muitos poderão estar não operacionais. A probabilidade de falha de um módulo é independente da dos demais pois ela reflete a probabilidade de defeito e não a possibilidade de não operação.

A maneira como trabalhamos com essas crenças é semelhante à usada no FIS, Fault Isolation System [PIPITONE 86], que é um sistema de diagnose para dispositivos eletrônicos analógicos que trabalha com

um modelo qualitativo do comportamento, ou seja emprega descrições qualitativas sobre as relações existentes entre a entrada e a saída de cada componente do sistema. Como, ao contrário do FIS, nosso modelo não possui informações acerca do comportamento físico dos componentes do sistema, mas tão somente uma descrição teleológica dos mesmos (uma descrição funcional) algumas alterações foram feitas nessa estratégia. A opção por uma descrição funcional e não do comportamento visa dar uma abrangência maior ao modelo, o que será discutido posteriormente. Além disso, impusemos a simplificação de trabalharmos apenas com a hipótese de falha simples, ou seja, supomos que a cada momento apenas um componente encontra-se defeituoso.

A diagnose por localização, que consiste em delimitar o conjunto de componentes possivelmente falhos a um número cada vez menor, possui como fraqueza a necessidade de que os módulos do equipamento possam ser isolados para fins de teste [BREUKER 87b], o que freqüentemente, embora possível, não é prático. Quando os componentes não podem ser testados isoladamente, devemos levar em conta, durante a diagnose, a forma como eles interagem entre si, e como essa interação se reflete no resultado dos testes. Em casos reais esse tipo de análise é complexo pois os módulos interagem de modos qualitativa e quantitativamente diversos. Além disso, e principalmente, os testes alteram a natureza desta interação, como quando isolamos (total ou parcialmente) o componente a ser testado do restante do equipamento para testá-lo. Por exemplo, uma lâmpada piloto para funcionar depende do funcionamento da fonte de alimentação do equipamento, porém, um teste da lâmpada piloto pode ser feito retirando-se a lâmpada de seu soquete e verificando a

continuidade de seu filamento, eliminando assim a dependência da lâmpada e fazendo com que o resultado desse teste isoladamente nada permita concluir acerca da fonte de alimentação.

Para tratar com profundidade esse tipo de problema seria necessário nos restringirmos a domínios específicos, como o faz o FIS. Como não desejamos tal limitação para o domínio do modelo, a solução que se adotou foi compilar a informação acerca das interações e dependências entre os módulos do equipamento no que elas têm de relevante ao processo de seleção e teste de hipóteses. Para tanto empregamos uma tabela de falhas do equipamento no qual estão implícitas as dependências entre os possíveis sintomas e as falhas dos módulos. Essa tabela, juntamente com a representação das explicações, formam o modelo de falhas do equipamento.

Assim, por exemplo, ao invés de representarmos diretamente a dependência de uma lâmpada com a fonte de alimentação (que nos habilita a concluir que se a fonte estiver com problema a lâmpada não acenderá) explicitamos diretamente que um dos sintomas associados à falha da fonte é o não acendimento da lâmpada. Temos assim uma descrição mais simples do equipamento do que se tivéssemos, por exemplo, a equação da relação existente entre a luminosidade da lâmpada e a energia fornecida pela fonte de alimentação.

A desvantagem dessa solução é a excessiva dependência das informações fornecidas pelo especialista acerca dos sintomas associados às falhas. Para superar tal limitação é que recorre-se a um conhecimento profundo representado pela estratégia de localização que não depende do particular equipamento sob teste. Assim, quando não houver um conhecimento mais especializado, heurístico, acerca da

falha em particular, é possível recorrer, assim como faria um especialista humano, a um conhecimento da diagnose em geral.

Outra diferença com o FIS, é que trabalhamos com a hipótese de que apenas um componente está falho no equipamento, pois a hipótese de falhas múltiplas, com que trabalha o FIS, leva a um crescimento exponencial da complexidade dos cálculos com o número de componentes, tornando o problema inviável para a maioria dos casos reais [PENG 87]. Como, para equipamentos usuais, a probabilidade de falha múltipla é muito pequena, justifica-se que nos limitemos apenas às falhas simples. A extensão do nosso tratamento para cobrir também falhas múltiplas não oferece, porém, grandes dificuldades conceituais, sendo necessário apenas adotar tratamento análogo ao usado pelo FIS, o que implicaria unicamente na necessidade de um maior poder computacional.

4.2.2. **Análise do resultado de um teste**

Cada módulo de um equipamento pode ser associado a um certo conjunto de sintomas que indicam quando esse módulo apresenta-se defeituoso e, inversamente, cada sintoma está associado a um certo conjunto de módulos que podem levar à presença desse sintoma. Cada teste está associado ao sintoma que ele verifica.

No nosso modelo representamos esta situação por meio de uma tabela de falhas como a da tabela 4.1.

MÓDULOS / SINTOMAS	A	B	C
S 1	X	X	X
S 2	X	X	
S 3			X
S 4	X		
S 5	X	X	

Sendo S1 = a lâmpada não acende,
S2 = a lâmpada acende mais forte que o normal,
S3 = a lâmpada acende intermitentemente,
S4 = o driver faz um barulho estranho,
S5 = o driver não faz o barulho normal.

Teste 1: "Observe a lâmpada piloto"
-> Sintomas: S1, S2 e S3.
Teste 2: "Ouça o som produzido pelo driver"
-> Sintomas: S4 e S5.

TABELA 4.1 - Exemplo de uma tabela que explicita a relação entre sintomas e causas de falhas.

Os testes não são diretamente associados aos módulos, como talvez pudesse parecer mais natural, por que cada teste fornece informações acerca da probabilidade de mais de uma hipótese de falha e cada uma dessas, para ser testada, pode depender de mais de um teste.

Vamos agora analisar a maneira como trabalhamos com essa tabela de falhas para calcular a probabilidade de falha de um componente após a realização de um teste, empregando a probabilidade a priori de falha do componente como ponto de partida.

Suponhamos, para tanto, que o teste 1 seja aplicado e que se observe o sintoma S2. Nesse caso, apenas as hipóteses "A está falho" e "B está falho" são consistentes com tal observação. Assim a probabilidade de A, B e C estarem falhos após a observação de S2

passa a ser:

$$P(A|S2) = P(A|(A \cup B)) = \frac{P(A \,\&\, (A \cup B))}{P(A \cup B)} = \frac{P(A)}{P(A) + P(B)}$$

$$P(B|S2) = P(B|(A \cup B)) = \frac{P(B \,\&\, (A \cup B))}{P(A \cup B)} = \frac{P(B)}{P(A) + P(B)}$$

$$P(C|S1) = 0$$

onde $P(X|Y)$ é a probabilidade corrente da hipótese de falha do módulo X dada a ocorrência de Y.

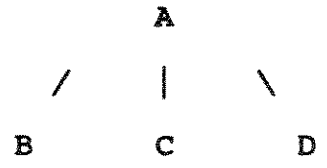
Porém se o teste 1 for feito e nenhum sintoma for observado, isso não significa que A não está com problemas pois existe ainda a possibilidade de determinar-se que A está falho se for observado o sintoma S4 ou S5. De maneira semelhante à usada no FIS, avaliamos a hipótese residual de A estar falho a partir da equação:

$$P(A \mid (\overline{S1} \wedge \overline{S2})) = (1 - n/N)\ P(A)$$

onde N é o número total de sintomas associados à falha de A e n é o número de sintomas já testados e descartados associados a A.

O algoritmo descrito para atualizar as probabilidades após a realização de um teste deve ser aplicado dentro do contexto de uma decomposição funcional do equipamento. Isso é feito empregando a relação "parte-de" para propagar as probabilidades resultantes da aplicação de um teste.

Suponhamos, por exemplo, que tenhamos a seguinte decomposição funcional:



nesse caso a propagação é feita com o auxílio da seguinte equação:

$$P(A) = P(B \cup C \cup D) = P(B) + P(C) + P(D)$$

assim toda vez que, em função do resultado de um teste, a probabilidade de qualquer dos módulos se alterar, as demais probabilidades também se alterarão para que a equação acima permaneça válida.

4.2.3. **Seleção do melhor teste.**

A estratégia de diagnose ótima é aquela que minimiza o custo de obtenção do diagnóstico. Para atingi-la seria preciso analisar todas as possíveis seqüências de testes, a árvore de testes, por meio de um algoritmo de minimização do custo médio. Isso, porém, é impraticável para árvores muito ramificadas. Adotamos assim a mesma solução do FIS: aplicamos um algoritmo de minimização do custo médio para apenas um nível de profundidade na árvore.

O teste ótimo é aquele que minimiza a função de avaliação para todos os possíveis resultados desse teste. A ponderação é feita em função das respectivas probabilidades de resultado do teste, que é a probabilidade de um certo sintoma ser observado, dada por:

$$P(Sn) = 1 - \prod(1 - f_1 \, P(A_1))$$

$P(A_1)$ é a probabilidade do módulo A_i estar falho, f_i é a fração de sintomas que ainda faltam ser verificados para o módulo A_i. Assim a probabilidade de um teste determinar certo sintoma S_n, é a probabilidade de termos a falha de um dos módulos que poderia dar origem a esse sintoma.

A função de avaliação é a soma do custo do teste menos a informação adquirida através do teste e mais um fator G de avaliação do custo de testar o restante da árvore. A informação adquirida com o teste é a mudança da seguinte fórmula da informação (a entropia do sistema):

$$E = -\sum_{i=1}^{m} p_i \log p_i$$

Aonde p_i é a probabilidade corrente de falha da $i_{ésima}$ hipótese e M é o número de hipóteses. A alteração desse valor será tanto maior quanto maior for a relevância do teste.

A função de avaliação assumirá um valor tanto menor quanto menor for a dificuldade de executar esse teste, quanto maior for a probabilidade dele detectar um certo sintoma, o que por sua vez depende das probabilidades das hipóteses que ele testa, e quanto mais próximo ele esteja de um módulo reparável.

Assim a função de avaliação será dada por:

$$H(teste) = \sum_{s=1}^{N} P(Sn)*(Custo(ajuste) + Custo(Medida) - dE)$$

$$F_{av}(teste) = H(teste) + G(teste)$$

Aonde F_{av} é a função geral de avaliação, N é o número de sintomas que o teste verifica, Custo() é uma função de avaliação do custo do teste, dE é a variação da entropia, H() é a função heurística de avaliação do teste e G() é uma função proporcional à distância do módulo sendo testado de um módulo diretamente reparável. A função G avalia o custo dos testes restantes necessários para chegar à diagnose.

A estratégia geral de diagnose dependerá dessa função heurística de avaliação, que por sua vez depende da forma como definimos a função custo do teste e também da definição de G(). Se a função de custo tiver valor relativo alto, teremos uma política de minimização de ajustes, caso a variação da entropia tenha grande peso teremos diagnose por localização top-down, pois serão preferidos testes de amplo espectro e focalizados para módulos de alta probabilidade de falha, que são justamente os módulos hierarquicamente mais elevados do dispositivo. Assim dependendo da função heurística usada teremos toda uma gama de estratégias diferentes de diagnose.

4.2.4. A tarefa de reparo

O reparo é uma tarefa que obtém uma explicação plausível para a causa da falha do equipamento e restaura esse equipamento ao seu estado normal. O não funcionamento de um aparelho pode ser causado por um componente falho, pelo uso inadequado do mesmo ou por ambos os motivos. No caso de haver um componente falho a solução é prescrever um reparo, no caso de ter ocorrido uso inadequado, o usuário deve ser avisado para que proceda de maneira correta no futuro.

Uma vez determinado qual componente do equipamento está falho, para que se possa repará-lo, é necessário determinar o que causou a falha. Por exemplo, se a motocicleta não funciona por falta de combustível, o reparo dessa falha dependerá da causa ser um furo no tanque ou ela não ter sido reabastecida.

A explicação de uma falha pode ser de três tipos: uma falha intrínseca ao componente falho (por exemplo, a lâmpada queimou por que estava velha), poderá ser a falha de um outro componente do dispositivo (por exemplo, a lâmpada queimou por que a fonte estava fornecendo uma tensão muito alta) ou poderá ser um agente externo (por exemplo, a lâmpada não está acendendo por que está faltando energia) [STEELS 89]. A explicação não pode ser inferida pelo uso de modelos pois para tanto seria necessário modelar não apenas o equipamento como também o mundo externo e as interações sutis entre os componentes do equipamento. No modelo que propomos as explicações são obtidas a partir da aplicação de um conjunto de regras heurísticas.

4.3. O MODELO DO EQUIPAMENTO

Alguns tipos de modelos que podem ser formulados para representar o equipamento a ser diagnosticado são:

1) modelo físico;

2) modelo do comportamento;

3) modelo funcional;

O modelo físico representa o equipamento por meio da disposição espacial dos seus constituintes ou pela descrição da conecção entre os componentes, o modelo do comportamento descreve o comportamento físico do dispositivo e o modelo funcional representa o equipamento em termos das operações que ele realiza. O modelo físico pode ser estrutural ou geométrico. A descrição estrutural [MILNE 87] refere-se à ligação dos elementos, por exemplo as conecções entre os componentes de um equipamento elétrico. A geométrica [STEELS 89], diz respeito à disposição desses componentes, por exemplo, as coordenadas das trilhas em uma placa de circuito impresso ou das engrenagens de uma máquina.

Podemos colocar a diferença entre essas descrições dentro do paradigma de linguagens declarativas, linguagens voltadas para a descrição de procedimentos e linguagens orientadas para objeto. O modelo físico fornece uma descrição declarativa do equipamento ou seja, é um conjunto de dados que deve ser manipulado por algum algoritmo extrínseco a estas informações para adquirir significado. Já a descrição do comportamento é procedimental, ela descreve o equipamento a partir dos processamentos que nele ocorrem. Por sua vez, a descrição funcional é orientada para objetos, ela associa a cada componente as suas características. Os módulos adquirem uma função, um objetivo, que devem atingir, como se fossem entidades vivas.

Dentre as alternativas de modelagem apresentadas, nossa opção foi pela funcional. Cumpre então distinguir claramente a diferença

entre um modelo funcional e um do comportamento: o primeiro associa a cada componente um objetivo, enquanto o segundo associa ações físicas e não teleológicas. Na descrição do comportamento há sempre uma descrição dos módulos em termos de uma função, contínua ou discreta, que associa estados de entrada a estados de saída. Na descrição funcional podemos ter também uma descrição em termos de entrada/saída do módulo, apenas que na descrição funcional tanto a entrada quanto a saída são estados teleológicos, como quando dizemos que certo componente realiza certa função quando entramos com certo comando. Já na descrição do comportamento temos uma equação numérica (por exemplo, a lei de Ohm: V = R.I) ou uma versão qualitativa dessa (por exemplo, se a corrente é alta e a resistência não é pequena, então a tensão é alta). Caso a descrição dos componentes em termos de entrada/saída não seja fornecida então teremos um modelo causal [STEELS 89], caso em que apenas as dependências entre os componentes é fornecida em termos de: se X não funciona então Y também não opera.

O modelo funcional é composto pela descrição estrutural do equipamento e pela descrição das funções que cada elemento da estrutura executa. Ao contrário de Chandrasekaran [STICKLEN 89] que utiliza um modelo funcional baseado numa hierarquia de componentes cada qual associado a um conjunto de funções, nós empregamos uma hierarquia de funções onde, naturalmente, cada parte da decomposição funcional está associada a apenas uma função [STEELS 89].

A limitação normalmente apontada nos modelos de falha é que a descrição exaustiva de todas as falhas é muito extensa. O uso de uma decomposição funcional em conjunto com o modelo de falhas ajuda a

superar essa limitação. Suponhamos, por exemplo, que num equipamento tenhamos 5 módulos amplificadores usados para fins diferentes, sendo cada um composto por 5 circuitos integrados. Na abordagem convencional teríamos que descrever 25 modos de falhas referentes à maneira como a falha de cada um dos 25 circuitos afetaria o funcionamento do equipamento como um todo. Na nossa abordagem, temos que descrever apenas 10 modos de falha relativos à forma como a falha de cada um dos amplificadores afeta o dispositivo como um todo e a forma como a falha de cada circuitos integrados afeta o funcionamento do amplificador.

Parece-nos, também, que essa é uma descrição mais natural para o especialista que ao ter restringido a falha a um módulo M durante o processo de localização, concentra-se apenas em M, ou seja, ele não pensa em como a falha de um sub-módulo afeta o equipamento como um todo, mas sim em como ela afeta apenas o módulo M.

A obtenção de todos os sintomas associados a um equipamento é um trabalho muitas vezes infactível devido ao elevado número de sintomas possíveis. Porém, como trabalhamos com uma hierarquia de funções, não é necessário obter todos os sintomas, podemos nos limitar a uma descrição completa dos sintomas até o nível da hierarquia funcional que julgarmos conveniente. Existem, assim, diversos níveis possíveis de completeza na construção do modelo de falhas.

O modelo funcional que empregamos é composto pelas relações "parte-de", "explica", "repara" e "testa" e pelos conceitos que descrevem os componentes e os módulos funcionais do equipamento.

4.4. O MODELO DE COOPERAÇÃO

O modelo de cooperação descreve a forma como o especialista e o sistema interagem durante a diagnose. Existem dois tipos de interações entre o usuário e o sistema de diagnose: aquela que é de iniciativa do sistema e a que é dirigida pelo usuário. Durante a diagnose as mensagens para o usuário são necessárias para fins de prescrição de testes e reparos (e/ou aconselhamentos acerca de como o usuário deverá proceder futuramente para que o equipamento não apresente novamente a falha) e para informar acerca da causa e da explicação da falha. Na direção oposta, o usuário pode inquerir o sistema acerca de conclusões atingidas, sobre a motivação de um teste ou reparo e acerca de questões mais gerais relativas à diagnose como um todo.

4.4.1. Interações de iniciativa do sistema.

4.4.1.1. Testes

O teste de um componente consiste em checar o valor de um conjunto de seus atributos, sendo que pode haver mais de uma forma de faze-lo. Por exemplo, para testar uma fonte de alimentação que forneça 5V, 9V e 12V, nós precisamos testas essas três tensões da fonte e, para testar a fonte de 9V, por exemplo, talvez possamos usar

um voltímetro ou observar se a lâmpada piloto está acendendo. Assim a descrição do procedimento de teste da fonte poderia ser:

(teste com um voltímetro a fonte de 5V) e
(teste com um voltímetro a fonte de 12V) e
((teste com um voltímetro a fonte de 9V) ou
(teste se a lâmpada está acesa))

No caso de um modelo de testes por decomposição funcional, porém, a situação é diferente por que cada componente do modelo funcional executa apenas uma função, assim, testá-lo significa simplesmente observar se ele está executando essa função. Para esclarecer isto, tomemos por exemplo a fonte que fornece três tensões de saída cuja decomposição funcional é apresentada a seguir:

A alimentação é um módulo composto dos sub-módulos alimentação de 5V, alimentação de 9V e alimentação de 12V. A função do módulo alimentação é fornecer energia para o restante do equipamento. Para que essa função possa ser executada, necessitamos de sub-funções que forneçam tensões para a operação de cada parte do equipamento. Testar a alimentação, porém, significa apenas checar se as tensões de 5, 9 e 12V estão sendo fornecidas pelo módulo alimentação de tal forma que o equipamento como um todo possa operar, sem no entanto testar

isoladamente estas tensões. Se isto não puder ser feito então não faz sentido conceber o conceito de um módulo global de alimentação para fins de diagnóstico, já que a eliminação desse conceito em nada alteraria a diagnose. Para testar o módulo alimentação podemos, por exemplo, observar se algum módulo do equipamento que só possa operar com essas três tensões está de fato operando. Por exemplo, se um televisor está apresentando imagem e som, como essas funções só podem ser executadas com a ocorrência simultânea de todas as tensões fornecidas pela fonte, então, provavelmente, a fonte estará operando corretamente.

Nenhum teste em particular, porém, poderá garantir com completa certeza que a fonte está funcionando. Necessitamos assim de um conjunto de testes para aumentar o grau de confiança no diagnóstico. Portanto podemos modelar a descrição de um teste como sendo um conjunto de observáveis que devemos testar para confirmar a correta operação do módulo.

4.4.1.1.1. Descrição de um teste

Um teste é um ajuste seguido de uma medida. Para que um teste possa ser realizado é necessário que o equipamento esteja num certo estado. Antes de aplicar um teste devemos portanto verificar se o equipamento está de fato nesse estado para, caso não esteja, alterarmos o estado do dispositivo para o estado apropriado. Isso é um ajuste. Por exemplo, desconectar um componente, levar o equipamento para o laboratório, abri-lo, etc. A medida consiste em determinar o valor do atributo do módulo que está sendo testado.

Para descrevermos um procedimento de teste precisamos dos seguintes elementos:

- Procedimento de teste: é a orientação de como proceder para realizar o teste: os ajustes que devem ser feitos no equipamento e como proceder para obter o resultado do teste.

- Custo do teste: pode ser de diversas naturezas tais como a sua dificuldade (a destreza necessária para realizá-lo), a sua demora, o custo financeiro, a necessidade de equipamentos, etc. A avaliação do custo de um teste depende do estado atual do equipamento (devido aos custos dos ajustes necessários), de uma descrição do usuário do sistema de diagnose (para determinar sua habilidade e as ferramentas de que dispõe), da urgência do conserto e quanto se está disposto a pagar em troca dessa urgência. Essas informações são fundamentais para se determinar o procedimento de reparo que se deve adotar.

- Escopo é o nome do atributo de um componente que está sendo testado. Cada componente tem atributos cujos valores são o valor normal desse atributo e os valores anormais que são sintomas. A montagem de um teste é feita a partir desses valores do escopo do teste e o seu resultado se reflete na definição do valor do atributo do componente,

4.4.1.2. Reparos

A descrição de como reparar um componente de um equipamento envolve uma descrição do reparo (o que deve ser reparado) e os procedimentos associados (como consertar). Pode existir mais de uma

forma de reparar um componente. Por exemplo, pode-se consertar o pára-choques amassado de um carro trocando-o ou desamassando-o. Assim sendo a descrição do procedimento de reparo envolve a descrição dos atributos do componente a serem reparados e das alternativas possíveis a cada reparo.

4.4.1.2.1. Procedimento de reparo.

A descrição do processo de reparo que consiste numa seqüência de atividades a serem executadas, que por exemplo pode assumir a seguinte forma:

reparo do teclado:

custo: alto;

procedimento: substitua o teclado por um outro.

reparo do teclado:

custo: baixo;

procedimento:
abra o teclado com a ajuda de uma chave de fenda E
localize a tecla que está com mal contato E
execute o reparo da tecla E
feche o teclado.

reparo da tecla:

custo: médio;

procedimento: troque a borracha condutora.

reparo da tecla:

custo: baixo;

procedimento:
retire a borracha condutora da tecla E
raspe a borracha condutora com uma faca E
verifique com um ohmímetro se ela está conduzindo E
recoloque a borracha no lugar.

Para descrevermos um reparo precisamos assim dos seguintes elementos:

- Descrição do reparo;

- Custo do reparo;

O procedimento de reparo fornece todas a possíveis seqüências para recuperar o funcionamento do equipamento. O procedimento em particular que será adotado dependerá dos custos de reparo e da habilidade do operador.

4.4.1.3. Aconselhamentos

Uma vez tendo descoberto a explicação de uma falha, se ela for de origem externa, a ação a ser tomada pelo sistema será aconselhar o usuário acerca da correta manipulação do equipamento para que futuramente a falha não se repita. Algumas das possíveis falhas de origem externa são: ajuste incorreto do equipamento, manipulação incorreta (por exemplo, esquecer de colocar combustível no tanque ou colocar gasolina ao invés de álcool) ou eventos de natureza não controlável (por exemplo, falta de energia da rede). O aconselhamento é feito em função da causa observada e do estado do equipamento por meio de um conjunto de regras simples do tipo:

- Se explicação é: "não colocou combustível" então
  aconselhe: "Nunca esqueça de por combustível no tanque".

4.4.1.4. Causas e explicações

Enquanto as mensagens anteriores tinham como objetivo induzir um certo comportamento no usuário do sistema, as mensagens relativas a causas e explicações têm função meramente de informar, servem apenas para que o usuário saiba o que ocorreu com o equipamento e o que levou a tal evento.

4.4.2. **Interações de iniciativa do usuário**

Durante a diagnose o usuário poderá querer saber como o sistema chegou a certa conclusão ou porque certa ação foi prescrita. Para tanto ele poderá fazer perguntas simples do tipo: "Qual é o defeito" ou "Qual é o remédio", poderá questionar sobre a motivação de certo teste, da forma como certa conclusão foi atingida ou mesmo poderá fazer questões mais complexas como: "Porque a falha ocorreu?" ou "Irá o remédio Y corrigir a falha?" ou "Posso testar W sem afetar Z?".

O tratamento dessas questões deve ser feito no contexto do modelo de diagnose adotado, pois ele representa a estratégia usada por um especialista humano e, assim, a resposta que deve ser dada às perguntas do usuário devem refletir a maneira como o especialista as responderia. Isso envolve a modelagem do meta-conhecimento do especialista ou da compreensão que o Engenheiro de Conhecimento tem do modelo. Tal assunto não será analisado em profundidade aqui.

4.5. REPRESENTAÇÃO DO MODELO NA KCML

Um Modelo de Interpretação, na concepção da metodologia KADS [BREUKER 87b], é um molde, um gabarito, que permite que se construa um sistema especialista a partir de um refinamento "top-down" deste modelo. Esse "molde" se compõe de quatro camadas, que correspondem a níveis crescentes de abstração, que são: camada de domínio, camada de inferência, camada de tarefa e camada de estratégia, que são descrevidas por meio da KCML.

4.5.1. Camada do domínio.

Um Modelo de Interpretação não possui uma camada de domínio completa. A KCML não propõe uma forma rígida para representar essa camada. Para suprir essa deficiência, definimos aqui uma linguagem para que o usuário possa descrever os elementos dessa camada. Essa linguagem se compõe das seguintes primitivas:

4.5.1.1. Conceitos:

A descrição de um conceito depende de seu tipo.

a) Os módulos funcionais que compõem o equipamento com seus atributos, que são caracterizados por:

- nome;
- probabilidade a priori de falha de cada função;
- descrição textual da função para fins de geração automática de teste;

- atributos específicos com seus possíveis valores, incluindo as características testáveis do módulo funcional;

b) Os testes existentes;

- nome;
- Procedimento de teste;
- escopo (módulo funcional e atributo);
- valor esperado para o teste: valor esperado do escopo;
- custo.

c) Os reparos.

- nome;
- procedimento;
- custo;

d) A explicação de cada possível causa de falha com sua caracterização, os reparos e aconselhamentos associados.

- nome;
- condições de validação (regras de decisão);
- aconselhamento (texto).

4.5.1.2. Relações.

As relações que empregamos são: parte-de, explica, testa e repara.

a) Parte-de. Exemplo: "A" parte-de "B", onde A e B são módulos funcionais, significa que "A" é sub-parte de "B" ou seja que "B" é composto de, ao menos, "A".

b) Explica. Exemplo: "A" explica "B", onde "A" é uma explicação e "B" é uma causa (um valor de uma característica de um conceito).

c) Testa. Exemplo: "A" testa "B", onde "A" é um teste e "B" é um módulo funcional.

d) Repara. Exemplo: "A" repara "B", onde "A" é um reparo e "B" é uma explicação.

Uma relação é definida a partir de um conjunto de pares ordenados de conceitos e, possivelmente, uma definição lógica, tal como:

SUB-PARTE(A,B) => PARTE-DE(A,B) ou

SUB-PARTE(A,B) => PARTE-DE(C,B) e SUB-PARTE(A,C),

que significa que A é sub-parte de B se A é parte de B ou se existe um C tal que C é parte de B e A é sub-parte de C.

Antes que o modelo fornecido pelo usuário possa ser compilado, é necessário verificar a consistência do mesmo, através desse tipo de conhecimento profundo do domínio. Por exemplo, definições como a que se segue:

SUB-PARTE(A,B) => não existe SUB-PARTE(B,A).

podem ser usadas para verificar a inconsistência da seguinte parte de um modelo, como por exemplo:

PARTE-DE(A,B).

PARTE-DE(B,C).

PARTE-DE(C,A).

4.5.1.3. Estruturas.

As estruturas são definidas como um conjunto de conceitos que se relacionam por meio de um certo conjunto de relações.

a) Tabela de falhas. Associa módulos e possíveis sintomas característicos.

b) Estrutura funcional. Descreve a decomposição funcional do equipamento em módulos por meio de relações "parte-de".

4.5.2. **Camada de inferências.**

A camada de inferência descreve as inferências que podem ser feitas sobre a camada de domínio. A camada de inferências se subdivide em Metaclasses, que indicam o papel que os conceitos podem ter durante a diagnose, e em Fontes de Conhecimento, que indicam as inferências que podem ser feitas com base nas relações da camada de domínio.

4.5.2.1. Metaclasses.

Temos as seguintes Metaclasses associadas às Fontes de Conhecimento:

- descrição do caso
- discrepância
- hipótese
- hipótese ordenada
- comportamento observado
- diagnóstico
- explicação
- reparo

4.5.2.2. Fontes de Conhecimento.

As Fontes de Conhecimento representam passos básicos de inferência executados pelo especialista. Convém salientar que, embora tais inferências sejam imediatas para o especialista, a sua representação na forma de um programa de computador ou outra qualquer, poderá não ser trivial. A origem das 7 Fontes de Conhecimentos que passaremos agora a descrever é esclarecida no item 5.3.

Embora o modelo de diagnose que analisamos empregue um tratamento probabilístico, a nossa implementação da ferramenta de Aquisição de Conhecimento, não faz uso desse recurso. Assim, a descrição das Fontes de Conhecimento faz menção à questão do tratamento das probabilidades apenas a título de ilustração, para exemplificar como isso pode ser feito, e não por que tal recurso seja efetivamente utilizado.

4.5.2.2.1. Identifica.

Mapeia o equipamento defeituoso, caracterizado por uma descrição do caso, em uma das possíveis classes de equipamentos, representada por um modelo do dispositivo. Para tanto a Fonte de Conhecimento apresenta ao usuário uma lista dos equipamentos e/ou módulos do equipamento que é capaz de diagnosticar, podendo também apresentar, caso seja necessário, esclarecimentos sobre a aplicabilidade de cada um dos modelos para casos concretos. Por exemplo, ela pode apresentar o seguinte menu para o usuário:

"O equipamento que está com defeito é:

a) um caminhão;

b) um carro;

c) uma moto."

Vale notar que num nível de sofisticação maior, essa Fonte de Conhecimento poderia dispor de diversos tipos de conhecimentos sobre um mesmo domínio, porém analisado sobre diferentes prismas, como na abordagem de Davis [DAVIS 84]. Nesse caso podemos ter, para um equipamento eletrônico, as seguintes opções: modelo do dispositivo para falhas simples, modelo para curto-circuitos, modelo para componentes operando de maneira inesperada (devido, por exemplo, como cita Davis, à perda do contato da alimentação de um circuito integrado), modelo para múltiplas falhas, modelo para erros de montagem do equipamento e modelo para erro de projeto. Em tal situação, essa Fonte de Conhecimento deve permitir que se passe de um modelo para outro mantendo as informações já levantadas pelos testes.

4.5.2.2.2. Decompõe.

Guiada pela discrepância, percorre o modelo do equipamento, visando gerar novas hipóteses ou um diagnóstico. As hipóteses são geradas a partir da análise do modelo de falhas do equipamento. "Decompõe" pede que o usuário escolha dentre uma lista de sintomas aqueles que está observando no equipamento e então, através de um raciocínio abdutivo, gera um conjunto de falhas que podem explicar tais sintomas (o diferencial).

O objetivo dessa Fonte de Conhecimento é fazer um corte nas hipóteses que são analisadas conjuntamente pelo sistema de diagnose. O número dessas hipóteses pode ser fixado em, por exemplo, 7 (o

número mágico da psicologia) que é o maior número de conceitos com que o especialista pode lidar simultaneamente na sua memória de curto prazo.

Essa Fonte de Conhecimento parte de uma dada discrepância. Por exemplo, se a discrepância for a função 'F', decompõe coleta sintomas que permitam distinguir as hipótese de falha de cada uma das sub-funções imediatas de 'F'. O objetivo é apresentar ao usuário um conjunto de sintomas facilmente identificáveis, sem a necessidade de procedimentos complexos de teste, para que esse indique quais estão ocorrendo. A partir dessa informação, consultando o modelo de falhas, gerando-se as hipóteses mais prováveis, estejam elas associadas a sub-funções imediatas de 'F' ou não.

Suponhamos que a estrutura de um equipamento seja tal que "A", "B" e "C" sejam filhos de um mesmo módulo e "D" seja filho de "A" (figura 4.3).

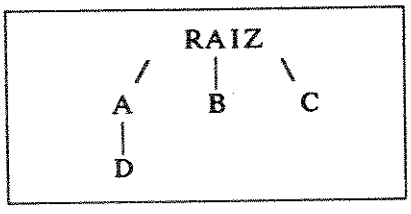


FIGURA 4.3 - Exemplo da decomposição de um equipamento.

Tomando como exemplo a situação representada na tabela 4.1, o melhor conjunto de sintomas para isolar esses módulos são: S2 ou S5, S3 e S4 (pois S1 é inconcluso e S2 e S5 são equivalentes). Assim a seguinte mensagem seria apresentada ao usuário:

"Identifique dentre os seguintes sintomas aqueles que você está observando no equipamento:

a) A lâmpada está acendendo intermitentemente.

b) O driver está fazendo um barulho estranho.

c) O driver não está fazendo seu ruido normal."

Suponhamos que o usuário selecione as opções "a" e "b", nesse caso "decompõe" irá gerar as hipóteses: o "defeito está em C" e o "defeito está em D" (o conjunto dessas duas hipóteses, na linguagem da diagnose médica, é chamado de diferencial). Observe que "D" é uma hipótese que inicialmente não havia sido levantada, mas que surgiu em função dos sintomas observados. Uma situação como essa permite que o sistema de diagnose "pule" diretamente para o defeito sem ter que passar por todos os estágios de uma decomposição funcional, tal como o faz, freqüentemente, um especialista.

Além de gerar as hipóteses, "decompõe" atualiza as probabilidades de falhas dos módulos de maneira condizente com as informações fornecidas pelo usuário.

É importante salientar que a observação de um sintoma é conclusiva para afastar uma hipótese de falha que não possa gerar esse sintoma, porém a não observação de um sintoma que seria esperado a partir da falha de certo módulo não é suficiente para excluir essa hipótese de falha. Isto ocorre porque para saber se um módulo está em perfeito funcionamento é preciso verificar se nenhum de seus modos de falha está ocorrendo.

4.5.2.2.3. Ordena.

Converte as hipóteses em hipóteses ordenadas, de acordo com critérios heurísticos. Para tanto "ordena" analisa o conjunto de

hipóteses a partir da aplicação da função de avaliação conforme descrito no item "Seleção do melhor teste".

4.5.2.2.4. Seleciona.

Essa Fonte de Conhecimento coleta evidências na forma de testes para a hipótese de causa de falha definida pelo primeiro elemento de hipóteses ordenadas. "Seleciona" é responsável pela aplicação dos testes de menor custo para essa hipótese que é a hipótese corrente de trabalho. Um teste é selecionado e apresentado ao usuário a partir da definição de seu escopo (o atributo do conceito que o teste checa). As opções de resultado do teste que são apresentadas ao usuário representam os valores possíveis desse atributo.

Um teste poderá ser montado automaticamente, caso ele não tenha sido especificado. Para tanto emprega-se a descrição da função fornecida pelo usuário. Por exemplo, suponhamos que a descrição da função "alimentação" seja "fornece energia para que os demais módulos do equipamento operem adequadamente" e que o componente que a executa seja "fonte de alimentação", nesse caso o teste automaticamente gerado seria:

"Verifique se a seguinte afirmação é verdadeira:

A fonte de alimentação fornece energia para que os demais módulos operem adequadamente."

"Seleciona" atualiza então as probabilidades e o valor do atributo testado para que passem a refletir o resultado do teste.

4.5.2.2.5. Calcula.

A partir do comportamento observado, verifica se é possível

identificar qual a função discrepante. Uma função é considerada discrepante se sua probabilidade de falha estiver acima de um certo limiar fixado, a hipótese de falha do módulo é eliminada das hipóteses ordenadas se a probabilidade estiver abaixo de outro limite. A hipótese é mantida entre as hipóteses ordenadas se sua probabilidade estiver na região intermediária e por fim, a hipótese é definitivamente excluída, se sua probabilidade se tornar nula.

4.5.2.2.6. Explica.

A partir do diagnóstico obtido e, possivelmente, de novos testes, gera uma explicação para a causa da falha. Essa Fonte de Conhecimento opera sobre a caracterização dos modos de falha do equipamento conforme descrito no item 4.1.3. "Explica" determina quais explicações são compatíveis com as informações já levantadas pelo sistema. Caso haja mais de uma hipótese viável, seleciona-se as características mais convenientes para se determinar univocamente qual é a real origem da falha. Para tanto essa Fonte de Conhecimento poderá lançar mão de testes adicionais.

Para obter a explicação de uma falha utilizam-se regras. O especialista caracteriza os principais tipos de falhas e suas explicações por meio delas. Por exemplo, podemos ter uma meta-regra afirmando que se não existe uma regra que explique a falha e o usuário é inexperiente, então a explicação mais provável é o uso inadequado do equipamento. Por exemplo:

explicação: falha do operador.
mensagem: "Tenha mais cuidado ao manipular o equipamento".

características:

- mais de um módulo defeituoso.
- nenhuma outra explicação foi encontrada.
- usuário inexperiente.

explicação: falha na fonte.

características:

- fonte defeituosa.
- fusível queimado.
- fusível trocado e novamente queimado.

Essa Fonte de Conhecimento pode ser empregada para superar uma limitação da diagnose por modelo funcional, que é o fato dela não poder ir até o nível dos componentes individuais que não executam nenhuma função por si só (um transistor, por exemplo, só executa uma função quando dentro de um circuito de polarização, assim "transistor" não está associado diretamente a nenhum módulo e, portanto, não existe nesse ponto de vista). Para tanto podemos usar regras associando falhas funcionais a componentes falhos, da mesma forma como isto é feito numa diagnose baseada em sintomas, sem a desvantagem de se ter que empregar um número enorme de regras uma vez que a falha já foi restringido ao nível de um sub-módulo.

4.5.2.2.7. Prescreve.

Associa um reparo à causa da falha se ela for de origem interna e determina qual o reparo que apresenta o menor custo total. Tal

procedimento é então apresentado ao usuário. Caso o usuário não consiga executá-lo, um novo procedimento será apresentado.

Se, para certo componente, não for especificado um procedimento de reparo específico uma mensagem do seguinte tipo será elaborada:

"Substitua o componente <NOME> de código <PART-NO>".

4.5.2.3. A Estrutura de Inferência

A Estrutura de Inferência do modelo de diagnose é apresentada na figura 4.4.

FIGURA 4.4 - Estrutura de Inferência do modelo de diagnose.

4.5.3. **Camada de tarefa.**

Essa camada indica a forma como as Fontes de Conhecimento são

combinadas para atingir um certo objetivo, no nosso caso, a diagnose e o reparo do equipamento.

4.5.3.1. A tarefa de diagnose.

A metodologia KADS apresenta um modelo de interpretação para a diagnose baseada em sintomas, para a diagnose por localização e a baseada em busca causal, porém não há um modelo para a diagnose baseada em modelo de falhas. Sendo assim desenvolvemos nosso próprio modelo de interpretação para essa tarefa a partir do ciclo de raciocínio na averiguação do diagnóstico [PEARCE 88], apresentado a seguir:

1. Formação do contexto inicial;
2. focaliza na explicação parcial mais promissora sobre as observações anormais;
3. explora a explicação mais promissora;
4. incorpora observações novas nas soluções parciais;
5. verifica se foi atingido uma explicação satisfatória, se não volte para 2;
6. compila e discute as recomendações.

O diagnóstico é obtido abduzindo hipóteses, deduzindo suas expectativas e formulando e executando experimentos para testar tais expectativas. Esse ciclo é repetido até que uma explicação para as observações anormais possa ser induzido.

Inicialmente temos um processo de inferência abdutiva na qual algumas hipóteses são geradas. Isto pode ser feito num processo

associado a um contexto ou não. A inferência abdutiva não-contextual geralmente ocorre no início da diagnose. A hipótese é uma hipótese geral que restringe a quantidade de causas possíveis sobre o defeito do dispositivo para as causas mais prováveis. Uma indução contextual ocorre quando uma hipótese é gerada no contexto definido por outra hipótese, podendo ocorrer da nova hipótese ser um refinamento ou generalização da hipótese anterior ou uma antítese ou complemento da mesma.

Uma vez levantada as hipóteses segue-se um processo de inferências dedutivas no qual se determina quais observações derivam da hipótese. Assim, experimentos para testar as hipóteses podem ser formulados. Por fim temos um processo de inferências indutivas no qual se decide se as hipóteses podem ser aceitas ou rejeitadas, ou se deve-se testar novamente, dependendo de quão próximas as observações estão das expectativas das hipóteses. Esse passo lida com o fim do processo de diagnose pois ele decide se uma explicação satisfatória sobre o defeito do dispositivo foi obtida.

Este ciclo foi adaptado para a metodologia KADS [BREUKER 87b] e ligeiramente modificado:

1. seleciona o contexto apropriado;
2. focaliza nas hipóteses parciais mais promissoras sobre as observações anormais, se há apenas uma hipótese viável então vá para 6;
3. ordena as hipóteses mais promissoras;
4. incorpora observações novas nas hipóteses parciais;

5. verifica se foi obtida uma explicação satisfatória, se não volte para 2 ou 3;
6. obtenha uma explicação e um reparo ou conselho apropriado.

Esse procedimento de reparo foi convertido em duas tarefas, os itens de 1 a 5 formam a tarefa de diagnose apresentada a seguir e o item 6 representa a tarefa de reparo.

ache(diagnóstico)
identifica(descrição do caso; modelo do sistema)
REPITA
  decompõe(modelo do sistema, discrepância; hipótese, diagnóstico)
  ordena(hipóteses; hipóteses ordenadas, diagnóstico)
  ENQUANTO o número de hipóteses de hipóteses ordenadas >= 1
        E nenhuma causa foi encontrada
    seleciona(hipóteses ordenadas, observáveis; comportamento)
    calcula(comportamento observado; discrepância)
    ordena(hipóteses; hipóteses ordenadas, diagnóstico)

  ATÉ hipóteses ordenadas estar vazia

4.5.3.2. A tarefa de reparo.

O KADS não fornece um modelo de interpretação para o reparo, porém, considerando as idéias de Steels [STEELS 88], propomos a seguinte estrutura de tarefa para o reparo:

faça(reparo)
ache(diagnóstico)
explica(diagnóstico; explicação)
se explicação é um componente falho
        então **prescreve**(explicação; reparo)

Inicialmente a causa do defeito é localizada através de um processo de diagnose. A partir dessa causa uma explicação é obtida e, caso se trate de um componente defeituoso, um procedimento de reparo para o componente é prescrito.

4.5.4. **A camada de estratégia.**

Essa é uma camada que nós não implementaremos. Dentro de nossa concepção a camada de estratégia do KADS é responsável pelo ajuste da "máquina" definida pelos três níveis inferiores. No nosso caso isso consistiria no ajuste fino da função de avaliação a partir da análise da situação em que a diagnose ocorre, das prioridades feitas pelo usuário, da avaliação do usuário quanto à sua habilidade e outros fatores eventualmente relevantes.

4.6. **CONCLUSÃO**

O modelo aqui descrito assume certas características do domínio. Por exemplo, não se trabalha com descrições do comportamento do equipamento, como a lei de Ohm. Opera-se com heurísticas e dados

como taxas de falha, junto com um modelo funcional do equipamento, de maneira a permitir a localização da falha no equipamento. O ponto fundamental é a representação do equipamento como uma hierarquia funcional de módulos. Essa é a informação básica requerida, outras informações são opcionais e podem ser adicionadas incrementalmente.

O modelo proposto para a diagnose possui grande maleabilidade, podendo ser adaptado para representar diversos procedimentos diferentes de diagnose. Essa maleabilidade se origina do uso de uma função heurística poderosa que guia o processo de diagnose. A modelagem dessa função pode ser feito observando a forma como o especialista procede na prática ao escolher os testes e hipóteses e realizando uma análise estatística das escolhas feitas pelo especialista.

Ainda mais interessante é que podemos realizar ajustes finos nessa função, antes e durante a diagnose, para adaptar o sistema de diagnose a situações particulares. Com isso o sistema ganharia um planejamento estratégico que corresponderia ao conhecimento de estratégia que falta ao modelo.

Utilizando-se de um algoritmo de aprendizado seria possível tornar o sistema capaz de aperfeiçoar seu desempenho a partir da experiência. Isso pode ser feito a partir da atualização das probabilidades de falha assumidas a priori para os módulos. Para tanto, toda vez que se localiza a falha em um módulo sua probabilidade de falha deve ser aumentada de um fator inversamente proporcional ao número de diagnósticos já realizados pelo sistema.

CAPÍTULO V

A FERRAMENTA DE AQUISIÇÃO DE CONHECIMENTO

5.1 INTRODUÇÃO

Vimos no capítulo 2 que existem muitos sistemas capazes de interagir de uma forma amigável com um especialista de maneira a obter o seu conhecimento e com isso construir diretamente um SE. Essas ferramentas possuem um conhecimento prévio da área em que atuam. No capítulo 2, por exemplo, é apresentado um trecho do dialogo entre o ROGET e um especialista aonde o especialista é inquerido acerca da finalidade de certo teste e dos seus possíveis resultados. Vemos, nesse exemplo, que ROGET sabe de antemão que para confirmar a ocorrência de qualquer doença é necessário realizar certos testes, que esses testes visam detectar alguma coisa e que existe um certo conjunto de resultados possíveis para cada teste.

Tal conhecimento a priori que a ferramenta possui é o que chamamos de modelo do domínio. Esse modelo é um conhecimento genérico e parcial sobre a atividade a que se destina a ferramenta e que é empregado para guiar a extração do conhecimento mais específico que é necessário para se construir o SE, pois o modelo do domínio não é completo, ROGET, por exemplo, não sabe previamente quais são os testes existentes, qual a finalidade de cada um, nem seus possíveis resultados.

Existem também ferramentas de Aquisição de Conhecimento que não possuem um modelo, são os editores inteligentes de conhecimento (como o CYC ou o KPT). Eles, porém, são apenas ferramentas de apoio ao Engenheiro de Conhecimento e não podem ser utilizadas diretamente por um especialista não familiarizado com os conceitos envolvidos na Inteligência Artificial. Em compensação, os editores potentes de

conhecimento são de uso geral, não são destinadas a um domínio específico.

A ferramenta que aqui descreveremos visa superar as limitações dessas duas abordagens. Para tanto conceituamos uma ferramenta que possui um modelo do domínio em que atua e assim é capaz de interagir diretamente com o especialista na construção de um SE. O modelo empregado por tal ferramenta, porém, não é implícito e inalterável, como nas ferramentas usuais de Aquisição de Conhecimento. A idéia é criar uma ferramenta que possa operar com diversos modelos. Assim, conforme o domínio a que se destine a ferramenta, um modelo adequado poderá ser construído por um Engenheiro de Conhecimento e inserido na mesma, que poderá então ser utilizada pelo próprio Engenheiro de Conhecimento ou diretamente por um especialista no domínio, para construir um SE.

Existem diversas vantagens nessa abordagem. Em primeiro lugar caso se deseje construir mais de um SE, se eles se destinarem ao mesmo tipo de atividade, será possível simplificar enormemente o trabalho pela adoção de um modelo único para todos eles. Caso se adotasse a abordagem convencional, com a construção de um SE por vez, a complexidade da tarefa seria proporcional ao número de sistemas desejados pois seria necessário localizar um especialista para cada caso, convencê-lo a participar de uma série cansativa de entrevistas, converter o conhecimento para uma representação adequada, para então codificar o SE. Pouco se poderá aproveitar do conhecimento de um SE que possa ser aproveitado em outro sistema em virtude das grandes diferenças das abordagens heurísticas entre especialistas distintos e da forma como eles organizam seu conhecimento. Embora o conhecimento

teórico que embasa os especialistas possa ser o mesmo, como esse conhecimento é explicitado "compilado" em regras práticas, resulta uma aparente total discrepância entre os Sistemas Especialistas.

Outro problema da abordagem convencional é a dificuldade de manutenção e adaptação dos Sistemas Especialistas obtidos dessa forma. No caso, por exemplo, de um SE para diagnose, alterações no SE serão constantemente necessários pois os equipamentos estão frequentemente sendo adaptados em função de aperfeiçoamentos e reconfigurações. Isso poderá fazer com que todo o desenvolvimento do SE tenha de ser refeito para criar um novo sistema. Além disso, devido a essas constantes alterações, nem sempre haverá um especialista pois é necessário tempo para uma pessoa com certo conhecimento teórico se tornar um especialista em uma atividade.

Por fim, embora as ferramentas de Aquisição de Conhecimento usuais sejam capazes de adquirir o conhecimento de que precisam, tal conhecimento finda por ficar "escondido" nas centenas ou milhares de regras que compõem uma base de conhecimento. O conhecimento adquirido e armazenado pelo sistema tenderá a se tornar hermético, inacessível para aqueles que desejem se instruir ou aperfeiçoar sua capacidade profissional além de dificultar a manutenção da base de conhecimento e tornar difícil a compreensão do comportamento de um SE assim construído, tornando o sistema imprevisível.

A ferramenta aqui apresentada procura superar tais dificuldades através do uso de um modelo explícito da atividade a que ela se destina. O modelo pode ser reutilizado, simplificando o desenvolvimento de novos SEs para o mesmo domínio. Caso não haja um especialista disponível, esse modelo auxiliará a construção do SE

através do fornecimento de sua estrutura básica. O conhecimento adquirido pela ferramenta é organizado de uma forma coerente em função da existência de um estrutura básica que é o conhecimento teórico sobre a diagnose ou qualquer outra atividade a que se destine a ferramenta. Assim evita-se que a base de conhecimento se torne hermética, de difícil compreensão.

Em geral não é uma tarefa simples construir um modelo de um domínio em vista da pouca clareza sobre o deve conter esse modelo ou como ele deve ser construído. Para facilitar a tarefa de conceber um modelo que possa ser utilizado na ferramenta recorremos à metodologia KADS que não só prescreve métodos para construir tal modelo, como também fornece uma linguagem para descrevê-lo e nos dá uma biblioteca de modelos básicos com os quais podemos criar novos modelos.

Nosso objetivo, entretanto, não é criar uma ferramenta que dê suporte à metodologia KADS, mesmo por que tal ferramenta já existe, é o SHELLEY - ambiente desenvolvida no projeto KADS que, ao contrário da ferramenta que aqui descrevemos, destina-se a pessoas familiarizadas com a metodologia KADS e com a Inteligência Artificial. O SHELLEY é um editor inteligente de conhecimento e um gerenciador de projeto. Nossa ferramenta destina-se a especialistas de qualquer área, mesmos os não familiarizados com a Inteligência Artificial.

A ferramenta que passaremos a descrever, a qual chamamos CAKE (Computer Aided Knowledge Engineering) tem como objetivo permitir a construção de Sistemas Especialistas voltados para o reparo de equipamentos, de uma maneira amigável, por qualquer especialista.

Essa ferramenta possui um modelo explícito do domínio a priori descrito através da KCML (KADS Conceptual Modeling Language).

Sendo assim o modelo que essa ferramenta emprega pode ser modificado, por um Engenheiro de Conhecimento, de maneira a adaptá-lo a particularidades do domínio de diagnose que se deseja tratar. Deve-se notar que para empregar a ferramenta fora desse domínio, alterações consideráveis da ferramenta poderão ter que ser feitas, tornando necessário um trabalho adicional do Engenheiro de Conhecimento, ou seja, ele deverá não só construir um novo modelo como poderá ter que adaptar a ferramenta de Aquisição de Conhecimento. Para que a ferramenta de Aquisição de Conhecimento pudesse ser totalmente geral sem que adaptações fossem necessárias, seria preciso fornecer uma gama suficientemente grande de primitivas computacionais voltadas para a implementação de cada uma das inferências básicas associadas à taxonomia de Fontes de Conhecimento do KADS. Como, porém, esse não é o nosso objetivo, nos restringiremos apenas ao domínio da diagnose sistemática. A expansão do domínio de aplicação da ferramenta, porém, não apresenta grandes dificuldades conceituais.

A implementação dos conceitos que previamente exploramos, tanto no que tange à construção de uma ferramenta de Aquisição de Conhecimento com o uso do modelo de diagnose que descrevemos em toda sua extensão, é uma tarefa de grande porte. Como nosso objetivo é tão somente demonstrar a viabilidade dos conceitos aqui expostos, certas simplificações foram feitas na interação entre o usuário e a ferramenta e no próprio modelo de diagnose.

A interface com o especialista não é feita de uma maneira que a ferramenta possa ser utilizada por um especialista totalmente não familiarizado com os conceitos envolvidos. Para que isso fosse necessário a ferramenta deveria obter as informações necessárias por meio de perguntas como as do ROGET ou de outras técnicas de extração de conhecimento. Para tanto, a ferramenta deveria permitir que o Engenheiro de Conhecimento especificasse um modelo de cooperação no qual descreveria detalhadamente as interações envolvidas. Simplificando, adotamos a opção de obter as informações por meio de um editor restringido de conhecimento.

Para obter uma ferramenta capaz de interagir diretamente com um especialista leigo restaria ainda desenvolver um módulo de interface suficientemente maleável para ser utilizado na aquisição de todo tipo de conhecimento. A concepção de um tal módulo não é de maneira alguma trivial, pois envolveria a definição de um editor de Modelo de Cooperação, para permitir que o Engenheiro de Conhecimento definisse a forma como a ferramenta deve interagir com o especialista durante a Aquisição de Conhecimento. Seria também necessário fornecer um conjunto suficientemente poderoso de ferramentas de Aquisição de Conhecimento capazes de trabalhar conjuntamente (tarefa essa por si só de grande dificuldade). Optamos, assim, por suprimir tal módulo, fornecendo apenas um editor de conhecimento com o qual é possível ao usuário definir o conhecimento do domínio por meio de uma série de prompts. O usuário visado pela ferramenta que implementamos deve estar previamente familiarizado com os conceitos que devem ser fornecidos à ferramenta.

5.2 REQUISITOS

A ferramenta de Aquisição de Conhecimento que desejamos construir deve atender às seguintes exigências:

. Interagir com um especialista em diagnose, gráfica e textualmente, para realizar o processo de Aquisição de Conhecimento na diagnose de equipamentos;

. gerenciar a informação que é usada no processo de Aquisição de Conhecimento, permitindo que o especialista examine e modifique a base de conhecimento de uma forma simples;

. fornecer processos de solução de problemas predefinidos na forma de estratégias gerais utilizáveis em todas as aplicações similares no domínio da diagnose;

. verificar, ainda que parcialmente, a consistência do modelo fornecido pelo usuário;

. ser capaz de construir Sistemas Especialistas que realizem diagnose automaticamente;

5.3 ESTRUTURA GERAL

O objetivo da ferramenta CAKE é mapear o conhecimento de um especialista para um SE. Para tanto, a ferramenta emprega um modelo intermediário entre o conhecimento do especialista e o conhecimento representado em um SE através da KCML, a linguagem de modelagem conceitual do KADS. O modelo contém uma descrição do conhecimento

estático (camada de domínio), do conhecimento sobre as inferências que podem ser realizadas (camada de inferências), e um conhecimento sobre o processo geral de realização da tarefa em questão (camada de tarefa). Não há uma descrição do conhecimento estratégico. No diagrama a seguir representamos esta abordagem.

CONHECIMENTO ESPECIALIZADO
↓
MODELADOR
↓
MODELO DO CONHECIMENTO ESPECIALIZADO
↓
COMPILADOR
↓
SISTEMA PARA DIAGNOSE

O modelador é um editor amigável associado a um sistema de verificação de consistência, um browser de conhecimento (visualizador) e um gerenciador da base de conhecimento. O compilador é um módulo que converte a representação KCML para uma linguagem computacional. A seguir apresentamos de forma esquemática os principais módulos da ferramenta:

1) EDITOR

- Inserção/remoção de conceitos e relações
- Definição de testes e procedimentos de reparo
- Definição de estruturas
- Definição de Metaclasses
- Edição de Fontes de Conhecimento e da tarefa

a) Editor de conceitos
   PARTES
   TESTES
   REPAROS
   EXPLICAÇÕES

b) Editor de relações

c) Editor de estruturas

d) Editor de Fontes de Conhecimento

e) Editor de Metaclasses

f) Editor da tarefa

2) VERIFICADOR DE CONSISTÊNCIA DO MODELO

a) Completude do modelo funcional

b) Consistência das relações

3) VISUALIZADOR

a) Conceitos

b) Relações

c) Estruturas

d) Metaclasses

e) Fontes de conhecimento

f) Tarefa

4) MÓDULO GRÁFICO

a) Visualização de estruturas

b) Navegação pelo modelo por meio de diagramas

c) Expansão de conceitos

5) GERENCIADOR DA BASE DE CONHECIMENTOS

Armazenamento/recuperação de arquivos

Analisaremos agora cada um desses itens, sem grande preocupação com a forma como eles foram implementados. A descrição detalhada da

implementação de cada item encontra-se no ANEXO. Maiores informações acerca do uso da ferramenta podem ser obtidas em [SCHIAVINI 91].

5.3.1 Editor de conhecimento

O editor de conhecimento permite que o usuário descreva um conhecimento especializado de uma forma amigável e interativa por meio de uma série de prompts que o conduzem na descrição do modelo de diagnose e reparo. O conhecimento relativo à tarefa, às Fontes de Conhecimento e às Metaclasses são fornecidos pelo Engenheiro de Conhecimento, enquanto o conhecimento estático relativo aos conceitos, relações e estruturas, é obtido do especialista.

A descrição da tarefa e das Fontes de Conhecimento possui uma sintaxe específica que pressupõe um conhecimento mais profundo dos princípios que fundamentam a ferramenta, razão pela qual deve ser feita pelo Engenheiro de Conhecimento. A descrição da tarefa é feita de uma forma procedimental usando uma linguagem procedimental específica, enquanto a descrição das Fontes de Conhecimento é feita por meio da linguagem declarativa Prolog (com o auxílio de certas primitivas).

No KADS, uma Fonte de Conhecimento é uma descrição de uma inferência básica feita pela especialista, ela explicita algo que é feito sem no entanto dizer como é feito. Assim, uma Fonte de Conhecimento não é representada por um algoritmo ou algo equivalente, que descreva como realizar certa atividade pois o que importa é tão somente explicitar que a atividade é realizada. O KADS distingue a descrição do conhecimento conceitual da descrição do conhecimento para fins de projeto. Nossa abordagem é conceitualmente semelhante,

porém não idêntica a essa. O uso de uma linguagem declarativa permite, até certo ponto, que o Engenheiro de Conhecimento descreva as Fontes de Conhecimento sem ter de preocupar com a forma como esse problema será computacionalmente resolvido.

É interessante notar que o KADS não é radical sobre esse ponto pois, embora afirme que o Modelo Conceitual é um modelo de um conhecimento especializado sem qualquer vínculo com a implementação, o KADS conceitua que o Modelo Conceitual poderá vir a ser executável, ainda que de forma ineficiente [BREUKER 87b]. Identificamos, assim, na ferramenta, Fontes de Conhecimento com predicados Prolog e, assim como as Fontes de Conhecimento no KADS são descritas por meio de referências a outras Fontes de Conhecimento num processo de detalhamento sucessivo até se chegar a Fontes de Conhecimento que se refiram a inferências elementares que não necessitem de detalhamento, os predicados que associamos às Fontes de Conhecimento são descritos por meio de referências a outros predicados até se chegar a predicados que são considerados primitivos, que realizam inferências básicas.

O editor de Metaclasses permite que definamos quais são as Metaclasses com que trabalhamos. A função das Metaclasses na ferramenta é permitir a troca de informações entre as Fontes de Conhecimento.

Uma vez tendo preenchido a camada de tarefa e de inferências, a ferramenta estará pronta para ser utilizada para a criação de um SE de diagnose para um equipamento em particular. Para tanto inicialmente será necessário definir, por meio do editor de conceitos, quais são as partes componentes do equipamento e, para

cada uma delas, quais são seus atributos, suas características (informação essa que pode ser empregada para determinar o que se pode testar com relação a essas partes). Cada atributo deverá estar associado a dois ou mais valores, por exemplo um deles poderá ser o valor do atributo de um conceito quando o equipamento opera normalmente, e o(s) outro(s) valor(es) o valor em operação em estado defeituoso do equipamento. Por exemplo, para o conceito "vídeo", relativo ao equipamento televisão, algumas dos atributos do vídeo (que são suas características testáveis) são "presença da imagem", "sincronismo", etc., os valores associados à "presença de imagem" podem ser "sim" e "não", sendo "sim" o valor normalmente encontrado e "não" é um valor observado quando o vídeo está com algum problema. A característica "sincronismo" do "vídeo" pode ter os valores "normal", "movimentação vertical da imagem", "movimentação horizontal da imagem", etc.

Além dos conceitos relativos às partes do equipamento, temos também os conceitos de testes e de reparos. O editor de testes define a forma como proceder para testar cada característica de cada conceito (por meio de um texto explicativo que será apresentado ao usuário) e o custo do teste. Todo teste é associado a um conceito, que é o módulo a ser testado, a uma característica, que é a característica do módulo que o teste visa, e a um valor, que o valor esperado para o teste em função da forma como o teste foi feito. O editor de reparos, igualmente, permite que se descreva o procedimento de reparo associado a uma dada explicação para o defeito e o custo desse reparo.

Após, ou durante, o processo de definição dos conceitos devemos definir as relações entre esses conceitos. No caso do modelo simplificado com que trabalhamos, a única relação relevante é a relação "parte-de" que define a forma como as partes do equipamentos se organizam. Existem outras relações que são definidas implicitamente pelo usuário, como por exemplo a relação "testa" que é definida toda vez que se edita um conceito de teste. Por fim temos o editor de estruturas cuja única função é associar um nome a um conjunto de conceitos e relações bem como permitir que se visualize graficamente a estrutura.

5.3.2 **Verificador de consistência**

A grande vantagem de se empregar um editor, ao invés de diálogos dirigidos, para se adquirir o conhecimento é permitir com maior facilidade correções no modelo. Essas correções são necessárias por que certamente serão cometidos erros ao se entrar com o modelo do equipamento e do domínio. Existem diversos tipos de erros possíveis que vão desde erros da forma como o modelo é descrito (erros de sintaxe) até erros no próprio modelo. Alguns desses erros podem ser detectados em função de inconsistências que introduzem devido a informações incorretas ou incompletas. Porém grande parte dos erros são impossíveis de detectar por não haver com o que confrontar as informações fornecidas. Todo verificador de consistência é incompleto, por ser incapaz de detectar falhas semânticas na base de conhecimento. Porém, para facilitar o trabalho do usuário da ferramenta indicando sempre que possível a origem dos erros,

cometidos é muito importante termos a disposição o mais completo verificador de consistência que pudermos construir.

A verificação da consistência de bases de conhecimento convencionais é uma tarefa muito complexa e é um campo de estudos muito ativo, porém ainda mais complexo é a verificação de bases de conhecimento que envolvem um modelo de um conhecimento profundo. Não é nosso intento solucionar tal problema: concebemos tão somente um verificador de consistência simplificado que se destina a ilustrar a necessidade e finalidade desse módulo. Maiores detalhes podem ser obtidos no ANEXO (item A.3.3.).

5.3.3 Visualizador

Através do visualizador (browser) podemos observar todas as partes do modelo de maneira a identificar possíveis erros ou falta de elementos. O visualizador apresenta os dados por meio de tabelas e por meio de gráficos, produzidos pelo módulo gráfico, permitindo assim que o usuário possa checar globalmente as informações que forneceu ao sistema.

5.3.4 Módulo Gráfico

O módulo gráfico apresenta o equipamento na forma de um grafo no qual se pode navegar pelos conceitos que o compõe com o auxílio de um "mouse" ou do teclado, usando as setas de direção. Todo conceito pertencente ao grafo pode ser selecionado, e posteriormente expandido, mostrando os conceitos com que ele se relaciona. Caso se selecione um conceito que seja a raiz do grafo, poderemos observar os atributos e valores associados a esse conceito e, através de uma nova

seleção, as estruturas a que ele pertence. É possível também selecionar para expansão um atributo, um valor, uma estrutura, as partes, os testes e os reparos existentes, bastando para tanto que se navegue pelo gráfico de maneira conveniente.

No grafo que representa a estrutura do equipamento, certas notações são empregadas para indicar casos particulares. Quando um conceito do grafo pode ser expandido, ou seja quando esse conceito se relaciona com outros conceitos que presentemente não estão sendo mostrados por falta de espaço na tela, esse conceito é seguido do símbolo "+". Quando um conceito se relaciona com outro conceito que já se encontra na tela expandido, e que portanto seria redundante expandir novamente, esse conceito é seguido do símbolo "!".

O grafo tem seu tamanho automaticamente ajustado, inclusive o tamanho das letras, de maneira a poder ser integralmente apresentado na tela. Quando porém isto não é possível, pode-se observar as demais partes do grafo fazendo-o se movimentar na tela, com o auxílio das setas do teclado, para cima ou para baixo, ou por meio da seleção de conceitos a expandir por meio da navegação pelo grafo e a seleção do conceito desejado por meio da tecla "ENTER".

5.3.5 **Gerenciador da base de conhecimento**

Através do gerenciador de base de conhecimento podemos executar as operações usuais com arquivo, tais como carregar ou salvar um arquivo, pedir o diretório, ou outras operações através de um "shell", ou seja de um retorno temporário ao sistema operacional.

5.3.6 Compilador

Uma vez editado todo conhecimento necessário, a ferramenta permite que a base de conhecimento seja convertida em um programa Prolog que possa ser compilado para dar origem a um SE para diagnose. Como nosso intuito é apenas demonstrar a viabilidade de certas idéias, construiu-se um compilador simples capaz de gerar um SE sem grande sofisticação. Tal SE, por exemplo, não permite fazer indagações acerca de suas conclusões, de sua motivação, ou dos procedimentos prescritos para teste e reparo. A sintaxe recohecida pelo compilador encontra-se descrita no item A.4.4. do ANEXO.

CAPÍTULO VI

CONCLUSÃO

Nesse trabalho apresentamos um estudo detalhado da área de Aquisição de Conhecimento, da metodologia KADS, da diagnose de equipamentos e construímos uma ferramenta de Aquisição de Conhecimento que emprega um modelo explícito da diagnose de equipamentos, concebido com o auxílio da metodologia KADS.

Exploramos os problemas da Aquisição de Conhecimento e as técnicas existentes para tratá-los, incluindo abordagens pouco conhecidas como a Aquisição de Conhecimento baseada em linguagem, por aprendizado de máquina, por ferramentas e mesmo certas técnicas psicológicas não muito conhecidas. Julgamos que a taxonomia que propusemos pode auxiliar o Engenheiro do Conhecimento e é um passo na direção de uma metodologia mais geral que trate com toda profundidade especificamente do problema da extração do conhecimento do especialista.

Expusemos de uma maneira detalhada a metodologia KADS que apesar de suas importantes características ainda é pouco conhecida fora da Europa devido, em parte, à falta de textos acessíveis sobre a mesma. Com esse trabalho ajudamos a superar essa dificuldade apresentando um resumo da metodologia ajudando a divulgá-la assim como aos importantes avanços que ela apresenta na construção de Sistemas Baseados em Conhecimento. A própria ferramenta de Aquisição de Conhecimento de que trata esse trabalho é um importante fator de auxílio ao uso e compreensão da metodologia KADS.

Propusemos nesse trabalho uma abordagem para a concepção e representação de modelos do domínio que, embora inspirada na KCML do KADS, não se prende totalmente a essa linguagem. Aonde necessário

ajustes e soluções de compromisso foram feitas. O KADS, por exemplo, prescreve uma separação entre o modelo da especialidade e o modelo de como esse conhecimento será implementado em um computador. Na nossa ferramenta não há uma separação explícita entre esses dois modelos. Apesar disso, deve-se mencionar que o KADS não é totalmente claro a este respeito pois considera que o Modelo Conceitual pode ser executável, ainda que de maneira ineficiente.

Com nosso estudo da diagnose apresentamos e explicamos os conceitos, problemas e estratégias dessa atividade. Contribuiu-se com um modelo que possue uma abordagem simples e poderosa para tratar esse problema, assim como ajuda a esclarecer melhor o conceito de modelagem de nosso trabalho. Propusemos também uma possível abordagem para incluir uma camada de estratégia que dê maior flexibilidade à diagnose.

O trabalho culminou com a concepção de uma ferramenta de Aquisição de Conhecimento capaz de gerar Sistemas Especialistas para diagnose de equipamentos com o emprego de um modelo explícito. Tal ferramenta pode ser aperfeiçoada em um grande número de aspectos. No que tange à interface com o usuário, por exemplo, a ferramenta poderia ser capaz de solicitar ao especialista elementos do modelo de uma maneira mais direta por meio de um diálogo e testar simultaneamente a consistência da informação obtida, de maneira a não exigir do especialista conhecimentos prévios acerca da metodologia KADS ou, de maneira mais ampla, da inteligência artificial. Uma tal abordagem por diálogos exigiria a especificação de um Modelo de Cooperação por parte do Engenheiro de Conhecimento através de um editor para Modelos de Cooperação, cuja formulação não é trivial.

Outro importante aperfeiçoamento da ferramenta seria aprofundar o tratamento do problema da integração de técnicas de extração de conhecimento mais sofisticadas com o modelo implícito à ferramenta. Existem certas técnicas de Aquisição de Conhecimento que são particularmente indicadas a adquirir conhecimentos de certos tipos. Para obtermos a decomposição funcional do equipamento a ser diagnosticado, por exemplo, poderia-se usar a técnica de separação de cartões, a técnica de diagramas, a análise hierárquica de Johnson ou as árvores ordenadas por recordação. No caso mais geral, diversas técnicas poderiam ser utilizadas, sendo que cada uma inevitavelmente possuiria uma maneira distinta de representar o conhecimento obtido do especialista. Ao trabalharmos com mais de um modelo seria necessário abordar questões relativas à consistência e completude das representações e da conversibilidade entre os resultados da aplicação das diversas técnicas.

Do ponto de vista da representação do conhecimento, utilizamos uma linguagem inspirada na KCML que é construída parte sobre a linguagem PROLOG e parte com uma linguagem própria. Isto exige que o Engenheiro de Conhecimento conheça PROLOG e dificulta o entendimento do modelo utilizado pela ferramenta pelo especialista. Seria interessante conceber uma linguagem mais adequada, mais intimamente ligada ao modelo de quatro camadas do KADS.

É interessante notar que a KCML, da maneira como a empregamos, torna-se uma técnica de representação do conhecimento, assim como o são os frames, as regras, a lógica e outros. Essa é um uso novo para a KCML uma vez que o KADS a concebe não como uma representação de conhecimento para uma máquina de inferências, mas como uma linguagem

puramente descritiva do conhecimento especializado. Seria interessante analisar as formas como inferências podem ser feitas usando a representação KCML e que tipo de máquina de inferência seria mais adequada para processá-la.

A verificação de consistência que fazemos é superficial, o que se justifica pelo fato de ser esse um aspecto de difícil tratamento que não é nosso principal foco de interesse. O estudo do problema da consistência de uma base de conhecimento com representação de conhecimentos profundo e, além disso, de diferentes tipos, demandaria uma análise detalhada.

A ferramenta descrita aqui apresenta o conhecimento especializado de uma forma mais estruturada, pois tal conhecimento é organizado de acordo com um modelo que descreve os princípios mais básicos que alicerçam esse conhecimento. Isso permite uma maior transparência da base de conhecimento facilitando a sua manutenção ou o seu emprego para fins didáticos.

O modelo serve também de guia para o processo de Aquisição de Conhecimento, fazendo com que essa atividade deixe de depender de uma compreensão abstrata do conhecimento empírico inferida pelo Engenheiro de Conhecimento a partir da análise de informações fornecidas por um especialista e passe a se guiar por uma compreensão oriunda de uma análise ordenada da atividade executada pelo especialista, feita de uma maneira racional prescrita por uma metodologia abrangente.

De maneira geral, o trabalho divulgou o emprego de modelos na engenharia de conhecimento, campo novo, que vem ganhando crescente importância através de estudos de Sistemas Especialistas de segunda

geração, com um conhecimento acerca dos princípios que alicerçam a atividade especializada a que se destinam, através de estudos de linguagens para descrever esse modelo e, particularmente, da metodologia KADS, que além dos tópicos já citados, concebe modelos pré-definidos de tarefas elementares que são empregados como módulos construtivos de Sistemas Especialistas. Finalizando, apresentamos uma ferramenta que introduz o conceito de uma Aquisição de Conhecimento baseada em um modelo explícito e alterável. O objetivo era demonstrar a viabilidade dessa abordagem o que acreditamos ter conseguido.

BIBLIOGRAFIA

[ALBERT 90] P. ALBERT & G. VOGEL, KOD-STATION: un environnement intégré pour le génie cognitif. Génie Logiciel & Systèmes Experts, n. 19, Juin 1990.

[ALEXANDER 86] J.H. ALEXANDER, M.J.FREILING, S.J. SHULMAN, S. REHFUSS & S.L. MESSICK, Ontological Analysis: An Ongoing Experiment. International Journal of Man Machine Studies, 1986.

[ANJEWIERDEN 87] A. ANJEWIERDEN, The KADS System. Proceedings of the First European Workshop on Knowledge-Acquisition for Knowledge-Based Systems, Reading University, September 1987.

[BELL 87] J. BELL, Teaching Engineers: The Human Side of Knowledge Engineering. Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems, Reading University, September 1987.

[BENBASAT 89] I. BENBASAT & J.S. DHALIWAL, A framework for the validation of knowledge acquisition. Knowledge Acquisition, vol. 1, no. 2, 1989.

[BENNET 85] J. BENNETT, ROGET: a knowledge-based system for acquiring the conceptual structure of a diagnosis expert system. Journal of Automated Reasoning, vol. 1, 1985.

[BERRY 87] D.C. BERRY, The problem of implicit knowledge. Expert Systems, vol. 4, no. 3, August 1987.

[BOOSE 88] J.H. BOOSE & B.R. GAINES, Knowledge Acquisition Tools for Expert Systems. 2nd Edition, Vol. 2, Academic Press, London, 343pp, 1988.

[BOOSE 89] J.H. BOOSE, A Survey of Knowledge Acquisition Techniques and Tools. Knowledge Acquisition, vol. 1, no. 1, March 1989.

[BOOSE 89A] J.H. BOOSE, D.B. SHEMA & J.M. BRADSHAW, Recent progress in AQUINAS: a knowledge acquisition workbench. Knowledge Acquisition, vol. 1, no. 2, 1989.

[BRANCHMAN 78] R.J. BRANCHMAN, On the Epistemological status of semantic networks. N.V.Findler (Ed.) Associative Networks, New York, Academic Press, 1978.

[BREUKER 87a] J. BREUKER & B. WIELINGA, Knowledge Aquisition as Modeling Expertise: The KADS Methodology. Proceedings of the First European Workshop on Knowledge-Acquisition for Knowledge-Based Systems, Reading University, September 1987.

[BREUKER 87b] J. BREUKER (Ed.), B. WIELINGA, G. SCHREIBER, P. DE GREEF, R. DE HOOG, M. VAN SOMEREN, J. WIELEMAKER, J.P. BILLAULT, M. DAVOODI & S. HAYWARD, Model Driven Knowledge Aquisition: Interpretation Models. Deliverable task A1, ESPRIT P1098, University of Amsterdam, 1987.

[BREUKER 87c] J. BREUKER & B. WIELINGA, Use of Models in the Interpretation of Verbal Data. Knowledge Acquisition for Expert Systems: A pratical Handbook, Alison L. Kidd, Plenun Press (Ed.), New York and London, 1987.

[BREUKER 88] J. BREUKER & B. WIELINGA, Models of Expertise in Knowledge Acquisition. Memorandum 103 of the VF-project Acquisition of Expertise, ESPRIT P1098, University of Amsterdam, August 1988.

[BRUNET 90] E. BRUNET & G. DORBES, KADS & Merise vers une unification du génie cognitif et du génie logiciel. Génie Logiciel & Systèmes Experts, n. 19, Juin 1990.

[BYLANDER 86] T. BYLANDER & B. CHANDRASEKARAN, Generic Tasks in Knowledge-based Reasoning: The 'right' Level of Abstraction for

Knowledge Acquisition. Special issue of the 1st Knowledge Acquisition for Knowledge Based Systems Workshop, 1986.

[BYLANDER 86] T. BYLANDER & S. MITTAL, CSRL: A Language for Classificatory Problem-solving and Uncertainty Handling. AI Magazine, August 1986.

[CARBONELL 89] J.G. CARBONELL, Introduction: Paradigms for Machine Learning. Artificial Intelligence, vol. 40, September 1989.

[CARNOTA 87] R.J. CARNOTA & A.D. TEZKIEWICZ, Sistemas expertos y representacion del conocimiento. I-EBAI, Ed. Unicamp, Campinas, Brasil, 1987.

[CHASE 73] W.G. CHASE & H.A. SIMON, The mind's eye in chess. W.G. Chase (ed.), Visual information processing. Academic Press, New York, 1973.

[DAVIS 82] R. DAVIS & D.B. LENAT, Knowledge-Based Systems in Artificial intelligence. McGraw Hill, New York, 1982.

[DAVIS 84] R. DAVIS, Reasoning from first principles in electronic troubleshooting. Developments in Expert Systems, 1984.

[DAVIS 88] R. DAVIS & W. HAMSCHER, Model-based Reasoning: troubleshooting. Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence. H.E. Shrobe (ed.), Morgan Kaufmann Publishers Inc., San Mateo, California, 1988.

[ESHELMAN 88] L. ESHELMAN, MOLE: A knowledge acquisition tool that buries certainty factors. International Journal of Man-Machine Studies, vol. 29, No. 5, 1988.

[EVANSON 88] S.E EVANSON, How to Talk to an Expert. AI Expert, February 1988.

[FINK 85] P. FINK, J. LUSTH & J.DURAN, A general expert system design for diagnostic problem solving. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 7, no. 5, 1985.

[FINK 87] P. FINK & J. LUSTH, Expert systems and diagnostic expertise in the mechanical and electrical domains. IEEE Trans. on Systems, Man and Cybernetics, vol. 17, no. 3, 1987.

[FREILING 85] M.J. FREILING, J. ALEXANDER, S.L. MESSICK & S. SHULMAN, Starting a knowledge engineering project: a step by step approach. AI-Magazine, no. 6, 1985.

[FREILING 89] M.J. FREILING & C.E. JACOBSON, A New Look at Inference Synthesis. Knowledge Acquisition, vol. 3, no. 1, 1989.

[FULTON 90] S.L. FULTON & C.O. PEPE, An introduction to model-based reasoning. AI Expert, no. 1, January 1990.

[GALE 86] W.A. GALE, Knowledge Based Knowledge Acquisition for Statistical Consulting System. Proceedings of the Knowledge Acquisition for KBS Workshop, Banff, Canada, 1986.

[GIORNO 88] F. GIORNO, V. DUARTE, J. DAMSKI & P. MILANI, Methods and Techniques for Knowledge Elicitation. IBM Rio Scientific Center, CCR065, December 1988.

[GONÇALVES 86] C.A. GONÇALVES, Aquisição e Representação de Conhecimento para Sistemas Especialistas. Tese de Doutorado, FEA/USP, 1986.

[GRECO 87] G. GRECO & A.F. ROCHA, The Fuzzy Logic of Text Understanding. Fuzzy Sets and Systems, no. 19, 1987.

[GREEF 85] P. de GREEF & J. BREUKER, A case study in structured knowledge acquisition. Proceedings of the Ninth International Joint Conference on Artificial Intelligence. Los Angeles, CA, 1985.

[GREEF 87] P. de GREEF, G. SCHREIBER, J. WIELEMAKER & B. WIELINGA, The Statcons Case Study. Esprit Project P1098, Delivarable E2.3 (experiment F2), University of Amsterdam, 1987.

[GRUBER 87] T.R. GRUBER & P.R. COHEN, Design for Acquisition: Principles of Knowledge System Design to Facilitate Knowledge Acquisition. International Journal of Man-Machine Studies, Vol. 26, No. 2, 1987.

[HAYES-ROTH 84] F. HAYES-ROTH, The Knowledge Based Expert System: A tutorial. IEEE Computer, September 1984.

[HAMSCHER 89] W. HAMSCHER & R. PATIL, Model-Based Diagnosis. Tutorial: MA1, Proc. of the International Joint Conference on Artificial Intelligence, Detroit, 1989.

[HART 87] A. HART, Role of induction in Knowledge elicitation. A. Kidd (ed) Knowledge Acquisition for Expert Systems. New York 1987.

[HOFFMAN 87] R.R. HOFFMAN, The Problem of Extracting the Knowledge of Experts from the Perspective of Experimental Psychology. AI Magazine, Summer, 1987.

[JOHNSON 86] P. JOHNSON, I. ZAULKERMAN & S. GARBER, Specification of Expertise. International Journal of Man Machine Studies, no. 26, 1986.

[KASSIRER 82] J.P. KASSIRER, B.J. KUIPERS & G.A. GORRY, Toward a Theory of Clinical Expertise. The American Journal of Medicine, Vol. 73, August 1982.

[KERAVNOU 86] E.T. KERAVNOU & L. JOHNSON, Competent expert systems: a case study in fault diagnosis, London, Kogan Page, 1986.

[KINTSCH 77] W. KINTSCH, Memory and Cognition. John Wiley & Sons, Inc., 490pp, 1977.

[KLINKER 87] G. KLINKER, J. BENTOLILA, S. GENETET, M. GRIMES & J. McDERMOTT, KNACK: Report-driven knowledge acquisition. International Journal of Man-Machine Studies. Vol. 26, No. 1, 1987.

[KOLODNER 84] J.L. KOLODNER, Towards an understanding of the role of experience in the evolution from novice to expert. Development in expert systems, M.J. Coombs (ed.), Academic Press, London, 1984.

[KOLOKOURIS 86] A.T. KOLOKOURIS, Machine Learning. Byte, November 1986.

[KUIPERS 84] B. KUIPERS & J.P. KASSIRER, Causal Reasoning in Medicine: Analysis of a Protocol. Cognitive Science, no. 8, October 1984.

[LEÃO 87] B.F. LEÃO & A.F. ROCHA, A Methodology Proposal for Knowledge Acquisition. Proc. 7th Int. Congress on Medical Informatics, September 1987.

[LESSER 77] V.R. LESSER & L.D. ERMAN, A retrospective view of HEARSAY-II. Proceedings of the 5th International Joint Conference on Artificial Intellingence, London, Pitman, 1988.

[LINSTER 89] M. LINSTER, Towards a second generation knowledge acquisition tool. Knowledge Acquisition, Vol. 1, No. 2, 1989.

[MARCUS 85] S. MARCUS, J. MCDERMOTT & T. WANG, Knowledge Acquisition for Constructive Systems. Proceedings of the Ninth International Joint Conference on Artificial Intelligence, Los Angeles, CA, August 1985.

[McCARTHY 82] J. McCARTHY, Some Expert Systems Need Common Sense. Annals New york Academy of Sciences, 1982.

[MICHALSKI 83] R.S. MICHALSKI, A theory and methodology of inductive learning. Machine Learning: An Artificial Intelligence Approach,

R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds), Tioga Publishers, Palo Alto, California, 1983.

[MILANI 89] P. MILANI, F. GIORNO & J.DAMSKI, Um Modelo de Interpretação para Diagnose de Equipamentos. VI SBIA, RJ, Novembro de 1989.

[MILNE 87] R. MILNE, Strategies for Diagnosis. IEEE Trans. on Systems, Man, and Cybernetics, no. 3, May/June 1987.

[MORIK 86] K. MORIK, Acquiring Domain Models. Proceedings of the Knowledge Acquisition for KBS Workshop, Banff, Canada, 1986.

[MOTTA 89] E. MOTTA, T. ROJAN & M. EISENSTODT, Knowledge Acquisition as a Process of Model Refinement. International Journal of Man-Machine Studies, 1989.

[MUSEN 87] M.A. MUSEN, L.M. FAGAN, D.M. COMBS & E.H. SHORTLIFE, Use of a domain model to drive an interactive knowledge-editing tool. International Journal of Man-Machine Studies, no 26, 1987.

[OBERMEIER 89] K.K. OBERMEIER & J.J. BARRON, Time to Get Fired Up. Byte, August 1989.

[OLSON 87] J.R. OLSON & H.H.REUTER, Extracting Expertise from Experts: Methods for Knowledge Acquisition. Expert Systems, Vol.4, No.3, 1987.

[PEARCE 88] D.A. PEARCE, The induction of fault diagnosis systems from qualitative models. Proceedings of the National Conference on Artificial Intelligence, AAAI-88, Saint Paul 1988.

[PENG 87] Y. PENG & J.A. REGGIA, A Probabilistic Causal Model for Diagnostic Problem Solving. Part II: Diagnostic Strategy. IEEE Transactions on Systems, Man, and Cybernetics, Transactions on Systems, Man, and Cybernetics, no. 3, May/June 1987.

[PIPITONE 86] F. PIPITONE, The FIS Electronics Troubleshooting System. Computer, July 1986.

[RICH 83] E. RICH, Artificial Intelligence. McGraw-Hill International Editions, 436pp, 1983.

[RUBERG 89] K. RUBERG, S.M. CORNICK & K.A. JAMES, House calls: building and maintaining a rule-base. Knowledge Acquisition, December 1989.

[SCHIAVINI 90] M.M. SCHIAVINI, "Aquisição de Conhecimento". Relatório Técnico do Depto. de Computação e Automação Industrial da Unicamp, RT 03/90.

[SCHIAVINI 91] M.M. SCHIAVINI, "CAKE - Uma Ferramenta de Aquisição de Conhecimento". Relatório Técnico do Depto. de Computação e Automação Industrial da Unicamp, RT 06/91.

[SCHREIBER 88] G. SCHREIBER, J. BREUKER, B. BREDEWEG & B. WIELINGA, Modelling in KBS Development. Second European Knowledge Acquisition Workshop EKAW'88. Bonn, 1988.

[SLOMAN 80] A. SLOMAN, The Computer Revolution in Philosophy. Harvester, Brighton 1980.

[STEELS 89] L. STEELS, Diagnosis with a function-fault model. Applied Artificial Intelligence: An International Journal, Special issue on causal modeling, Hemisphere Publishing, New York, Vol. 3, no. 2-3, 1989.

[STICKLEN 89] J. STICKLEN, B. CHANDRASEKARAN & W.E. BOND, Distributed causal reasoning. Knowledge Acquisition, no. 1, 1989.

[VALTORTA 89] M. VALTORTA, KADS vs. KEATS. Technical Report TR89009, Department of Computer Science, University of South Carolina, Columbia, August 1989.

[VANDAMME 87] F. VANDAMME, Knowledge Extraction from Experts in View of the Construction of Expert Systems. Expert Judgement and Expert Systems, J.Mumpower et al. (Eds), Springer-Verlag, 1987.

[VAUDET 90] J.P. VAUDET, J. GUYOT & D. DELMAS, Des méthodes pour les systèmes experts. Génie Logiciel & Systèmes Experts, n.19, Juin 1990.

[WESTPHAL 89] C.R. WESTPHAL & D.R. BLAUCHARD, A compendiun of Knowledge Acquisition References. Knowledge Acquisition Special Issue, no. 110, october 1989.

[WIELINGA 84] B.J. WIELINGA & J.A. BREUKER, Interpretation of verbal data for knowledge acquisition. Advances in Artificial Intelligence, T. O'Shea (ed.), Elsevier Science Publishers, B.V. NorthHolland, 1984.

[WIELINGA 86] B.J. WIELINGA & J.A. BREUKER, Models of Expertise. Proceedings of the Seventh European Conference on Artificial Intelligence (ECAI'86), Brighton, July 1986.

[WIELINGA 88] B.J. WIELINGA, B. BREDEWEG & J.A. BREUKER, Knowledge Acquisition for Expert Systems. R. Nossum (ed) Proceedings of the ACAI'87, Berlin, Springer, 1987.

[WIELINGA 89] B.J. WIELINGA, G. SCHREIBER & P. DE GREEF, Synthesis Report. Deliverable Y3, Esprit Project P1098, Memorandum 109, March 1989.

ANEXO

DESCRIÇÃO DA IMPLEMENTAÇÃO

A.1. INTRODUÇÃO

A implementação da ferramenta de Aquisição de Conhecimento foi feita por meio da linguagem TURBO PROLOG versão 2.0, com o auxílio do TURBO TOOLBOX - conjunto de predicados fornecidos opcionalmente com o TURBO PROLOG que agilizam o desenvolvimento de aplicativos.

O sistema será decomposto em quatro módulos para facilitar o seu entendimento:

a) Editor de conhecimento. O editor permite a entrada e a edição de conceitos, relações, estruturas, Metaclasses, Fontes de Conhecimento e Tarefas.

b) Verificador de consistência. O verificador checa a consistência das informações e informa os problemas localizados.

c) Visualizador (browser). O visualizador permite que o usuário analise as informações digitadas por meio de gráficos ou de tabelas.

d) Compilador. O compilador converte as informações para um Sistema Especialista escrito em TURBO PROLOG, que faz a diagnose de acordo com as informações fornecidas.

A construção de um Sistema Especialista com a ferramenta CAKE envolve duas etapas: inicialmente o Engenheiro do Conhecimento elabora um modelo de interpretação e o transfere para a ferramenta que se encarrega de converte-lo para um "shell" de diagnose. Na segunda etapa o especialista, partindo desse "shell" específico, e com o auxílio da ferramenta, constroi o Sistema Especialista final.

1) O Engenheiro do Conhecimento elabora o Modelo de Interpretação da diagnose em que está interessado a partir do

processo usual de Aquisição de Conhecimento prescrita na metodologia KADS. O modelo é transcrito para a sintaxe empregada pela ferramenta CAKE e digitado com o auxílio do editor de Metaclasses, de Fontes de Conhecimento e de Tarefas. A base de dados do modelo é convertida para um "shell" escrito em TURBO PROLOG com o uso do compilador da ferramenta. Esse "shell" pode ser diretamente executado com o auxílio do interpretador do TURBO PROLOG ou convertido para uma forma executável a partir do Sistema Operacional com o compilador TURBO PROLOG que é o produto final da primeira etapa.

Caso o Modelo de Interpretação concebido pelo Engenheiro do conhecimento seja significativamente diferente do Modelo básico da ferramenta, poderá ser necessário que o Engenheiro do Conhecimento altere o código fonte da ferramenta CAKE de maneira a adaptá-la ao seu modelo. Tanto a interface de entrada de dados quanto o verificador de consistência poderão ter que ser adaptados.

2) Feito isso, a ferramenta está pronta para ser entregue ao especialista ou especialistas para que o(s) sistemas(s) especialista(s) final(is) de diagnose seja(m) construído(s). Para tanto o especialista preencherá a camada do domínio do modelo de interpretação, concebido pelo Engenheiro do Conhecimento, com o auxílio do editor de conceitos, relações e estruturas. Durante esse processo, o especialista poderá com o visualizador (browser) ter uma visão global das informações que ele já forneceu e com o verificador de consistência poderá localizar erros em seu modelo de uma maneira simples.

Terminado o processo de edição da camada de domínio, o especialista chamará o "shell" de diagnose criado pelo Engenheiro do Conhecimento para verificar o desempenho global do sistema.

A.2. ORGANIZAÇÃO GERAL DA IMPLEMENTAÇÃO

O código fonte da ferramenta foi armazenado nos seguintes arquivos:

a)"GERAL.PRO" - Contém o menu principal e os predicados que gerenciam os demais módulos. Engloba o sistema de manipulação de arquivos (carregar, salvar e diretório), acesso ao sistema operacional, o sistema de verificação de consistência, todo o sistema de visualização (browser) - com exceção da parte gráfica - e o compilador (a menos do compilador de tarefas).

b)"DIAGRAMA.PRO" - Contém a parte gráfica do visualizador que permite a construção de um grafo que representa graficamente a estrutura do equipamento. É possível caminhar pelo gráfico, selecionar itens e solicitar informações mais detalhadas sobre o elemento selecionado.

c)"COMPILAD.PRO" - É o compilador de tarefas que analisa sintaticamente a descrição da tarefa feita pelo Engenheiro do Conhecimento, apresenta mensagens explicativas para os eventuais erros localizados, permite a edição desses erros e gera o código Prolog associado à tarefa.

d)"SHOW.PRO" - Contém a tela de apresentação do sistema com a palavra CAKE que surge no meio da tela em letras pequenas e, em

seguida, caminha pela tela ao mesmo tempo em que o tamanho das letras vai sendo aumentado.

e)"EDITOR.PRO" - Contém toda a parte de definição e edição de conceitos, relações, estruturas, Metaclasses, Fontes de Conhecimento e tarefas. É um editor amigável que orienta, a cada momento, que tipo de informação deve ser fornecida.

f)"DESENHO.PRO" - Apresenta graficamente a Estrutura de Inferências. Tal módulo é ativado quando se seleciona a opção "VER" do editor de Fontes de Conhecimento.

g)"GLOBDEF.PRO" - É o arquivo que contém as definições dos predicados globais ou seja dos predicados que são usados em mais de um dos arquivos que compõem o sistema CAKE.

Além desses arquivos existem também os arquivos referentes ao TOOLBOX responsáveis por certas facilidades destinadas à entrada de dados. Os arquivos são os seguintes:

h)"PULDOWCK.PRO": implementa um menu tipo "pulldown" - que é um tipo de menu aonde cada item está associado a um outro menu que é apresentado abaixo do item selecionado. Esse predicado do TOOLBOX foi alterado para fazer com que o "pulldown" menu apareça na tela três linhas abaixo da posição normal.

i)"LINEINP.PRO": implementa uma linha de entrada de dados do tipo "lineinput" - que é um campo de entrada de dados alterável, de tamanho ajustável, com um título apresentado à esquerda e um possível valor "default" associado ao campo de edição.

Esse predicado do TOOLBOX foi alterado para facilitar a entrada dos dados e, principalmente, evitar erros de digitação. Para tanto, cada vez que o usuário digita uma letra apresenta-se o nome completo

de um elementos da base de dados a que ele possivelmente esteja se referindo. Por exemplo, se é esperado que o usuário digite o nome de um teste, e se existem na base de dados, entre outros, testes cujos nomes são "tanque" e "torque", quando for digitada a letra "t", aparecerá na linha de edição a palavra "tanque" (ou "torque") e, ao ser digitada a segunda letra "o", pelo usuário, será exibida a palavra "torque". Esse sistema de ajuda pode ser desativado a qualquer momento bastando para tanto que o usuário saia do modo "overwrite" e entre no modo "insert".

j)"FILENAME.PRO": permite a entrada de uma forma mais amigável de nomes de arquivos que devem ser carregados. Para tanto existe um "lineinput" aonde é apresentada a extensão esperada para o nome do arquivo. Caso o usuário tecle <ENTER> sem fornecer o nome do arquivo, esse predicado apresentará um menu com os nomes dos arquivos existentes no diretório atual com a extensão presente no "lineinput".

k)"MENU.PRO": possui predicados que implementam diversos tipos de menu.

l)"LONGMENU.PRO": implementa um menu aonde apenas parte das opções disponíveis é visível a cada momento. Com isso torna-se possível a seleção de um item dentro de uma lista com muitos itens que não poderiam ser visualizados simultaneamente na tela.

m)"STATUS.PRO" e "TPREDS.PRO": contém definições e predicados de uso geral.

Os quatro módulos em que se divide o sistema estão esquematizados na figura a.1.

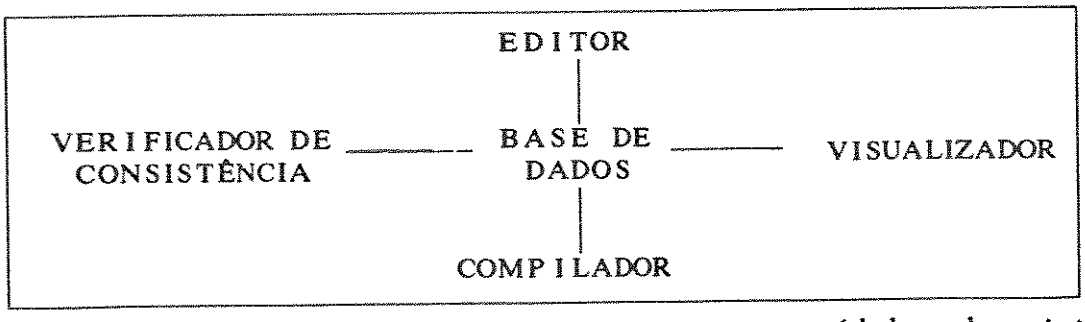


FIGURA A.1 - Diagrama da relação entre os módulos do sistema.

Todos esses módulos operam sobre a base de dados que descreve o modelo conceitual que é representado por meio dos seguintes predicados:

a)CCT(nome, atributo, valor). É a definição dos conceitos associados às partes do equipamento e a outros conceitos necessários para a realização da diagnose. Tais conceitos devem ser caracterizados por um conjunto de atributos e os valores associados a esses atributos.

b)RELAÇÃO(nome, primeiro conceito, segundo conceito). Descreve uma relação do domínio da diagnose que é representada por um conjunto de pares ordenados de conceitos.

c)ESTRUTURA(nome, relação). Uma estrutura é um conjunto de conceitos associados por meio de uma relação. O nome da estrutura deve ser o conceito ao qual estão associados, direta ou indiretamente, o conjunto de conceitos pertencentes à estrutura por meio da relação especificada.

d)METACLASSE(nome, dimensão). Define as Metaclasses existentes e as DIMENSÕES de cada uma delas. Uma Metaclasse é o elo de

comunicação entre duas ou mais Fontes de Conhecimento. A dimensão da Metaclasse representa o número de elementos das mensagens que são trocadas entre as Fontes de Conhecimento por meio dessa Metaclasse.

e)KS(nome, descrição). Esse predicado da base de dados descreve as Fontes de Conhecimento existentes o que é feito pela descrição da inferência realizada pela Fonte de Conhecimento por meio da linguagem declarativa PROLOG.

f)TASK(nome, descrição). Define e descreve as tarefas existentes. A descrição das tarefas é feita por meio de uma linguagem "procedimental" de acordo com uma sintaxe que será explicitada mais à frente nesse anexo.

g)TST(nome, conceito a testar, atributo, valor esperado, custo, descrição). É a declaração dos testes existentes. Cada teste está associado ao atributo do conceito que deve ser testado. A caracterização de um teste envolve também o valor esperado desse teste (ou seja o valor do atributo do conceito observado com o teste quando o equipamento opera normalmente), o seu custo (que pode ser nulo, baixo, médio ou alto) e a descrição do teste que é feita por meio de um texto apresentado ao usuário.

h)EXPLIC(nome, conceito, atributo, valor). Uma vez determinada a causa da falha - que é um conceito associado a uma parte do equipamento defeituoso - o passo seguinte é determinar qual a explicação da falha. É com esse objetivo que esse predicado da base de dados especifica as explicações possíveis e caracteriza cada uma delas. Tal caracterização é feita por meio de um conjunto de observáveis - valores de atributos de conceitos que devem estar presentes caso a explicação em questão seja aplicável - que permitem

que se determine se uma certa explicação pode ou não explicar a falha observada.

Suponhamos, por exemplo, que desejássemos definir a explicação "furo no tanque de combustível", caracterizada pelos observáveis "o nível do tanque de combustível está baixo" e "existe um furo no tanque". Isso poderia ser feito, por exemplo, da seguinte forma:

EXPLICAÇÃO(furo no tanque, tanque, nível, baixo)

EXPLICAÇÃO(furo no tanque, tanque, furo, existe)

O termo "tanque" é um conceito, "nível" e "furo" são dois atributos desse conceito, "baixo" e "existe" são os valores desses atributos que são observados quando há um furo no tanque.

i)RPR(nome, explicação, custo, descrição). Tendo sido determinada a explicação da falha o passo seguinte é reparar o equipamento. Um reparo é definido pelo seu nome, pela explicação da falha a que está associado, pelo seu custo e por sua descrição textual, que destina-se a ser apresentada ao usuário.

j)CONSELHO(nome da explicação, texto). Um conselho é uma mensagem opcionalmente apresentada ao usuário toda vez que a explicação da falha é encontrada. Sua finalidade é evitar que as condições externas adversas que levaram à falha não se repitam no futuro quando a falha tiver sido reparada.

l)SINTOMA(conceito, atributo, valor, identificador). Esse elemento da base de dados, juntamente com a descrição das causas, é utilizado para criar uma tabela de falhas - que é opcional e serve para fornecer uma heurística à diagnose. A tabela de falhas associa um conjunto de manifestações, sintomas, à causa que leva a tal

conjunto de manifestações. O identificador presente nesse predicado da base de dados é um ponteiro gerado internamente.

m)CAUSA(ponteiro para o sintoma, conceito). As causas são possíveis módulos, partes do equipamento, falhos. Cada causa deve estar associada a um conceito. A causa possui um ponteiro para cada um dos sintomas que a caracteriza.

o)NOME(nome da base de dados). Trata-se do nome que foi dado ao arquivo que contém a base de dados.

A.3. OPERAÇÃO

Para se utilizar o sistema deve-se digitar o comando "CAKE" a partir do sistema operacional. No mesmo diretório em que se encontrar o sistema deverão também estar o driver do monitor - que é um arquivo com extensão "BGI" - e os arquivos de definição de letras gráficas - arquivos com extensão "CHR". Em seguida o sistema solicitará o nome do arquivo. O usuário fornecer essa informação de três maneiras:

1) Se desejar criar um novo SE bastará digitar o nome que deseja dar ao mesmo. Fazendo isso o sistema apresentará a mensagem "NOVO ARQUIVO!", para indicar que o arquivo não foi encontrado entre os já conhecidos.

2) Ele poderá alterar a descrição de um SE já existente. Para tanto deverá digitar o nome da base de dados aonde está tal descrição. Assim será criado um arquivo de trabalho com extensão "AUX" que ao final da consulta poderá ser convertido para a versão definitiva "BIN" ou poderá ser descartado.

3) O usuário poderá digitar apenas <ENTER> caso não se recorde do nome da base de dados. Nesse caso o sistema apresentará um menu aonde pode-se selecionar qualquer um dentre os arquivos CAKE presentes no diretório.

A base de dados da descrição de um SE do CAKE é composta por dois arquivos, um com extensão "BIN" aonde se encontra a descrição da camada de inferência e de tarefa, e um com extensão "CKE", com as definições do domínio. O arquivo "BIN" deve ser criado pelo engenheiro do conhecimento e serve de base para criação de um, ou mais, arquivos "CKE".

Uma vez tendo-se declarado o nome do arquivo, o sistema apresentará, no canto superior esquerdo, o nome do arquivo que está sendo editado e no canto superior direito a data atual. Na última linha da tela é apresentado o estado do sistema, ou seja qual é a atividade que está sendo executada. O centro da tela é reservado para as mensagens ao usuário. Na segunda linha encontra-se um menu do tipo "pull-down".

Nesse menu o usuário poderá selecionar uma das seguintes opções:

1) "ARQUIVO". Essa opção possui as seguintes sub opções:

1.1) "CARREGAR" ("carregar arquivo"). Permite que se carregue uma nova base de dados. Caso a base atual tenha sido alterada, o sistema apresentará a seguinte mensagem: "SALVA BASE DE CONHECIMENTO <S/N> ?". Digitando "N" a base de dados atual será descartada.

1.2) "SALVAR" ("salvar arquivo"). Essa opção permite que se salve a base de dados atual.

1.3) "DIRETÓRIO" ("diretorio de arquivos"). Apresenta os arquivos presentes no diretório atual.

1.4) "SISTEMA" ("ir para o sistema operacional"). Permite o acesso ao sistema operacional sem que se abandone o programa. Ao se escolher essa opção aparecerá o "prompt" do DOS. Para retornar ao CAKE bastará que se use o comando "EXIT".

2) "EDITAR". Essa opção permite que se crie ou se altere a base de dados que descreve um determinado equipamento. A forma básica de entrada de dados do editor são os "lineinputs", o editor de texto e os menus. O editor de texto é usado apenas para entrar a descrição das Fontes de Conhecimento e das Tarefas, o restante das informações são fornecidas basicamente por meio dos "lineinputs". Um "lineinput" é uma linha de entrada de dados editável que possui um título e um texto "default". Todos os textos que são digitados em uma "lineinput" são convertidos para maiúsculas. O "lineinput" é um predicado do TOOLBOX que foi alterado para facilitar a entrada de dados (a esse respeito veja a descrição das alterações na parte do anexo que descreve o arquivo "LINEINP.PRO".

O editor possui um "flag" de controle que é o predicado da base de dados "flag_modificacao" que é posicionado toda vez que a base de dados é alterada e só é reposicionado quando a base de dados é salva.

A opção EDITAR possui como subopções (EDITAR) CONCEITO, (EDITAR) RELAÇÃO, (EDITAR) ESTRUTURA, (EDITAR) METACLASSE, (EDITAR) FONTE DE CONHECIMENTO e (EDITAR) TAREFA.

2.1) "CONCEITO" ("editar conceitos"). Existem quatro tipos de conceitos, os conceitos genéricos, associados a partes do equipamento ou a outros elementos necessários à diagnose, os conceitos de teste, que são conceitos com características pré-definidas que facilitam a

descrição de um procedimento de teste, os conceitos de explicação que permitem que se caracterize uma explicação de uma falha e os conceitos de reparo que são conceitos associados aos procedimentos de reparo.

A subopções de (EDITAR) CONCEITO são EDITAR, CRIAR, ELIMINAR e SAIR.

2.1.1) "EDITAR" ("editar um conceito já existente"). Essa opção permite que se altere a definição de um conceito já existente na base de dados. A edição é feita da mesma forma que a criação de um novo conceito. Primeiro seleciona-se o tipo de conceito a ser editado, o conceito é escolhido de uma lista de conceitos existentes na base de dados. Em seguida basta teclar "ENTER" até que se chegue no atributo específico que se deseja alterar. Pode-se então redigitar a característica ou valor do conceito. Teclando "ESC" retorna-se ao menu.

2.1.1.1) "PARTE" ("editar um conceito de parte já existente"). A alteração da definição de um conceito de parte é feita por meio de "lineinputs" cujos valores "default" são os valores atuais da definição do conceito. Para manter esse valor basta teclar "ENTER", para alterar a definição deve-se digitar o novo valor. Cada vez que se tecla "ENTER" passa-se a editar o valor seguinte da definição. Ao se atingir o último item existente, pode-se, acrescentar novos pares atributo-valor à definição, bastando para tanto digitá-los.

2.1.1.2) "TESTE" ("editar um conceito de teste já existente"). A edição das definições de um teste é feita de forma análoga à edição das definições de um conceito de parte. Teclando-se "ENTER" poder-se-á editar, sucessivamente, o nome do teste - lineinput: "NOME DO TESTE

:", o nome do conceito a ser testado - lineinput: "CONCEITO A TESTAR:", o atributo a ser testado - lineinput: "ATRIBUTO A TESTAR:", o valor esperado do teste - lineinput: "VALOR ESPERADO :", e o procedimento de teste por meio do editor cujo título é: "DESCREVA O PROCEDIMENTO DE TESTE".

2.1.1.3) "EXPLICAÇÃO" ("editar um conceito de explicação já existente"). Para editar uma explicação, inicialmente digita-se o nome da explicação com o auxílio do lineinput: "EXPLICAÇÃO:" e, em seguida, edita-se o conselho que será apresentado por, meio de um editor, e redefine-se o conjunto de sintomas que caracterizam a explicação por meio do preenchimento de uma tabela com três campos: conceito, atributo e valor.

2.1.1.4) "REPARO" ("editar um conceito de reparo já existente"). Inicialmente digita-se com o nome do reparo - lineinput: "NOME DO REPARO : ", em seguida digita-se o nome da explicação - lineinput : "EXPLICAÇÃO DA FALHA: ", o custo do reparo, por meio de um menu, e a descrição do procedimento de reparo, por meio de um editor.

2.1.2) "CRIAR" ("criar um novo conceito"). Permite que um novo conceito seja criado. Inicialmente o sistema solicita o nome do conceito a ser criado através de um "lineinput" com o título "NOME:", em seguida deve-se escolher de um menu o tipo do conceito a ser criado que poderá ser um dos seguintes:

2.1.2.1) "PARTE" ("criar um novo conceito de parte"). Um conceito de parte é criado a partir da definição de seus atributos e dos valores de cada um desses atributos. Isso é feito por meio de "lineinputs" cujos títulos são "ATRIBUTO:" e "VALOR:". Cada par atributo-valor que for digitado será apresentado em uma linha

sucessiva. Caso se tecle "ENTER" sem digitar um valor para esses elementos, o sistema retornará ao menu do editor de conceitos. O mesmo ocorrerá se for apertada a tecla "ESC". Caso o nome do conceito a ser criado já exista na base de dados, os novos pares atributo-valor que forem definidos serão acrescentados aos já existentes.

Em geral os valores e os atributos são utilizados com a finalidade de definir testes. Por exemplo, se declaramos que o atributo "TENSÃO" do conceito "FONTE DE ALIMENTAÇÃO" pode assumir os valores "5 V", "menor que 5 V" e "maior que 5 V" então possivelmente desejaremos definir um teste para o atributo "TENSÃO" da "FONTE DE ALIMENTAÇÃO", cujo valor esperado será "5 V". Se esse teste for aplicado o usuário será solicitado a escolher em um menu qual o valor observado da "TENSÃO" da "FONTE DE ALIMENTAÇÃO" e esse menu possuirá as seguintes alternativas: "5 V", "menor que 5 V"e "maior que 5 V".

2.1.2.2) "TESTE" ("criar um novo conceito de teste"). Para se definir um novo teste, as seguintes informações devem ser fornecidas:

a) "CONCEITO A TESTAR:". É o nome da parte do equipamento que deverá ser testada.

b) "ATRIBUTO A TESTAR:". É a característica a ser testada da parte do equipamento visada pelo teste. Por exemplo, pode-se testar o brilho ou o sincronismo da parte de um microcomputador chamada vídeo.

c) "VALOR ESPERADO :". É o valor esperado para o atributo da parte do equipamento verificada pelo teste.

d) "CUSTO:". O custo é selecionado a partir de um menu que possui as seguintes alternativas "NULO", "BAIXO", "MÉDIO" e "ALTO". Internamente o custo NULO é associado ao valor 1, BAIXO a 2, MÉDIO a

3 e ALTO a 4. Nesse menu apenas a opção ativa é visível, para alterar a opção deve-se empregar as setas do teclado.

e) "PROCEDIMENTO DE TESTE". O procedimento de teste é um texto que é digitado a partir de um editor, cujo título é: "DESCREVA O PROCEDIMENTO DE TESTE".

2.1.2.3) "EXPLICAÇÃO" ("criar um novo conceito de explicação"). O primeiro passo para definir uma explicação é digitar, em um editor de texto, o conselho que será apresentado no caso dessa explicação se aplicar a uma particular falha. Esse conselho é opcional, caso não se deseje definir um conselho basta teclar a tecla "ESC". Em seguida deve-se caracterizar a explicação da falha por meio do conjunto de sintomas que a define. Isso é feito por meio da edição de uma tabela cujos campos são: "CONCEITO", "ATRIBUTO" e "VALOR".

2.1.2.4) "REPARO" ("criar um novo conceito de reparo").

a) "EXPLICAÇÃO DA FALHA". É o nome da explicação da falha do equipamento em cuja ocorrência o reparo o reparo é aplicável. Por exemplo, o reparo "tapa buraco do tanque" pressupõe que a explicação da falha seja "tanque de combustível furado".

b) "CUSTO" Valem as mesmas observações do item 2.1.2.2.d.

c) "PROCEDIMENTO DE REPARO". O procedimento de reparo é um texto que é digitado a partir de um editor cujo título é "DESCREVA O PROCEDIMENTO DE REPARO").

2.1.3) "ELIMINAR" ("eliminar um conceito"). Essa opção permite que um conceito de determinado tipo seja eliminado da base de dados. Para tanto, após essa opção do menu ter sido selecionada, o tipo de conceito a ser eliminado é escolhido através de outro menu. Feito

isso, a lista de conceitos é apresentada e aquele que for escolhido será eliminado da base de dados.

A opção eliminar pode eliminar quaisquer dos seguintes tipos de conceitos:

2.1.3.1) "PARTE" ("eliminar um conceito de parte").

2.1.3.2) "TESTE" ("eliminar um conceito de teste").

2.1.3.3) "EXPLICAÇÃO" ("eliminar um conceito de explicação").

2.1.3.4) "REPARO" ("eliminar um conceito de reparo").

2.1.4) "SAIR". Retorna ao menu principal.

2.2) "RELAÇÃO" ("editar relações"). O editor de relações possui as seguintes opções:

2.2.1) "EDITAR" ("editar uma relação já existente"). Teclando-se "ENTER" os pares de conceitos que pertencem à relação em questão são apresentados sucessivamente. Caso se deseje alterar algum deles, basta digitar o novo valor. Teclando-se "ESC" retorna-se ao menu do editor de relações. Quando terminarem os pares de conceitos, novos pares de conceitos poderão ser acrescentados. Os conceitos que forem editados sofrem uma verificação de consistência para detectar a presença de ciclos (veja o item 2.2.2 a seguir) entre eles.

2.2.2) "CRIAR" ("criar uma nova relação"). Uma relação é um conjunto de pares de conceitos. Para se criar uma nova relação deve-se fornecer um nome para a mesma, através do lineinput "NOME" e em seguida o conjunto de pares de conceitos que definem a relação, por meio dos lineinputs "ORIGEM" e "DESTINO". Quando se tiver terminado de digitar os pares, basta que se tecle "ESC" para retornar ao menu do editor de relações. A opção "CRIAR" pode também ser usada para acrescentar novos pares a uma relação já definida anteriormente.

O sistema não admite que se entre com pares de conceitos que introduzam um ciclo. Um ciclo é um conjunto de pares de conceitos aonde há dois conceitos X e Y tais que os pares (X,Y) e (Y,X) pertencem à relação. Isso por que assume-se que toda relação é associativa, ou seja que se (A,B) e (B,C) são pares que pertencem à relação, então (A,C) também pertence. Assim os pares (A,B), (B,C) e (C,A) introduzem um ciclo. Tal tratamento dado às relações reflete o modelo adotado para a diagnose aonde a única relação que o usuário deve definir explicitamente é a relação "PARTE DE" que é associativa e aonde a presença de um ciclo representa uma inconsistência.

2.2.3) "ELIMINAR" ("eliminar uma relação"). Escolhendo essa opção todos os pares que definem a relação selecionada serão eliminados da base de dados.

2.2.4) "SAIR". Essa opção permite que se retorne ao menu principal.

2.3) "ESTRUTURA" ("editar estruturas"). Existem dois tipos de estruturas: a tabela de falhas e as estruturas genéricas (que são conjuntos de conceitos associados por meio de uma relação). A criação de uma tabela de falhas é opcional, ela serve para acelerar o processo de diagnose através da associação entre sintomas e possíveis causas.

2.3.1) "EDITAR" ("editar uma estrutura já existente").

2.3.1.1) "TAB. FALHAS" ("editar uma estrutura de tabela de falhas"). Para editar-se a Tabela de Falhas deve-se escolher o sintoma a ser editado, o que é feito com o auxílio de um menu. Em seguida entra-se com a nova definição do sintoma, ou seja o valor do atributo do conceito que caracteriza o sintoma, ou tecla-se "ENTER"

para manter a definição. Em seguida as causas associadas ao sintoma são editadas da maneira usual.

2.3.1.2) "OUTRAS" ("editar uma estrutura que não seja uma estrutura de falhas"). As "outras" estruturas são as estruturas definidas a partir de uma relação e de um conceito. Com essa opção pode-se alterar a relação associada a tal estrutura. Para tanto inicialmente seleciona-se de um menu a estrutura a ser editada e, em seguida, digita-se no lineinput "RELAÇÃO" o nome da nova relação a ser associada à estrutura.

2.3.2) "CRIAR" ("criar uma nova estrutura").

2.3.2.1) "TAB. FALHAS" ("criar a tabela de falhas"). Permite que se crie uma tabela de falhas. Inicialmente é solicitado que se defina o sintoma pela mensagem "Qual o sintoma <CONCEITO-ATRIBUTO-VALOR> ?" - que é digitado por meio de três lineinputs, cujos títulos são: "CONCEITO", "ATRIBUTO" e "VALOR". Caso o sintoma não tenha sido definido anteriormente, o sistema cria na base de dados esse novo conceito ao qual é associado um identificador numérico para uso como ponteiro. O usuário deverá, então, descrever quais as causas que levam à ocorrência de tal sintoma. Uma causa é o nome de um conceito de uma parte do equipamento que pode estar falho. As causas devem ser digitadas sucessivamente por meio do lineinput "CAUSA".

2.3.2.2) "OUTRAS" ("criar uma estrutura que não seja uma tabela de falhas"). Para se criar uma estrutura genérica deve-se fornecer o nome do conceito raiz - lineinput "NOME" - e o nome da relação - lineinput "RELAÇÃO". Uma estrutura é um conjunto de conceitos que se relacionam, direta ou indiretamente, com o conceito raiz por meio da relação especificada. Em seguida, caso a relação não tenha sido

definida anteriormente, pode-se fornecer o conjunto de pares de conceitos que definem a relação associada à estrutura.

2.3.3) "IDENTIFICAR" ("dar um nome a uma estrutura que não seja tabela de falhas"). Essa opção é idêntica à do item 2.3.2.2, com a exceção que se assume que a relação já foi previamente definida.

2.3.4) "VER" ("visualizar uma estrutura"). Com essa opção o usuário poderá observar uma estrutura na forma de uma tabela ou grafo.

2.3.4.1) "TAB. FALHAS" ("visualizar a tabela de falhas"). Veja item 4.3.1 para maiores detalhes.

2.3.4.2) "OUTRAS" ("visualizar uma estrutura que não seja uma tabela de falhas"). Veja item 4.3.2.

2.3.5) "ELIMINAR" ("eliminar uma estrutura"). Essa opção elimina partes de uma estrutura ou toda uma estrutura.

2.3.5.1) "TAB. FALHAS" ("eliminar parte de uma tabela de falhas").

2.3.5.1.1) "SINTOMAS" ("eliminar um sintoma da tabela de falhas"). O sintoma escolhido a partir do menu, assim como todas as causas a ele associadas, é eliminado da Tabela de Falhas.

2.3.5.1.2) "CAUSAS" ("eliminar uma causa da tabela de falhas"). A causa escolhida a partir do menu é eliminada da base de dados.

2.3.5.2) "OUTRAS" ("eliminar uma estrutura que não seja tabela de falhas"). A estrutura selecionada de um menu é eliminada da base de dados sem no entanto eliminar os pares que definem a relação a ela associada.

2.3.6) "SAIR". Retorna ao menu principal.

2.4) "METACLASSE" ("editar metaclasses"). Permite que se edite as Metaclasses.

2.4.1) "CRIAR" ("criar uma Metaclasse"). Essa opção permite que se crie uma nova Metaclasse. Para tanto o nome da Metaclasse é solicitado a partir de uma "lineinput" cujo título é: "METACLASSE:". Em seguida uma "lineinput" com título "ELEMENTOS:", com valor "default" igual a 1, solicita a dimensão da Metaclasse, que é um número inteiro que indica o numero de elementos que podem estar simultaneamente em uma Metaclasse. Por exemplo, a Metaclasse HIPÓTESE ORDENADA possui dimensão dois pois cada elemento dessa Metaclasse é composto pelo nome da hipótese e por um número que indica sua ordem.

2.4.2) "ELIMINAR" ("eliminar uma Metaclasse"). Permite que uma Metaclasse previamente criada seja eliminada da base de dados.

2.4.3) "SAIR". Volta ao menu principal do editor.

2.5) "F. CONHECIMENTO" ("editar Fontes de Conhecimento). Através dessa opção podemos editar, criar e eliminar Fontes de Conhecimento assim como ver a Estrutura de Inferência..

2.5.1) "EDITAR" ("editar uma Fonte de Conhecimento já existente"). Apresenta a lista das Fontes de Conhecimento existentes na base de dados, permitindo que se escolha aquela que deverá ser editada. A descrição da Fonte de Conhecimento é apresentada dentro de um editor de textos. Os comandos de tal editor são os mesmos do editor do Turbo Prolog (que são semelhantes aos comandos do "wordstar"). Para sair do editor basta que se tecle "ESC" ou "F10" e as alterações serão salvas.

2.5.2) "CRIAR" ("criar uma Fonte de Conhecimento"). Para criar uma nova Fonte de Conhecimento forneça inicialmente o nome da Fonte

de Conhecimento através de um "lineinput" cujo título é "NOME:". Em seguida será apresentado um editor de textos aonde se poderá digitar a descrição da Fonte de Conhecimento. Para maiores detalhes sobre a criação de uma Fonte de Conhecimento, veja o item 5.1.b.

2.5.3) "VER" ("ver a Estrutura de Inferências"). Apresenta, para simples referência, a Estrutura de Inferência da tarefa de diagnose. A Estrutura de Inferência pode ser alterada pelo Engenheiro de Conhecimento a partir de alterações no arquivo DESENHO.PRO.

2.5.4) "ELIMINAR" ("eliminar uma Fonte de Conhecimento"). Apresenta a lista das Fontes de Conhecimento existentes e elimina da base de dados aquela que for selecionada.

2.5.5) "SAIR". Retorna ao menu principal do editor.

2.6) "TAREFA" ("editar tarefas"). Opção que permite editar, criar e eliminar tarefas.

2.6.1) "EDITAR" ("alterar tarefa já existente"). Valem os mesmos comentários do item 2.5.1.

2.6.2) "CRIAR" ("criar uma nova tarefa"). Valem os mesmos comentários do item 2.5.2. Veja o item 5.1.c para obter maiores informações sobre como criar uma tarefa.

2.6.3) "ELIMINAR" ("eliminar uma tarefa"). Valem os mesmos comentários do item 2.5.3.

2.6.4) "SAIR". Retorna ao menu principal.

3) "VERIFICAR". A base de dados contém um conjunto de informações que estão semanticamente associadas. Essa opção do menu permite verificar se essas associações são consistentes. Para tanto

uma série de regras são usadas para verificar a ocorrência de uma das seguintes condições de erro:

- ERRO: "VALORES NÃO DEFINIDOS !".

Normalmente um conceito está associado a um atributo que possui pelo menos dois valores. Quando isso não ocorre o verificador de consistência avisa o usuário e informa qual o atributo e o conceito em questão.

- ERRO: "OBSERVÁVEL NÃO DEFINIDO !".

Um observável ou sintoma é o valor do atributo de certo conceito que está associado a um teste (que verifica sua ocorrência). Todo sintoma deve ser definido por meio da declaração de um conceito, atributo e valor homônimos. A mensagem de erro acima será apresentada se isso não ocorrer.

- ERRO: "TESTE NÃO DEFINIDO !".

Em geral os atributos dos conceitos são características observáveis que são usadas para testar o funcionamento de uma parte do equipamento. Quando há um atributo de um conceito que não segue essa regra (por não estar associado a um teste) é gerado tal aviso (o que não indica necessariamente que há um erro).

- ERRO: "EXPLICAÇÃO NÃO DEFINIDA !".

Essa mensagem avisa que há uma módulo não decomponível não associado a uma explicação. A utilidade dessa mensagem deriva do fato de que toda causa de falha é um conceito não decomponível que deve estar associada a uma explicação.

- ERRO: "EXPLICAÇÃO NÃO DEFINIDA !".

Essa mensagem de erro ocorre quando existe um explicação (referenciada em algum reparo) que não foi definida. Isso é

importante pois todo procedimento de reparo deve estar associado a uma explicação de falha. Deve-se notar que, apesar dessa mensagem de erro ser idêntica à anterior, o usuário é capaz de distinguir entre as duas pelo fato do sistema indicar, após a mensagem de erro, qual o conceito problemático e aonde ele se encontra.

- ERRO: "OBSERVÁVEL NÃO DEFINIDO !".

A Tabela de Falhas é composta por sintomas e por causas. Um sintoma, ou seja um observável, é o valor do atributo de um conceito. Assim todo sintoma deve estar associado à definição de um conceito, de um atributo e de um valor. Quando isso não ocorre essa mensagem de erro é apresentada.

- ERRO: "CONCEITO NÃO DEFINIDO !".

Uma causa de falha é um conceito da Tabela de Falhas associado a um módulo defeituoso. A causa deve estar associadas à definição de um conceito homônimo caso contrário será apresentada essa mensagem de erro.

- ERRO: "EXPLICAÇÃO NÃO DEFINIDA !".

Quando a explicação da causa da falha é localizada um conselho poderá ser apresentado ao operador. Para tanto o conselho deve estar associado à explicação. Se isso não tiver sido feito (se houver um conselho associado a uma explicação que não tenha definição) tal mensagem de erro será apresentada.

- ERRO: "OBSERVÁVEL NÃO DEFINIDO !".

Para determinar qual é a explicação de uma falha emprega-se um conjunto de sintomas que caracterizam as explicações. Esses sintomas devem ser definidos como sendo o valor do atributo de um conceito caso contrário o sistema apresentará essa mensagem de erro.

- ERRO: "CONCEITO NÃO DEFINIDO !".

Uma relação é definida a partir de um conjunto de pares ordenados de conceitos. Se um desses pares contiver um conceito que não tiver definição tal mensagem de erro surgirá na tela.

- ERRO: "CONCEITO NÃO DEFINIDO !".

Uma estrutura é definida por uma relação e um conceito. Em particular, o nome da estrutura, deve estar associado a um conceito homônimo para que não seja gerada essa mensagem de erro.

- ERRO: "REPARO NÃO DEFINIDO !".

Um procedimento de reparo é sugerido ao usuário toda vez que for encontrada a explicação da causa da falha do equipamento. Para tanto, toda explicação deve estar associada a um reparo. Se isso não ocorrer essa mensagem de erro será apresentada.

- ERRO: "CONCEITO NÃO UTILIZADO !".

Os conceitos são definidos para fins de teste, de explicação ou para caracterizar sintomas. Essa mensagem de aviso é gerada se for encontrado na base de dados um conceito que não sirva a nenhuma dessas finalidades.

4) "MOSTRAR". Trata-se do browser ou seja da função que permite que se observe as informações presentes na base de dados de uma forma amigável. Para tanto tais informações são apresentadas na forma de tabelas, grafos ou "frames" (telas com campos pré-definidos) conforme o caso.

4.1) "CONCEITO" ("mostrar conceito").

4.1.1) "PARTE" ("mostrar conceito de parte"). Essa opção cria uma tabela aonde são apresentados os atributos e os valores dos atributos de um conceito de parte. Caso a tabela não caiba

integralmente na tela é possível observá-la integralmente ao percorrê-la com o auxílio das teclas de setas. Se o conceito selecionado não tiver atributos a seguinte mensagem será apresentada: "CONCEITO NÃO DEFINIDO !!!!!!!!".

4.1.2) "TESTE" ("mostrar conceito de teste"). Apresenta, na forma de um "frame", a declaração de um teste, que é composta pelo nome do teste, o atributo do conceito que ele testa, o valor esperado para o teste, seu custo e a descrição textual de como realizá-lo.

4.1.3) "REPARO" ("mostrar conceito de reparo"). Apresenta, na forma de um "frame", a definição de um reparo, ou seja o seu nome, a explicação da falha a que ele se destina a reparar, seu custo e a descrição textual do procedimento de reparo.

4.1.4) "EXPLICAÇÃO" ("mostrar conceito de explicação). Apresenta, na forma de tabela, o conjunto de sintomas que caracterizam uma explicação. Cada sintoma é composto por um conceito, seu atributo e pelo valor desse atributo.

4.2) "RELAÇÃO" ("mostrar relação"). Apresenta os pares ordenados de conceitos através de uma tabela de dois campos: "ORIGEM" e "DESTINO". Se não houver nenhum par de conceitos associado à relação é apresentada a mensagem : "RELAÇÃO NÃO DEFINIDA !!!!!!!!".

4.3) "ESTRUTURA" ("mostrar estrutura"). Existem dois tipos de estruturas: a Tabela de Falhas, que é composta pelos sintomas e causas, e as estruturas definidas a partir de um conceito e de uma relação (pela qual outros conceitos se relacionam a esse conceito).

4.3.1) "TAB. FALHAS" ("mostrar a estrutura tabela de falhas"). É visualizada a partir dos sintomas ou das causas que a ela pertencem.

4.3.1.1) "SINTOMAS" ("mostrar a estrutura tabela de falhas a partir dos sintomas"). Se não houver na base de dados nenhum sintoma, a mensagem "NÃO Há SINTOMAS" é apresentada. Caso contrário a lista dos sintomas existentes é apresentada por meio de um menu. Selecionando-se um sintoma desse menu, serão apresentados em forma de tabela, as causas que sabidamente podem levar à observação desse sintoma.

4.3.1.2) "CAUSAS" ("mostrar a estrutura tabela de falhas a partir das causas"). Se não houver sintomas na base de dados a mensagem "NÃO Há CAUSAS" é apresentada. Caso contrário a lista das causas existentes é apresentada na forma de um menu. Para a causa selecionada, o sistema apresentará, numa tabela, os sintomas que se observa no caso de certo módulo estar falho (tal módulo é denominado causa da falha).

4.3.2) "OUTRAS" ("mostrar estruturas que não são tabela de falha"). Mostra a estrutura definida por uma relação e por um conceito em forma de diagrama. Pertencem a tal estrutura os conceitos que se relacionam direta, ou indiretamente, com o conceito inicial por meio da relação especificada.

A estrutura é apresentada na forma de um grafo aonde se pode observar os conceitos e as relações entre eles. O grafo começa com o conceito homônimo à estrutura, cujo nome pode ser observado na parte de cima do grafo que possui o título "DIAGRAMA DA ESTRUTURA <<nome do conceito e da estrutura>>". O tamanho das letras e o número de conceitos apresentados é ajustado de maneira a otimizar a legibilidade e a parte do grafo visível na tela.

Caso o grafo apresentado seja largo demais para ser representado na tela - mais de 20 conceitos lado a lado, apenas parte do mesmo será visualizada. Nesse caso é possível ver as demais partes utilizando-se as setas "PARA CIMA" - que fará o grafo se deslocar para cima - e "PARA BAIXO" - que fará o grafo se deslocar para baixo.

Caso o grafo seja profundo demais para ser visualizado - mais de quatro conceitos relacionados seqüencialmente contando com a raiz - o grafo será apresentado somente até certa profundidade e os conceitos que estiverem nessa profundidade limite aparecerão na tela seguidos do símbolo "+" que indica que ainda há "mais" conceitos a partir desse.

Para tornar mais simples o grafo apresentado na tela, os conceitos que ocorram mais de uma vez no grafo só são expandidos uma vez. Expandir um conceito significa apresentar os conceitos que com ele se relacionam. Um conceito que não for expandido, pelo fato de já ter sido expandido em outra parte do grafo, é apresentado na tela seguido do símbolo "!".

É possível "caminhar" pelo grafo com o uso das setas. Caminhar significa selecionar os conceitos do grafo sequencialmente (de acordo com a forma como eles se relacionam). Um conceito selecionado pode ser expandido teclando-se "ENTER". As seguintes situações são possíveis:

-Quando se expande um conceito que não é a raiz do grafo, um novo grafo é construído tendo por raiz esse conceito.

-Quando se expande um conceito que é a raiz de uma estrutura se obterá um novo grafo aonde se observam os atributos e valores do conceito. O título desse conceito é "ATRIBUTO DE <<nome do

conceito>>". Caso o conceito não possua atributos será apresentado o grafo dos sintomas associados à sua falha (veja o próximo item), caso o conceito também não possua sintomasserá apresentado o grafo das estruturas de que ele é membro.

-Selecionando-se para expansão um conceito que é a raiz de um grafo aonde se vêem os atributos e valores associados ao conceito, será obtido um grafo associando esse conceito aos sintomas que são observados no caso desse conceito estar associado a uma parte do equipamento defeituosa. O título desse grafo é "SINTOMAS DE <<nome do conceito>>". Caso o conceito não possua sintomasserá apresentado o grafo das estruturas de que ele é membro.

-Selecionando-se para expansão um conceito que é a raiz de um grafo que associa o conceito aos seus sintomas, obtém-se um grafo que relaciona esse conceito às estruturas a que esse conceito pertence. O título desse grafo é "ESTRUTURAS DE <<nome do conceito>>".

-Selecionando-se para expansão um conceito que é a raiz de um grafo que associa o conceito às estruturas a que ele pertence, se obterá novamente o grafo dos atributos do conceito.

-Quando se está no grafo dos atributos de um conceito e se seleciona para expansão um dos atributos desse conceito, se obterá um novo grafo que associará o atributo selecionado ao conceitos que possuem esse atributo e os respectivos valores assumidos pelo atributo em cada um desses conceitos. O título desse grafo é "EXPANSÃO DO ATRIBUTO <<nome do atributo>>".

-Estando no grafo dos atributos de um conceito e selecionando-se para expansão um dos valores do conceito se obterá um novo grafo que associará o valor selecionado aos conceitos que possuem esse valor e

os respectivos atributos associados ao valor em cada um desses conceitos. O título desse grafo é "EXPANSÃO DO VALOR <<nome do valor>>".

-Estando no grafo dos sintomas e selecionando-se um sintoma nada ocorrerá.

-Estando no grafo das estruturas e selecionando-se uma estrutura será apresentado o grafo da estrutura escolhida.

-As demais possibilidades resultam num dos grafos já citados.

A qualquer momento é possível abandonar o módulo gráfico e retornar aos menus de seleção teclando-se "ESC".

4.4) "METACLASSE" ("mostrar Metaclasses"). Permite visualizar, através de uma tabela, as Metaclasses existentes e a dimensão de cada uma. Se não houver nenhuma Metaclasse na base de dados a mensagem "METACLASSES NÃO DEFINIDA !!!!!!!!" é apresentada.

4.5) "F. CONHECIMENTO" ("mostrar Fontes de Conhecimento"). Essa opção apresenta a definição das Fontes de Conhecimento.

4.6) "TAREFA" ("mostrar tarefas"). Por meio dessa opção é possível observar a definição de cada uma das tarefas existentes.

5) "SIST EXP". A opção SE permite que se crie um shell para o domínio da diagnose a partir das informações da base de dados, ou que se teste os conhecimentos da camada de domínio, a partir de um shell construído previamente a partir das informações da camada de inferência e tarefa.

5.1) "COMPILAR" ("compilar sistema especialista"). Essa opção permite que as informações da base de dados referentes às Fontes de Conhecimento e às Tarefas sejam convertidas para um programa Prolog

que descreve um shell para o domínio da diagnose. Para tanto são feitas as definições necessárias, as Fontes de Conhecimento são descritas e a tarefa é compilada gerando assim um arquivo ".PRO" que poderá ser convertido, com o auxílio do Turbo Prolog, para um programa executável que é o próprio SE em diagnose. Só é necessário recompilar o programa quando as informações da camada de inferência ou de tarefa são alteradas. Assim essa opção, em princípio, não deverá ser utilizada pelo especialista, mas apenas pelo Engenheiro de Conhecimento.

O programa PROLOG gerado possui uma série de predicados pré definidos que interpretam as Fontes de Conhecimento e os predicados da tarefa, esses predicados, que também são responsáveis pela interface do SE, estão no arquivo IM.PRO. Além desses predicados pré-definidos outros são criados pelo compilador a partir do tratamento das informações contidas nas Metaclasses, nas Fontes de Conhecimento e nas tarefas.

a) Metaclasses. As Metaclasse estão associadas à comunicação entre as Fontes de Conhecimento, cada fonte de conhecimento é um predicado Prolog e a comunicação entre esses predicados é feito por meio de cláusulas do Prolog associadas às Metaclasses. O compilador da ferramenta usa a dimensão com que foram definidas as Metaclasses para determinar a "aridade" dessas cláusulas. Cada Metaclasse dá origem a uma declaração de uma cláusula para a base de dados do Prolog.

b) Fontes de Conhecimento. Cada fonte de conhecimento é associada a um predicado Prolog. Isso envolve a definição de um predicado homônimo. Para descrever uma Fonte de Conhecimento são

usados os comandos próprios do Turbo Prolog e alguns predicados que foram criados para tornar mais fácil a tarefa do Engenheiro do Conhecimento que são os seguintes:

- AVISA(T): Escreve centralizado a cadeia T na janela principal (centralizada horizontalmente na janela).

- SELECIONE(TITULO) : Seleciona uma dentre um conjunto de alternativas por meio de um menu. As alternativas devem ser definidas usando o predicado da base de dados "OPÇÃO" de aridade 1, através do comando "assert". Por exemplo, para incluir a opção "Não sei" no menu deve-se usar o comando: "assert" (opção: "Não sei"). O menu com as alternativas declaradas terá o título "TITULO". A opção escolhida pelo usuário é retornada através do predicado "OPÇÃO", assim, por exemplo, o comando "OPÇÃO(X)", retornará em X a alternativa selecionada.

- MULTI_SELECIONE(TITULO) : O mesmo que SELECIONE, apenas que permite que o usuário escolha várias opções. Acima do menu é apresentada a mensagem: "Indique os sintomas e, ao terminar, tecle <F10>".

- COMENTARIO(C) : Escreve a cadeia C na janela de comentários. É também possível usar o comando "COMENTÁRIO(C1,C2)" e "COMENTÁRIO(C1,C2,C3)" para escrever as cadeias C1 e C2 ou C1, C2 e C3 (todas na mesma linha).

- SEM_COMENTARIOS : Limpa a janela de comentários.

- PULA_COMENTARIO : Pula uma linha na janela de comentários.

- TESTA(C,A,V) : Retorna o valor "V" do atributo "A" do conceito "C". Se esse valor não for conhecido o sistema automaticamente

determinará um teste para o atributo do conceito e o aplicará. A aplicação de um teste resulta na apresentação da mensagem "Qual o valor dO/A/OS/AS/O(A) ATRIBUTO dO/A/OS/AS/O(A) CONCEITO ?" (aonde as palavras escritas em maiúsculas assumem um valor adequado ao teste em questão), na apresentação da descrição do teste e na apresentação de um menu com os possíveis resultados do teste. Caso não tenha sido definido nenhum teste para o atributo do conceito será apresentado um formato padrão de teste aonde o usuário será solicitado a informar o valor do atributo do conceito que está sendo testado.

- TESTA_HIPOTESE(H,A,V) : É idêntico ao comando "TESTA", apenas com a única diferença que o atributo a ser testado não precisa ser especificado. O próprio sistema determina qual o atributo a ser testado.

- GÊNERO(NOME,GENERO) : Determina o gênero e o número de uma palavra. GÊNERO pode assumir os seguintes valores:

a) "O" se a palavra terminar em "o" (mas não em "ão") e portanto for uma palavra masculina no singular.

b) "A" se a palavra terminar em "a" e portanto for uma palavra feminina no singular.

c) "OS" se a palavra terminar em "os" e portanto for uma palavra masculina no plural.

d) "AS" se a palavra terminar em "as" e portanto for uma palavra feminina no plural.

c) "O(A)" se a palavra terminar em "ão" ou se não for possível determinar o GÊNERO da palavra.

- REPARA(CONCEITO) : Determina um reparo que tenha sido definido para o conceito de explicação CONCEITO e apresenta ao usuário a descrição desse reparo.

- ESPERADO(C,A,V) : Determina o valor esperado "V" do atributo "A" do conceito "C". Isso é feito determinando um teste para o atributo "A" de "C" e obtendo o seu valor esperado, ou seja, o valor em operação normal do atributo do conceito.

- CONSULT_DB : Esse comando limpa a base de dados atual e recarrega a base de dados. Dessa maneira o sistema sera iniciado e estará pronto para uma nova consulta.

- HIPOTESE_MAIS_FACIL(H,C) : Esse predicado é bem sucedido quando existe uma hipótese mais fácil de ser testada que H, ou seja, cujo custo de teste seja menor que o custo de "C". O custo de se testar uma hipótese é o custo do teste de menor custo dessa hipótese ou, caso nenhum teste tenha sido definido para a hipótese, "100".

A descrição de uma Fonte de Conhecimento deve fazer referência à(s) Metaclasse(s) que são empregadas na comunicação com as demais Fontes de Conhecimento. Abaixo é apresentado o exemplo do código Prolog gerado a partir da definição da Fonte de Conhecimento "EXPLICA". Como se pode notar, existem linhas que são predicados do Turbo Prolog, outras que são um dos comandos previamente citados e outras que são referências às Metaclasses por meio de predicados da base de dados.

Primeira parte da definição:

```
EXPLICA :-
    clearwindow,
    fail.
```

Segunda parte da definição:

```
explica :-
     discrepancia(Componente),
     sem_comentarios,
     concat(Componente," ",X),
     concat(X," está com defeito !",Coment),
     comentario(Coment),
     explic(Nome,Componente,_,_),
       retract(explic(Nome,C,A,V)),
       testa(C,A,V),
       not(explic(Nome,_,_,_)),
     assert(explicação(Nome)),
     pula_comentario,
     GÊNERO(Nome,Genero),
     comentario("Por causa d",Genero,": "),
     comentario(Nome), !.
```

Terceira parte da definição:

```
explica :-
     not(discrepancia(_)),
     avisa("O defeito nao foi localizado."),
     readln(_), !.
```

Quarta parte da definição:

```
explica :-
     avisa("Nenhuma explicação foi encontrada !!!!"),
     readln(_), !.
```

O código associado à Fonte de Conhecimento "EXPLICAÇÃO" se compõe de quatro partes.

1) As primeiras três linhas são geradas automaticamente pelo compilador para todas as Fontes de Conhecimento com a finalidade de limpar a tela principal antes de iniciar a execução da inferência associada à Fonte de Conhecimento. O restante do código é transcrição direta da descrição da Fonte de Conhecimento.

2) Na segunda parte busca-se determinar uma explicação para a discrepância que foi determinada anteriormente. "DISCREPÂNCIA" é uma

Metaclasse que serve como elemento de comunicação entre a Fonte de Conhecimento "CALCULA" e "EXPLICA". Essa Metaclasse é representada pelo predicado da base de dados "discrepância" que tem aridade 1, pois a dimensão da Metaclasse "DISCREPÂNCIA" foi previamente declarada como sendo 1. O mesmo se aplica à Metaclasse "EXPLICAÇÃO".

O predicado EXPLICA reflete a definição da Fonte de Conhecimento EXPLICA através da linguagem descritiva Prolog. Uma explicação, descrita no predicado "EXPLIC" da base de dados, é a explicação correta quando todas as características dessa explicação já tiverem sido testadas e confirmadas, o que é feito através do comando TESTA. Quando a explicação é localizada, um comentário é apresentado informando qual é a explicação da falha.

3) Se o processo de localização da falha não for bem sucedido a discrepância não será determinada e nesse caso não será possível saber qual a explicação da falha. Em tal caso a seguinte mensagem será apresentada: "O defeito nao foi localizado." e o sistema aguardará que o usuário confirme que tomou ciência da mensagem teclando "ENTER".

4) Se não houver nenhuma explicação na base de dados que se aplique à situação corrente a seguinte mensagem será apresentada: "Nenhuma explicação foi encontrada !!!!" e o sistema aguardará que o usuário confirme que tome ciência da mensagem e tecle "ENTER".

Descreveremos agora as demais Fontes de Conhecimento utilizadas para testar a ferramenta CAKE, começaremos por IDENTIFICA que é apresentada abaixo. Essa Fonte de Conhecimento carrega a base de dados e solicita que o usuário escolha uma dentre as estruturas definidas na base de dados para ser a discrepância inicial.

```
/* IDENTIFICA */
IDENTIFICA :-
    clearwindow,
    fail.
identifica :-
  consult_DB,
  retractall(OPÇÃO(_)),
  retract(estrutura(Nome,"PARTE DE")),
     assert(OPÇÃO(Nome)),
     not(estrutura(_,"PARTE DE")),
  avisa("QUAL A ESTRUTURA A SER CONSERTADA?"),
  selecione(""),
  OPÇÃO(OPÇÃO),
  asserta(discrepancia(OPÇÃO)), !.
```

A Fonte de Conhecimento DECOMPÕE parte de uma discrepância e gera as hipóteses de falha mais plausíveis. A sua descrição da Fonte de Conhecimento DECOMPÕE "default", em Prolog, é a seguinte:

```
/* DECOMPOE */
DECOMPOE :-
    clearwindow,
    fail.
/* Esqueça as hipoteses e veja se a discrepancia pode ser decomposta
*/
decompoe :-
  retractall(hipotese(_)),
  discrepancia(D),
  not(relacao("PARTE DE",D,_)), !.

/* Assuma como hipotese todas as partes da discrepancia */
decompoe :-
  discrepancia(D),
  relacao("PARTE DE",D,H),
  asserta(hipotese(H)),
  fail.

/* Obtenha os sintomas */
decompoe :-
  retractall(OPÇÃO(_)),
  hipotese(H),
  causa(Id,H),
  sintoma(C,A,V,Id),
  concat(A,"de",X1),
  concat(X1,C,X2),
  concat(X2,"",X3),
  concat(X3,V,Sintoma),
```

```
  retractall(OPÇÃO(Sintoma)),
  assert(OPÇÃO(Sintoma)), fail.

/* Elimine as hipoteses incompativeis */
decompoe :-
  multi_selecione("SINTOMAS APRESENTADOS"),
  sintoma(C,A,V,Id),
  concat(A,"de",Y1),
  concat(Y1,C,Y2),
  concat(Y2,"",Y3),
  concat(Y3,V,Sintoma),
  OPÇÃO(Sintoma),

  retractall(cct(C,A,_)),
  assert(cct(C,A,V)),

  hipotese(H),
  not(causa(Id,H)),
  retract(hipotese(H)), fail.

/* Se nao sobrou nenhuma hipotese assuma todas como possiveis */
decompoe :-
  retract(discrepancia(D)),
  retractall(OPÇÃO(_)),
  not(hipotese(_)),
     relacao("PARTE DE",D,H),
     asserta(hipotese(H)),
     fail.

/* Mostre as hipoteses do diferencial */
decompoe :-
  pula_comentario,
  comentario("HIPÓTESES POSSIVEIS: "," "),
  hipotese(H),
  comentario(H,"    "), fail.

decompoe :- !.
```

A definição dessa Fonte de Conhecimento é composta das seguintes partes:

1) É responsável por limpar a tela.

2) Verifica se a discrepância é uma parte não decomponível do equipamento caso em que nenhuma hipótese de falha é gerada.

3) Assume, provisoriamente, como hipóteses plausíveis de causa da falha, todas as subpartes da parte do equipamento aonde foi observada a falha (ou seja a discrepância).

4) Na parte 4 são determinados os sintomas que possam ser úteis para determinar em que subparte da parte discrepante está a falha.

5) Na parte 5, o usuário escolhe os sintomas que ele está observando e, com base nessa informação, são eliminadas as hipóteses de causa de falha geradas na parte 3 que sejam incompatíveis com as observações.

6) Se a Tabela de Falhas estiver incompleta, pode ser que nenhuma hipótese de falha seja compatível com as observações feitas. Nesse caso nenhuma hipótese restará após a parte 5. Se isso ocorrer, serão assumidas como hipóteses de falha viáveis todas as subpartes da parte discrepante.

7 e 8) Essas partes da definição de DECOMPÕE se encarregam de apresentar ao usuário quais são as hipóteses de falha plausíveis.

A Fonte de Conhecimento ORDENA recebe as hipóteses de falha plausíveis e gera um conjunto de hipóteses de falha ordenados em ordem crescente de dificuldade de teste.

```
/* ORDENA */
ORDENA :-
    clearwindow,
    fail.
ordena :-
     retractall(hipotese_ordenada(_)),
     not(hipotese(_)), !.
ordena :-
      repita,
```

```
     hipotese(H),
     custo(H,C),
     not(hipotese_mais_facil(H,C)),
     assert(hipotese_ordenada(H)),
     retract(hipotese(H)),
     not(hipotese(_)), !.
```

A Fonte de Conhecimento SELECIONA testa a hipótese de falha de menor custo e gera o comportamento observado a partir do teste.

```
/* SELECIONA */
SELECIONA :-
    clearwindow,
    fail.
seleciona :-
     hipotese_ordenada(H),
     testa_hipotese(H,A,Valor),

     asserta(comportamento(H,A,Valor)).
```

CALCULA determina o comportamento esperado do sistema e o compara com o comportamento observado. Se eles forem diferentes a parte do equipamento que é a hipótese corrente de falha é considerada discrepante. Caso o comportamento observado seja o comportamento esperado, e caso não hajam mais testes para a parte do equipamento que é a hipótese corrente de falha, essa hipótese é descartada.

```
/* CALCULA */
CALCULA :-
    clearwindow,
    fail.
calcula :-
  hipotese_ordenada(H),
  retract(comportamento(H,Atributo,OB)),
  esperado(H,Atributo,EB),
  EB <> OB,
  asserta(discrepancia(H)), !.
calcula :-
  hipotese_ordenada(H),
  not(tst(_,H,_,_,_,_)),
  retract(hipotese_ordenada(H)), !.
```

Uma vez tendo sido determinada a explicação da falha, pela Fonte de Conhecimento EXPLICA, é apresentado ao usuário um conselho (caso ele tenha sido definido) e um procedimento de reparo adequado. Se nenhum reparo tiver sido definido para a explicação a mensagem "Não há nenhum reparo conhecido !!!" será apresentada.

```
/* PRESCREVE */
PRESCREVE :-
    clearwindow,
    fail.
prescreve :-
     explicação(E),
     conselho(E,Conselho),
     avisa(Conselho), nl,
     avisa("Para consertar o equipamento faça o seguinte: "),
     rpr(_,E,_,Texto),
     avisa(Texto), !.
prescreve :-
     explicação(E),
     not(conselho(E,_)),
     avisa("Para consertar o equipamento faça o seguinte: "),
     rpr(_,E,_,Texto),
     avisa(Texto), !.
prescreve :-
     avisa("Não há nenhum reparo conhecido !!!"), !.
```

c) Tarefa. A descrição da tarefa deve ser feita de acordo com uma sintaxe "procedimental". O KADS sugere as primitivas que se deve usar para tal fim porém não prescreve uma sintaxe rigorosa para tal fim. Na nossa ferramenta a descrição da tarefa deve ser feita por meio de uma linguagem cuja sintaxe é apresentada a seguir na notação de Wirth iterativa:

```
tarefa = {comandos}".".

comando = [].
comando = "ENQUANTO" condição "FACA" {comando} "FIM" ";".
comando = "REPITA" {comando} "ATE" condição ";".
comando = "SE" condição "FACA" {comando} "FIM" ";".
comando = "NOME DE UMA FONTE CONHECIMENTO" ";".
```

```
comando = "NOME DE UMA TAREFA" ";".

condição = ("NAO"| []) "EXISTIR" "(" "NOME DE UMA METACLASSE" ")"
                                        ("E" condição | []).
```

A seguir é apresentado, como exemplo, a declaração da tarefa de diagnose.

```
explica;
prescreve;
identifica;
repita
  decompoe;
  ordena;
  enquanto existir (hipotese ordenada) e nao existir(discrepancia)
      faca
        seleciona;
        calcula;
        fim;
  ate nao existir(hipotese ordenada);  .
```

As tarefas, declaradas de acordo com a notação previamente explicitada, são analisadas sintaticamente e convertidas por um gerador de código para um conjunto de predicados Prolog. Caso algum erro seja encontrado durante a análise sintática, a tarefa é apresentada e o cursor é posicionado no local aonde o erro ocorreu. Nesse caso, na última linha da tela, será apresentada a mensagem: "ERA ESPERADO: 'comando correto' AO INVÉS DE: 'comando incorreto'".

O nome de um predicado associado a uma tarefa começa sempre com "tarefa_de_" seguido pelo nome da tarefa, isso é feito para que não haja possibilidade de uma tarefa ser confundida com uma Fonte de Conhecimento ou Metaclasse. Em geral uma tarefa é descrita por vários predicados. Isso ocorre para tratar blocos do tipo "REPITA" ou "ENQUANTO". Cada um desses blocos é decomposto num predicado distinto. Um bloco REPITA dá origem a um predicado e um bloco ENQUANTO dá origem a dois predicados adicionais, como pode ser

observado no código gerado a partir da descrição "default" da tarefa, apresentado a seguir.

```
/*      TAREFA DIAGNOSE      */
tarefa_de_DIAGNOSE(0) :-
  identifica,
  tarefa_de_DIAGNOSE(1),
  !.

tarefa_de_DIAGNOSE(1) :-
  repit,
  decompoe,
  ordena,
  tarefa_de_DIAGNOSE(3),
  not(hipotese_ordenada(_)),
  !.

tarefa_de_DIAGNOSE(3) :-
  tarefa_de_DIAGNOSE(2),
  repit,
  seleciona,
  calcula,
  not(tarefa_de_DIAGNOSE(2)),
  !.
tarefa_de_DIAGNOSE(3).

tarefa_de_DIAGNOSE(2) :-
  hipotese_ordenada(_),
  not(discrepancia(_)),
  !.
```

5.2) "EXECUTAR" ("executar sistema especialista").

Essa opção permite que se chame o shell de diagnose associado à base de dados atual. Para tanto essa base de dados deve ter sido previamente compilada, com o auxílio da ferramenta CAKE, para a forma de um programa PROLOG e esse programa deve ter sido compilado, com o compilador do TURBO PROLOG, para um programa executável. Para compilar o programa PROLOG gerado pela ferramenta CAKE é necessários que os seguintes arquivos estejam no diretório: "tdoms.pro", "tpreds.pro" e "menu.pro".

Ao executar o SE de diagnose duas janelas serão criadas na tela. A primeira é a principal aonde é feita a entrada de dados por meio de menus, e a segunda é a janela de comentários, usada pelo sistema para informar o usuário sobre a causa e a explicação da falha e apresentar as hipóteses de falha corrente.

Inicialmente o sistema solicita que o usuário indique qual é o equipamento que será diagnosticado. Em seguida são apresentados um conjunto de sintomas dentre os quais o usuário deve escolher aqueles que estiver observando. São então geradas hipóteses de falha plausíveis que serão mostradas ao usuário na tela de comentários. Vários testes são propostos e, eventualmente, o usuário é solicitado a escolher novamente os sintomas observados. Isso prossegue até que se determine a causa da falha do equipamento. O passo seguinte é determinar uma explicação para essa falha. Para tanto um novo conjunto de testes poderá, se necessário, ser proposto. Uma vez localizada a explicação da falha, um reparo apropriado é determinado e apresentado ao usuário do sistema.

6) "FIM".

Opção que permite que se saia do CAKE definitivamente e se retorne ao sistema operacional. Caso a base de dados tenha sido alterada, e as alterações não tenham sido salvas, será apresentada a seguinte mensagem: "SALVA BASE DE CONHECIMENTO <S/N> ?".