Ambiente de Desenvolvimento de Sistemas Baseados em Conhecimento

Vera Maria Benjamim Werneck

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO EM ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS PARA OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Profª. Ana Regina Cavalcanti da Rocha, DSc
(Presidente)

Prof. Álvaro Rabelo Alves Junior, LD

Profª. Claudia Maria Lima Werner, DSc

Prof. Crediné Silva de Menezes, DSc

Prof. Guilherme Horta Travassos, DSc

Profª. Sueli Bandeira Teixeira Mendes, PhD

RIO DE JANEIRO, RJ - BRASIL
JUNHO 1995

Werneck, Vera Maria Benjamim

Ambiente de Desenvolvimento de Sistemas Baseados em Conhecimento (Rio de Janeiro 1995)

x,239 p. 29,7 cm (COPPE/UFRJ, Dsc, Engenharia de Sistemas e Computação, 1995)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Engenharia de Software 2. Sistemas Baseados em Conhecimento

I. COPPE/UFRJ II Título (série)

Aos meus pais,

ao meu amor e

à minha filha Carolina

AGRADECIMENTOS

À Professora Ana Regina, pela orientação, dedicação, amizade e competência que tanto me incentivou, estimulando para o caminho acadêmico, transformando este trabalho num grande aprendizado. Por sua confiança e especialmente pelo carinho, sendo não só a orientadora, mas uma amiga muito especial.

Ao Professor Dr. Álvaro, pela acolhida no projeto, pela confiança, e, não só por permitir, mas principalmente por participar ativamente para o desenvolvimento deste trabalho, que sem o seu apoio não seria o mesmo.

Ao Professor Guilherme, pela valiosa contribuição para a conclusão deste trabalho e por dar a honra de participar da minha banca.

Ao Professor Jano pelo apoio constante ao longo deste anos, contribuindo tanto para o meu ingresso quanto para a conclusão deste doutorado.

À Professora Sueli, por seus incentivos, sempre disposta a conversar, esclarecer minhas dúvidas e me oferecer um pouco do seu conhecimento e também por participar da minha banca.

À Professora Claudia pelas sugestões e incentivo em momentos decisivos deste trabalho e por participar da minha banca.

Ao Professor Crediné por aceitar participar da minha banca.

Aos Professores Mario e Luis Alfredo por participarem como orientadores dos meus exames de qualificação e por me incentivarem na realização deste trabalho.

À Kathia que tanto se empenhou e contribuiu de forma significativa para o desenvolvimento deste trabalho, com sua alegria, competência, esforço e perseverança.

Ao Dr. Aguinaldo, Dr. Nelson, Dr. Ximenes, Dr. Luiz Agnaldo, Werther, Dulcineia e Mário pela confiança, disponibilidade e sugestões realizadas ao longo do desenvolvimento do SEC e deste trabalho e também pelo clima de cooperação e amizade instalado na equipe, que são preponderantes para a realização e concretização de um bom trabalho.

À Cristina e ao Blashek pela amizade e pela disponibilidade para discutir as minhas idéias, ler meus trabalhos e compartilhar meus sentimentos.

À Neide e à Gilda pelas palavras sempre amigas e incentivadoras

Ao Ismael, Freedman, Arlindo e Clóvis pelo constante incentivo e pela minha aproximação com a IBM, permitindo que este trabalho fosse concluído mais rapidamente.

Aos meus amigos da COPPE e FBC, Márcio, Eliseu, Glória, Rosa, Marimar, Xexeo, Gisela, Clifton, Cleuza, Eduardo, Marta, Leonardo, Fernanda, Clicia, Paula, Márcia, Washington, Claudia Gama, Glorinha, pelo apoio e compreensão, tornando este trabalho mais fácil de ser realizado.

Ao Belchior, Renata e Francisco pela disponibilidade em utilizar o método KADS-estendido e pelas sugestões feitas.

Ao Ronald Schrostein pelas informações sobre o KADS e envio de artigos tão importantes para este trabalho.

Ao Julinho, Adilson, Frederico e Carlos, pelo apoio, paciência e ajuda constante no Laboratório.

À Claudinha, Rose, Mercedes, Ari e às duas Ana Paula cujo apoio foram imprescindíveis na realização deste trabalho.

Ao Gustavo pela contribuição, disponibilidade e entusiasmo na construção do protótipo.

Aos professores e amigos do Programa de Sistemas da COPPE pelo apoio dado a este trabalho, bem como pela amizade demonstrada nesses anos .

Ao Jorge, meu marido, pelo seu companheirismo, sempre me incentivando e ajudando com seu amor, compreensão, amizade e trabalho.

À minha mãe que sempre fez tudo para me oferecer o melhor, e ao meu pai que me ajudou tanto no desenvolvimento do sistema CRIANDO e também me mostrando que um bom especialista é aquele que gosta do que faz.

À Maria Carolina que com sua alegria, serenidade e saúde permitiram que este trabalho fosse desenvolvido com paz e tranquilidade.

À tia Anita e Leopoldo pela disponibilidade em revisar meus textos.

À IBM-Brasil e CAPES pelo apoio financeiro.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências (D. Sc.)

AMBIENTE DE DESENVOLVIMENTO DE SISTEMAS BASEADOS EM CONHECIMENTO

Vera Maria Benjamim Werneck

Março de 1995

Orientadora: Ana Regina Cavalcanti da Rocha, D. Sc.
Programa: Engenharia de Sistemas e Computação

Este trabalho define um processo adequado ao desenvolvimento de sistemas baseados em conhecimento contendo uma extensão ao método KADS e propõe um ambiente de desenvolvimento de software baseado neste processo.

O processo de desenvolvimento para sistemas baseados em conhecimento, foi definido a partir das normas ISO, contemplando as atividades relativas à construção e avaliação da qualidade do produto e à gerência do projeto. O processo foi definido e validado através da experiência no desenvolvimento de vários SBCs sendo, entre estas, a mais significativa a experiência no projeto *Sistemas Especialistas em Cardiologia*, da Fundação Bahiana de Cardiologia.

O ambiente de desenvolvimento de software foi definido, no contexto do Projeto TABA da COPPE/UFRJ, a partir do processo estabelecido. O ambiente ORIXÁS é, portanto, um ambiente instanciado TABA, adequado ao desenvolvimento de sistemas baseados em conhecimento.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for degree of Doctor of Sciences (D.Sc.)

KNOWLEDGE BASED SYSTEMS DEVELOPMENT ENVIRONMENT

Vera Maria Benjamim Werneck

March, 1995

Thesis supervisor: Ana Regina Cavalcanti da Rocha, D. Sc.

Department: Engenharia de Sistemas e Computação

The present work defines an adequate knowledge based systems development process containing an extension of the KADS method and also defines a software development environment based on this process.

The knowledge based systems development process has been defined based on the ISO norms, regarding the activities related to the construction and quality evaluation of the product and to the management of the project. The process has been defined and validated by the experience in the development of various Knowledge Based Systems, being the most significant amongst them the experience in the project Expert Systems in Cardiology of the Bahian Cardiology Foundation.

The software development environment has been defined in the context of the TABA Project of COPPE/UFRJ based on the established process. The ORIXÁS environment is therefore a TABA environment instance adequate for the development of knowledge based systems.

ÍNDICE

Capítulo I

Introdução

I.1 Sistemas Baseados em Conhecimento e seu Processo de Desenvolvimento

Sistemas Baseados em Conhecimento (SBC) é a área da Inteligência Artificial que enfatiza o conhecimento específico sobre um determinado domínio de problema. Neste ponto se diferencia de outras áreas da Inteligência Artificial, como a robótica, a visão artificial e o reconhecimento de fala (Hembry, 1990).

O desenvolvimento dos SBCs começou a partir dos anos 60 em universidades. A partir de 1980, ocorreu um crescimento significativo do número de aplicações construídas usando esta tecnologia. Atualmente vários sistemas baseados em conhecimento estão em uso, principalmente nas áreas de engenharia, ciências, medicina, militar e financeira. Entretanto, as promessas com relação a área foram exageradas. Muitos sistemas desenvolvidos com o objetivo de serem especialistas, de fato não agiam como tal. Por isso é, ainda, limitada a aceitação da tecnologia baseada em conhecimento.

Uma questão fundamental a ser considerada é a dos procedimentos para a construção dos SBCs. Inicialmente, o que é característico da etapa de introdução de uma nova tecnologia, o desenvolvimento de SBCs se fazia de forma bastante informal: "*run-understand-debug-edit*" (Bader et alii, 1988). Hoje, este procedimento já não é aceitável e buscam-se processos de desenvolvimento adequados que garantam uma maior qualidade para os SBCs.

Construir software de qualidade é uma meta perseguida por grupos de desenvolvimento de software, nem sempre, entretanto, de forma sistemática e produtiva.

Recente pesquisa, realizada com o objetivo de fazer um levantamento da qualidade no setor de software em empresas brasileiras (Weber et alii, 1994) evidenciou que a maior parte das empresas ainda não apresenta maturidade no tratamento da qualidade. Poucas empresas (26%) medem a qualidade de seus produtos não tendo, em geral, uma cultura voltada para a gestão da qualidade (cerca de 50% das empresas afirmaram não possuir um sistema da qualidade para o desenvolvimento).

Quanto ao uso de métodos, ferramentas e procedimentos de Engenharia de Software foi possível observar, nesta pesquisa, o pouco uso de técnicas para revisão de software e de ferramentas CASE. A produção de documentação insuficiente e de forma não sistemática mostrou-se, também, uma constante. Apenas 50% das empresas entrevistadas indicaram produzir especificações e, apenas, 30% afirmaram que fazem documentação de código. Torna-se necessário, portanto, estabelecer procedimentos capazes de reverter esta situação.

Outra pesquisa, também recente, realizada no contexto do Projeto Esprit (Bazzana, 1993) evidenciou a profunda relação entre a qualidade do produto (software) e a qualidade do processo de desenvolvimento. Assim sendo, para estabelecer um programa de garantia da qualidade de software é necessário estabelecer procedimentos para garantia da qualidade a nível do produto e a nível do processo de desenvolvimento. Neste contexto se situam as normas ISO 9000-3 (ISO, 1990), que trata da qualidade a nível do processo e ISO 9126 (ISO/IEC, 1991), que trata da qualidade a nível do produto. Atualmente, uma empresa de desenvolvimento de software, para ser competitiva no mercado internacional deve estar certificada segundo a norma ISO 9000-3 garantindo, assim, a outras empresas contratantes de seus serviços a qualidade do processo de desenvolvimento que utiliza.

Outra abordagem para esta questão é a do *Software Engineering Institute* (SEI), nos Estados Unidos, que define níveis de maturidade para empresas, dependendo da maturidade do processo que esta utiliza para desenvolvimento de seus produtos de software (Arthur, 1993). São definidos cinco níveis: inicial, repetível, definido, gerenciado e otimizado. O nível 1 é identificado com o caos no desenvolvimento de software e uma pesquisa realizada pela SEI em 1991 identificou que 88% das empresas encontravam-se neste estágio. No nível 2, projetos podem ser entregues no cronograma e com um nível razoável de qualidade. No nível 3, o processo de software é definido, treinado e seguido pelos engenheiros de software. A partir desta definição as empresas podem atingir o próximo nível, onde o processo e seus produtos começam a ser medidos

e analisados. No nível mais alto, todos os engenheiros de software aplicam as ferramentas e princípios de Gerência da Qualidade Total, para aperfeiçoar o processo e seus produtos.

De acordo com os princípios estabelecidos na ISO 9000-3, a primeira atividade ao se iniciar um projeto deve ser a elaboração do documento *Processo de Desenvolvimento* definindo o modelo de ciclo de vida a ser adotado, os métodos e ferramentas para construção do produto, a documentação a ser produzida, os procedimentos e métodos para controle da qualidade e gerência.

Entretanto, a definição do processo de desenvolvimento e avaliação da qualidade não pode ser realizada de forma universal. O processo de desenvolvimento, para ser eficaz e conduzir à construção de produtos de boa qualidade, deve ser adequado ao domínio de aplicação e ao projeto específico. Assim sendo, deve ser definido caso a caso, considerando-se as especificidades da aplicação, da tecnologia a ser adotada na construção do produto, da organização, do grupo de desenvolvimento e dos futuros usuários do produto (Rocha et alii, 1994).

I.2. Objetivo da Tese

Este trabalho tem como objetivo definir um processo adequado ao desenvolvimento de sistemas baseados em conhecimento e propor um ambiente de desenvolvimento de software baseado neste processo.

O processo de desenvolvimento para sistemas baseados em conhecimento foi definido a partir das normas ISO 9000-3 e ISO 9126, contemplando as atividades relativas à construção e avaliação da qualidade do produto e as atividades relativas à gerência do projeto. O processo foi definido e validado através da experiência no desenvolvimento de vários SBCs sendo, entre estas, a mais significativa a experiência no projeto *Sistemas Especialistas em Cardiologia*, da Unidade de Cardiologia e Cirurgia Cardiovascular do Hospital Universitário Prof. Edgard Santos da Universidade Federal da Bahia/Fundação Bahiana de Cardiologia (UCCV/FBC).

O ambiente de desenvolvimento de software foi definido, no contexto do Projeto TABA da COPPE/UFRJ, a partir do processo estabelecido.

Finalmente, este trabalho se complementa com outro realizado no mesmo contexto, e que aborda com detalhes a questão da avaliação da qualidade dos SBCs (Oliveira, 1995).

I.3. As Normas ISO 9000-3 e ISO 9126

As normas ISO 9000-3 e ISO 9126 cobrem os dois aspectos que devem ser considerados ao se buscar o desenvolvimento de software de qualidade: qualidade a nível do processo e qualidade a nível do produto.

A norma ISO 9000-3 faz parte da série ISO 9000, que contém as seguintes normas:

- ISO 9000, que fornece diretrizes gerais para sistemas de qualidade;
- ISO 9001, que trata do projeto, desenvolvimento, produção, instalação e serviço;
- ISO 9002, que trata da produção e instalação;
- ISO 9003, que trata da instalação e teste;
- ISO 9004, que fornece diretrizes para a gerência da qualidade.

A norma ISO 9000-3, definida em 1990, significa um reconhecimento de que o processo de desenvolvimento e manutenção de software é diferente do de outros produtos industriais. Esta norma foi definida com o objetivo de facilitar a aplicação da norma ISO 9001 ao contexto de software. Possui três partes:

- *Estrutura*, que descreve os aspectos organizacionais a serem considerados na produção de software (responsabilidade gerencial, sistema da qualidade, auditoria interna do sistema da qualidade, ação corretiva);
- *Atividades do ciclo de vida*, que define as ações necessárias para conduzir o desenvolvimento (revisão do contrato, especificação dos requisitos do cliente, planejamento do desenvolvimento, planejamento da qualidade, projeto e implementação, testes e validação, aceitação, reprodução, entrega e instalação, manutenção);

- *Atividades de apoio*, que define as atividades de suporte à produção, entrega e manutenção de software (gerência de configuração, controle da documentação, registros da qualidade, medições, regras práticas e convenções, ferramentas e técnicas, compras, software produto incluso e treinamento).

A norma ISO/IEC 9126 foi publicada em 1991 e define seis características de qualidade de software: funcionalidade, confiabilidade, utilizabilidade, eficiência, manutenibilidade e portatilidade. A norma, entretanto, não chega a estabelecer métricas que possibilitem a medição do software. Para suprir esta carência está sendo elaborado, pelo grupo de trabalho ISO/IEC JTC1/SC7/WG6, um conjunto de guias de auxílio à aplicação da ISO 9126.

I.4. O Projeto Sistemas Especialistas em Cardiologia

Este trabalho faz parte do projeto Sistemas Especialistas em Cardiologia (SEC), em desenvolvimento na UCCV/FBC. Este projeto é financiado pela FINEP (Convênio nº 66940058-00) e conta, também, com apoio da IBM-Brasil.

O projeto tem como um de seus objetivos construir um sistema especialista para apoio a médicos não cardiologistas no diagnóstico de eventos coronarianos agudos, o SEC-Diagnóstico. Este projeto está em desenvolvimento desde o início de 1994.

A área de Engenharia de Software da COPPE/UFRJ participa do projeto, colaborando com a UCCV/FBC, em aspectos relativos ao processo de desenvolvimento e à avaliação da qualidade do produto. É neste contexto que se insere este trabalho. O processo de desenvolvimento proposto, nesta tese, foi definido, experimentado, reformulado e validado no âmbito do projeto SEC-Diagnóstico, ao longo da construção das versões 1.0 e 2.0 e de seus refinamentos.

I.5. O Projeto TABA

Este trabalho está inserido, também, no contexto do Projeto TABA em desenvolvimento no Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ. Este projeto foi financiado pela FAPERJ, como projeto especial, de 1989 a 1993 e é atualmente apoiado pelo CNPq e pela reitoria da UFRJ.

O objetivo do projeto TABA é a construção de uma Estação de Trabalho para desenvolvimento de software, configurável para atender às características dos diferentes domínios de aplicação. A escolha de ambientes de desenvolvimento de software (ADS) deve considerar as características dos domínios de aplicação e da tecnologia a ser adotada na construção do produto. Não se pode esperar que um ADS, mesmo completo e de boa qualidade, seja adequado a qualquer área de aplicação e a qualquer projeto.

O projeto TABA visa, portanto, construir uma Estação de Trabalho para o engenheiro de software, que permita a definição e implementação de ambientes adequados ao desenvolvimento de software em diferentes domínios de aplicação e utilizando diversas tecnologias de desenvolvimento (Rocha e Souza, 1988), (Rocha et alii, 1990), (Werneck, 1990), (Aguiar, 1992), (Travassos, 1994).

Para atingir seu objetivo a Estação TABA contém um meta-ambiente com as funções de especificar o ambiente de desenvolvimento de software mais adequado a um projeto, instanciar e tornar operacional o ambiente especificado. A especificação de ambientes é feita através de XAMÃ (Aguiar, 1992), (Massolar, 1993), um assistente baseado em conhecimento que apoia o engenheiro de software nesta tarefa. O trabalho realizado por Travassos (1994) criou um modelo canônico para representação de informações em ambientes, tornando operacional a instanciação dos ambientes especificados por XAMÃ através de um modelo para integração e construção de ferramentas a partir da utilização de *frameworks*.

Este trabalho parte de XAMÃ e do trabalho realizado por Travassos para definir um ambiente de desenvolvimento para sistemas baseados em conhecimento. Este ambiente é, portanto, um ambiente instanciado TABA.

I.6. Organização da Tese

Este trabalho, além desta Introdução, contêm seis capítulos e 6 anexos.

No capítulo II - *Sistemas Baseados em Conhecimento* - são descritas as características dos sistemas baseados em conhecimento, sua arquitetura, evolução, aplicabilidade e tendências atuais.

No capítulo III - *Engenharia de Software no Desenvolvimento de Sistemas Baseados em Conhecimento* - é descrito o estado da arte relativo à Engenharia de Software para sistemas baseados em conhecimento. São abordados aspectos relativos ao

processo de desenvolvimento e aos métodos e ferramentas de desenvolvimento, avaliação, gerência e manutenção propostos na literatura.

No capítulo IV - *Definição de um Processo de Desenvolvimento para Sistemas Baseados em Conhecimento* - é relatada a evolução deste trabalho no que se refere ao processo de desenvolvimento. São descritas as experiências realizadas e a definição final do processo estabelecendo-se o modelo de ciclo de vida e os métodos adequados a cada atividade do desenvolvimento.

No capítulo V - *O Método de Desesenvolvimento KADS-estendido* - descreve-se uma extensão ao método KADS, definida a partir da avaliação realizada após o seu uso em vários projetos.

No capítulo VI - *Um Ambiente para Desenvolvimento de Sistemas Baseados em Conhecimento* - são descritos o ambiente proposto, para apoio ao desenvolvimento de SBCs segundo o processo definido no capítulo IV e no método KADS-estendido, e a implementação do protótipo deste ambiente no contexto da Estação TABA..

O capítulo VII - *Conclusão e Perspectivas Futuras* - contém as conclusões do trabalho, destacando-se suas contribuições e as pesquisas que devem dar continuidade ao mesmo.

O Anexo I contém os formulários para avaliação do processo de desenvolvimento utilizado no projeto Sistemas Especialistas em Cardiologia.

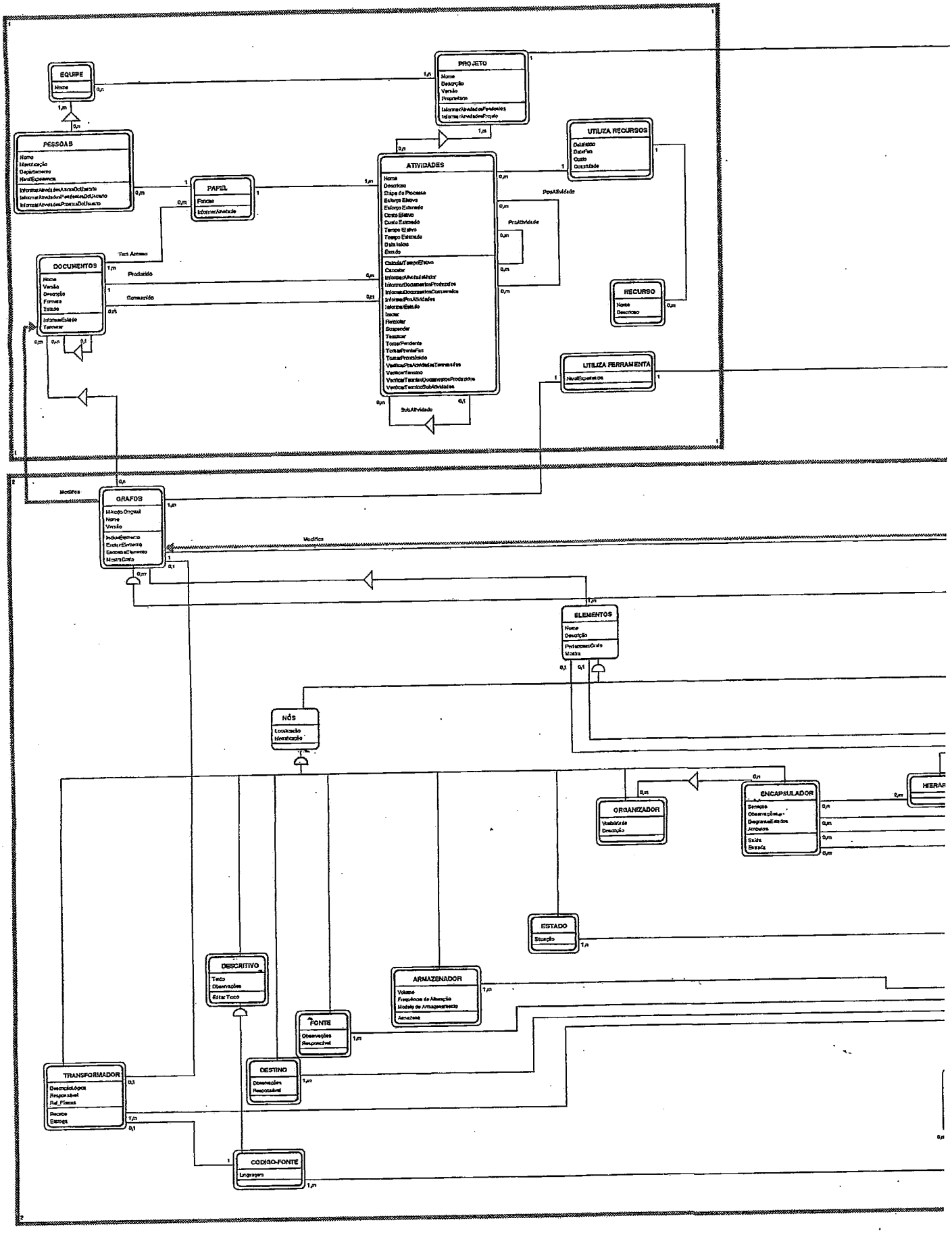
O Anexo II contém o formulário para avaliação e seleção de ambientes de programação para sistemas baseados em conhecimento, utilizado no projeto Sistemas Especialistas em Cardiologia.

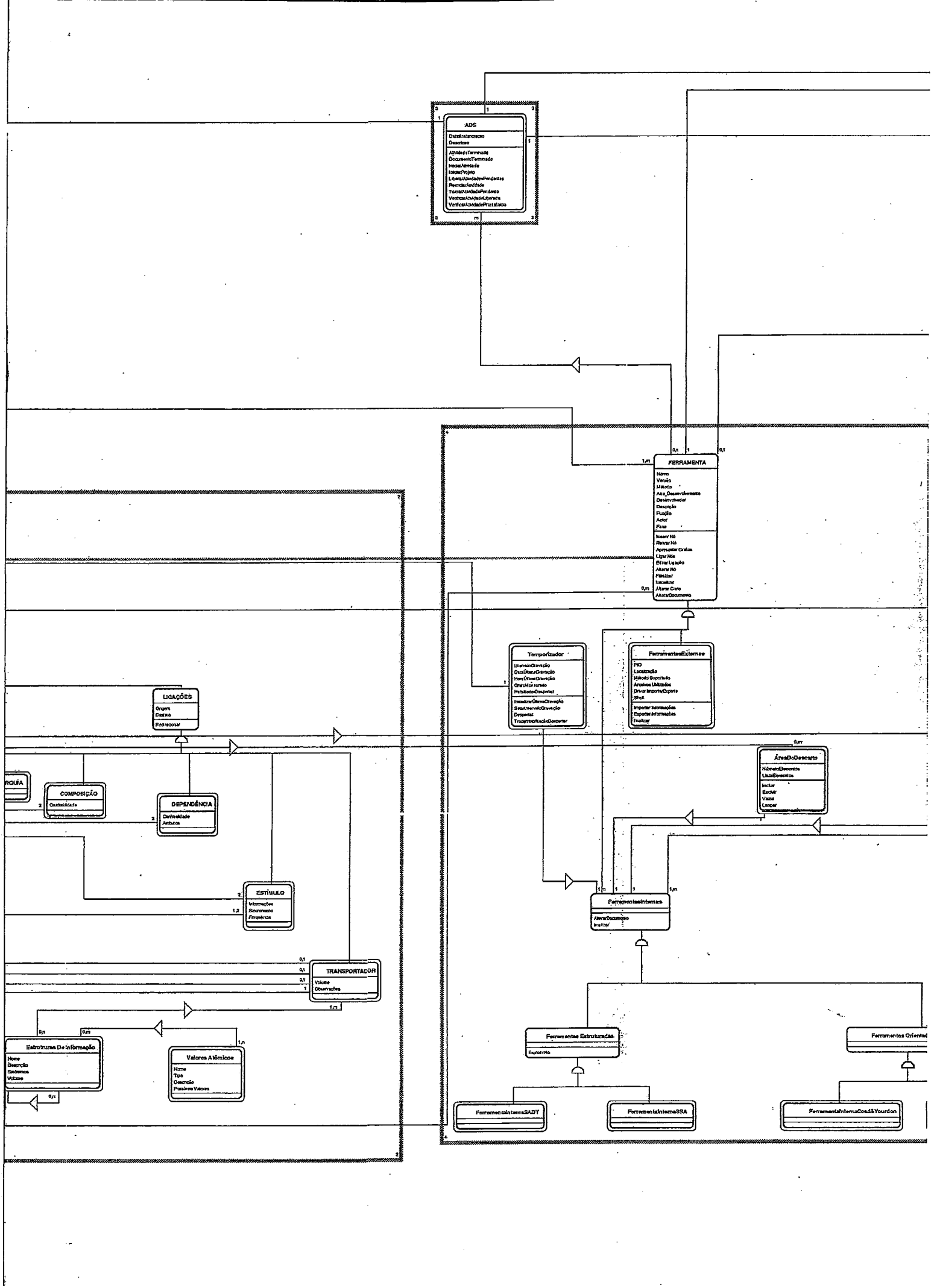
O Anexo III contém os diagramas da modelagem da Estação TABA.

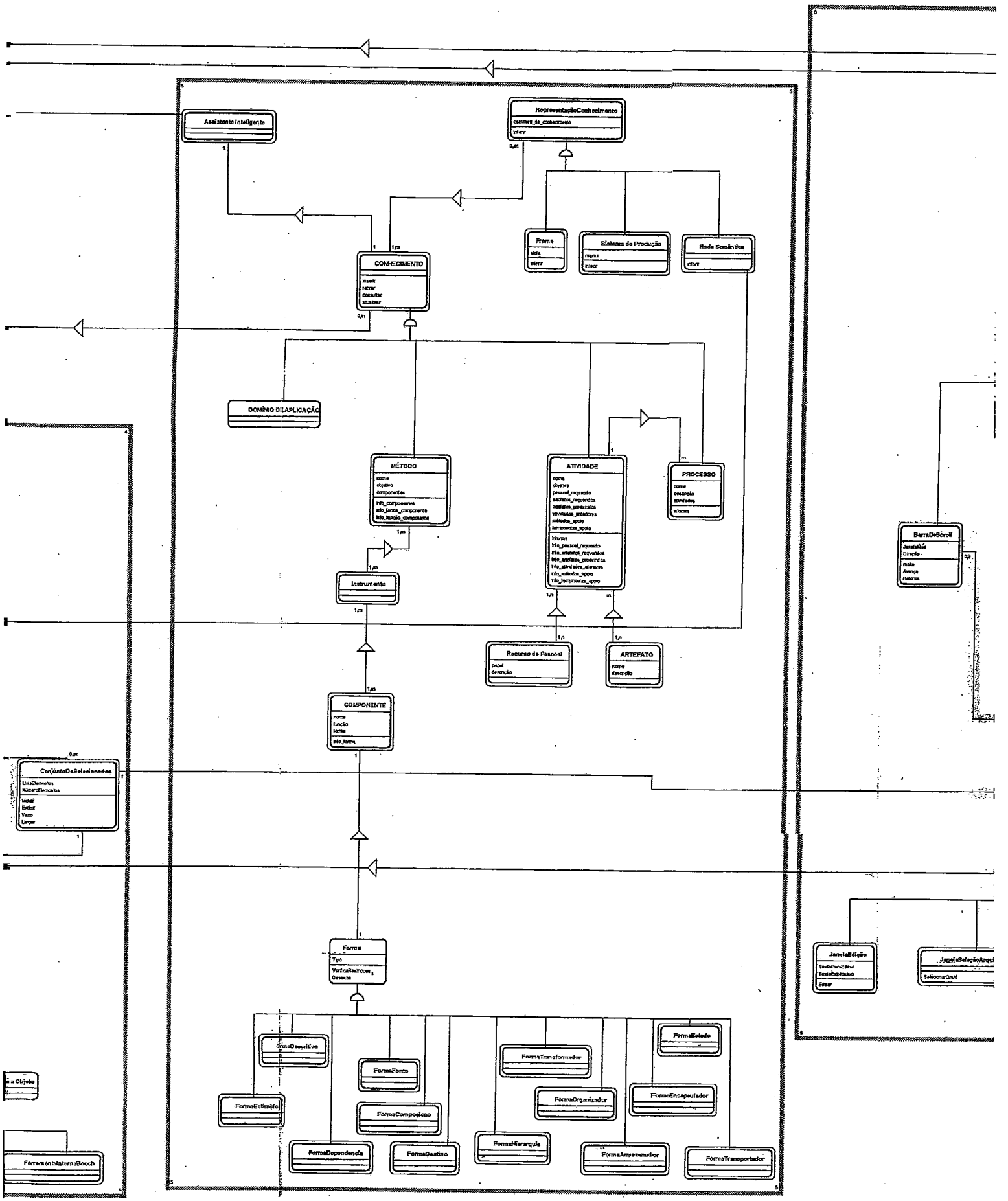
O Anexo IV contém a definição e extensão da Linguagem para Descrição e Manipulação de Métodos definida no contexto da Estação TABA e exemplos dos diagramas do método KADS-estendido nesta linguagem.

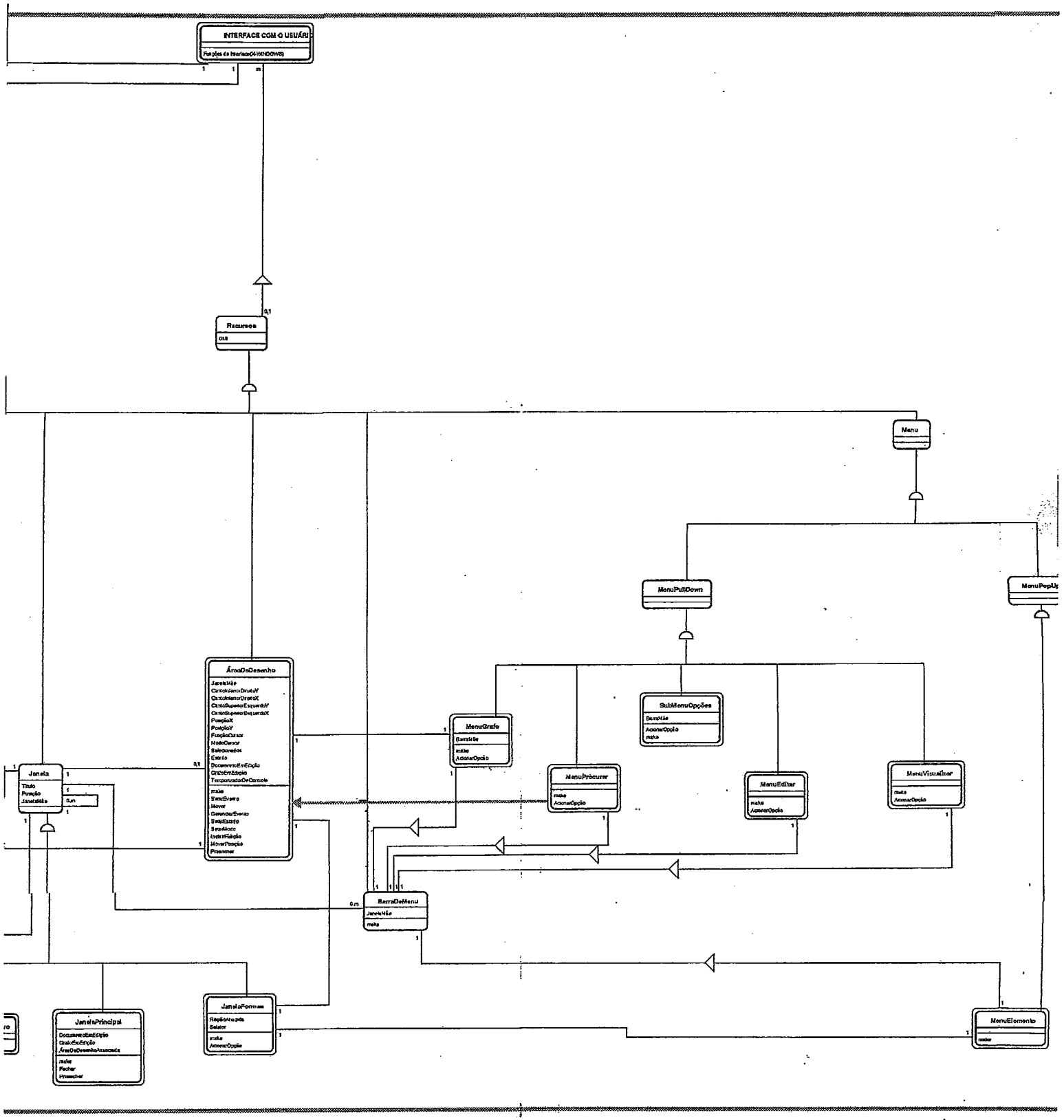
O Anexo V contém os diagramas da modelagem da Estação TABA especializados .para o ambiente de desenvolvimento proposto.

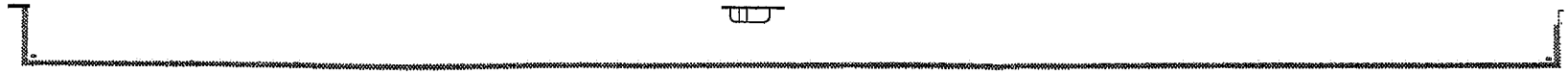
O Anexo VI contém a descrição do Modelo de Especialidade do método KADS.


EQUIPE
Nome
PROJETO
Nome
Descrição
Versão
Proprietário
PESSOAS
Nome
Identificação
Departamento
NívelExperiência
PAPEL
Função
ATIVIDADES
Nome
Descrição
Etapa do Processo
Esforço Efetivo
Esforço Estimado
Custo Efetivo
Custo Estimado
Tempo Efetivo
Tempo Estimado
Data Início
Estado
UTILIZA RECURSOS
DataIníc
DataFim
Custo
Quantidade
RECURSO
Nome
Descrição
DOCUMENTOS
Nome
Versão
Descrição
Formato
Estado
UTILIZA FERRAMENTA
NívelExperiência
GRAFOS
Método Original
Nome
Versão
ELEMENTOS
Nome
Descrição
NÓS
Localização
Identificação
ORGANIZADOR
Visibilidade
Descrição
ENCAPSULADOR
Semente
ESTADO
Situação
DESCRITIVO
Texto
Observações
ARMAZENADOR
Volume
Frequência de Alteração
Modelo de Armazenamento
Armazena
FONTE
Observações
Responsável
DESTINO
Observações
Responsável
TRANSFORMADOR
Descrição/Lógica
Responsável
Ref_Físicas
Recebe
Entrega
CÓDIGO-FONTE
Linguagem


ADS
DataInstanciacao
Descricao
AtividadeTerminada
OcorrenciaTerminada
InicioAtividade
InicioProjeto
LiberarAtividadesPendentes
PeriodoAtividade
TornarAtividadePendente
VerificarAtividadeLiberada
VerificarAtividadeProntaInicio
FERRAMENTA
Nome
Versão
Método
Ano_Desenvolvimento
Desenvolvedor
Descrição
Função
Autor
Fase
Inserir Nó
Retirar Nó
Apresentar Gráfico
Unir Nós
Extrair Ligação
Alterar Nó
Finalizar
Iniciar
Alterar Cena
AlterarDocumento
Temporizador
UltimaGravação
DataÚltimaGravação
HoraÚltimaGravação
GravaNoInstante
HabilitarDisparar
IncializarÚltimaGravação
SimultaneidadeGravação
Disparar
TrocarHabilitaçãoDisparar
FerramentasExternas
PID
Localização
Método Disparado
Arquivos Utilizados
Drive Importe/Exporte
Shell
Importar Informações
Exportar Informações
Inicializar
ÁreaDeDescarte
NúmeroElementos
ListaDeposição
Incluir
Excluir
Vazar
Limpar
FerramentasInternas
AlterarDocumento
Iniciar
LIGAÇÕES
Origem
Destino
Redirecionar
COMPOSIÇÃO
Cardinalidade
DEPENDÊNCIA
Cardinalidade
Atributos
ESTÍMULO
Informações
Sincronismo
Frequência
TRANSPORTADOR
Volume
Observações
Estruturas De Informação
Nome
Descrição
Sinônimos
Volume
Valores Atômicos
Nome
Tipo
Descrição
Possíveis Valores
Ferramentas Estruturadas
Exploratória
FerramentaInternaSADT
FerramentaInternaSSA
FerramentaInternaCoad&Yourdon


Assistente Inteligente
RepresentaçãoConhecimento
CONHECIMENTO
Frame
Sistema de Produção
Rede Semântica
DOMÍNIO DE APLICAÇÃO
MÉTODO
ATIVIDADE
PROCESSO
Instrumento
Recurso de Pessoal
ARTEFATO
COMPONENTE
ConjuntoDeSelecionados
Forma
FormaDescritiva
FormaFonte
FormaTransformador
FormaEstado
FormaEstímulo
FormaComposição
FormaOrganizador
FormaEncapsulador
FormaDependência
FormaDestino
FormaHierarquia
FormaArmazenador
FormaTransportador
BarraDeScroll
JanelaEdição


INTERFACE COM O USUÁRIO
Funções de Interface(WINDOWS)
Recursos
GUI
Menu
MenuPullDown
MenuPopUp
ÁreaDeDesenho
MenuGrafo
SubMenuOpções
MenuProcurar
MenuEditar
MenuVisualizar
Janela
Título
Posição
JanelaMãe
Run
BarraDeMenu
JanelaPrincipal
JanelaFormas
MenuElemento



Capítulo II

Sistemas Baseados em Conhecimento

II.1. Introdução

Ao longo dos anos a ciência da computação tem sofrido várias transformações, evidenciadas pela introdução de novas tecnologias. As quatro primeiras gerações de computadores tiveram por base os processadores de informação. A quinta lançou o desafio do processamento do conhecimento. Os computadores que anteriormente só calculavam e armazenavam dados, agora devem raciocinar e informar (Assis, 1992).

A Inteligência Artificial é a área que estuda a aplicação da tecnologia hoje disponível, de modo a permitir aos computadores realizarem tarefas que no momento as pessoas efetuam com melhor desempenho. Assim possibilita ao computador ter um comportamento inteligente na realização de suas tarefas (Rich, 1985). Os programas de Inteligência Artificial exibem um comportamento inteligente através da aplicação de heurísticas, diferente do processo algorítmico utilizado pelos programas convencionais. Os sistemas baseados em conhecimento são aplicações que geram soluções qualitativas para problemas onde o computador necessita raciocinar com conhecimento (Haynes-Roth e Jacobstein, 1994).

O objetivo desse capítulo é caracterizar os sistemas baseados em conhecimento, identificando sua evolução, aplicabilidade, arquitetura e tendências atuais.

II.2. Características dos Sistemas Baseados em Conhecimento

Sistemas baseados em conhecimento (SBC) resolvem problemas usando o conhecimento já existente sobre um determinado domínio. A capacidade desses sistemas difere da usada pelo homem, pois a máquina realiza tarefas padronizadas. Ao resolver um problema de modo inteligente necessitam construir sua resposta de forma seletiva e efetiva dentro de um quadro de alternativas (Assis, 1992). Esses sistemas são, hoje, uma realidade e têm sido alvo de atenção no mercado comercial e industrial.

Inicialmente, essa área foi identificada por aplicações que adquiriam conhecimento de especialistas humanos e transmitiam o conhecimento especializado através do diálogo com o computador, que agia como um especialista. Sistemas especialistas são sistemas em que o conhecimento é adquirido de pessoas cuja habilidade em resolver problemas situa-se acima da habilidade usual e cuja especialidade no assunto é pouco comum. O termo sistemas especialistas, a principio, foi bastante utilizado o que acarretou muitas expectativas e discussões, pois nem sempre os sistemas desenvolvidos agiam como era esperado. O que ocorreu, na prática, é que os primeiros sistemas possuíam um percentual de acerto muito baixo, acarretando frustrações em relação à expectativa.

Neste trabalho os sistemas especialistas serão considerados segundo as classsificações de Garcia e Chien (1991) e Waterman (1986), ou seja, são sistemas baseados em conhecimento que aplicam conhecimento de especialistas para resolver problemas, efetiva e eficientemente, numa área limitada do mundo real. Esses sistemas, como os especialistas, usam heurísticas para achar soluções e cometem erros. Além disso, seres humanos necessitam de tempo para amadurecer o conhecimento, o mesmo acontecendo com os sistemas especialistas que passam por estágios, dependendo do grau de especialização: assistente, crítico, segunda opinião, consultor especializado, tutor e automatizador (Hu, 1987), (Bader et alii, 1988).

Atualmente, os sistemas baseados em conhecimento não se restringem a sistemas especialistas, e podem ser encontrados como parte de sistemas convencionais. Entretanto, em Bader (1988) é ressaltado que os sistemas baseados em conhecimento, em geral, não atuam de forma abrangente como a inteligência humana, por não possuirem os três atributos chaves da inteligência: abstração, dedução e aprendizado.

Os sistemas baseados em conhecimento por sua característica ampla podem ser utilizados em diversos domínios de problemas e, normalmente, são agrupados nas seguintes tarefas: interpretação, predição, diagnóstico, projeto, planejamento, monitoramento, prescrição, conserto, instrução e controle (Waterman, 1986), (Rolston, 1988), (Assis, 1992), (Zualkerman, 1986), (Luger e Stubblefield, 1989), (Hu, 1987) (Jain e Chaturvedi, 1989) (Aguiar, 1992). Muitos SBC desenvolvidos, entretanto, não possuem uma única tarefa, conforme exemplos que podem ser encontrados em (Werneck, 1993), (Waterman, 1988), (Rolston, 1988), (Assis, 1992), (Zualkerman, 1986), (Luger e Stubblefield, 1989), (Hu, 1987), (Jain e Chaturvedi, 1989), (Aguiar, 1992).

Outra forma de classificar e identificar os sistemas baseados em conhecimento é através da estrutura de trabalho utilizada para resolver os problemas do domínio da aplicação, podendo ser divididos em *análise* e *síntese* (Jain e Chaturvedi, 1989). A estrutura de análise começa com um problema complexo que é sucessivamente dividido em sub-problemas até o ponto em que esses sub-problemas podem ser resolvidos. Essa forma tem sido empregada com sucesso em problemas de interpretação, predição, diagnóstico, planejamento e classificação. A síntese tem o seu processo iniciado com a solução de sub-problemas que combinados geram uma boa ou, de preferência, uma ótima solução para um problema complexo. Como resultado, a síntese necessita pesquisar uma grande variedade de caminhos possíveis até achar uma boa solução e muitas vezes essas possibilidades acarretam uma explosão combinatória de caminhos, sendo fundamental a utilização de heurísticas para guiar essa busca. Os sistemas relacionados com projeto, conserto, instrução e controle são exemplos de problemas que usam essa estrutura de trabalho.

Myer e Curley (1991) definiram alguns tipos de SBC, considerando o grau de complexidade do conhecimento e da tecnologia. A complexidade do conhecimento é determinada, considerando a complexidade do domínio do problema, sua abrangência que pode ser um único campo de conhecimento ou vários (domínio em largura), o grau de profundidade (domínio em profundidade) e completude do conhecimento. A complexidade da tecnologia considera as diversidades de plataformas de uso do sistema, a diversidade de tecnologias não inferenciais (interfaces gráficas, sistemas gerenciadores de banco de dados, tratamento de imagens, redes, ...), o grau de integração com outros sistemas e o tamanho do esforço de programação lógica.

Na literatura podem ser encontradas comparações entre programas convencionais e sistemas baseados em conhecimento (Lucena, 1987), (Waterman, 1986), (Hu, 1987), (Haynes-Roth, 1984). Uma diferença importante é que nos sistemas convencionais temos dados que são representados e usados com processamento algorítmico, enquanto que os sistemas baseados em conhecimento representam e manipulam conhecimento através do processamento inferencial com aplicação de heurísticas e estratégias. Essas comparações fornecem uma base para a caracterização dos sistemas baseados em conhecimento, que podem ser identificados pelos seguintes aspectos:

- conhecimento específico e intenso do domínio do problema;
- processamento simbólico;
- suporte para análises heurísticas;

- capacidade de reformular o conhecimento, inferindo sobre o conhecimento já existente;
- capacidade de lidar com problemas difíceis;
- habilidade de examinar seu próprio raciocínio e explicá-lo;
- separação entre controle e dados;
- os passos da solução do problema não são explícitos;
- cometem erros, isto é, fornecem respostas aceitáveis e respostas imprecisas.

II.3. Histórico

Os sistemas especialistas surgiram no mercado a nível comercial em meados de 1980, embora esses sistemas já existissem no ambiente de pesquisa desde meados dos anos 60.

Na década de 60, os pesquisadores de Inteligência Artificial, tentaram simular o processo de pensar através dos métodos gerais de solução de problemas, GPS, que foram usados em programas específicos. Os sistemas de produção (Post, 1943) são baseados no GPS, sendo introduzidos na Inteligência Artificial por Newel e Simon por volta de 1957, que perceberam que esses sistemas poderiam modelar alguns aspectos do comportamento inteligente. (Buchanan e Shortliffe, 1985), (Hu, 1987), (Waterman, 1986), (Davis e Lenat, 1982)

O desenvolvimento dos primeiros programas ficou concentrado, basicamente, em três universidades: Stanford, MIT e Carnegie Mellon. A primeira geração (1965-1970) de sistemas especialistas era baseada no desempenho da solução de problemas específicos, usando o conhecimento declarativo e inclui os sistemas DENDRAL, que deduz a estrutura molecular de componentes desconhecidos da massa espectral e MACSYMA, que manipula expressões algébricas e resolve problemas de matemática complexa. A segunda geração (1970-1975) explorou a característica de explicar o raciocínio, um início de aprendizado com a experiência, o uso de conhecimento declarativo de controle e o tratamento de incerteza. Como exemplo temos o MYCIN, que ajuda a diagnosticar e tratar infecções bacterianas; Hearsay, que faz reconhecimento de voz e Prospector, que auxilia geólogos a avaliar o potencial mineral da região. Estas duas primeiras gerações utilizaram linguagens de programação na implementação dos sistemas. A terceira geração (1975 até hoje) se caracteriza pelo uso de ferramentas e

estruturas de controle sofisticadas. Como exemplo de sistemas desta geração, tem-se CASNET, que diagnostica glaucoma e prescreve planos de tratamento; PUFF, que diagnostica doenças pulmonares e R1, que configura sistemas de computadores VAX-11/780.

Não há duvidas que o uso de sistemas baseados em conhecimento vem sendo incorporado ao mercado. Entretanto, a maioria dos sistemas atuais não possuem todas as características mencionadas anteriormente (Munakata, 1993), (Munakata, 1994a).

II.4. Arquitetura dos Sistemas Baseados em Conhecimento

O conhecimento reúne normalmente fatos aplicados a uma área em particular e o conhecimento de como e quando esses fatos devem ser usados para solucionar um problema neste domínio (Falcão, 1992).

Sistemas baseados em conhecimento têm como ponto principal da sua arquitetura a separação entre o conhecimento do domínio do problema e o conhecimento geral da solução do problema. Assim sendo, tem como componentes básicos de sua arquitetura (Figura II.1) a *base de conhecimento*, dependente do domínio e a *máquina de inferência*, independente do domínio, contendo o raciocínio e as estratégias de inferência à base de conhecimento. A base de conhecimento contém conceitos básicos do domínio, relações, fatos, regras ou outras representações do conhecimento que usam esses fatos como base no processo decisório. A máquina de inferência trabalha com as informações da base de conhecimento escolhendo a melhor seqüência e, dependendo dos dados, pode utilizar diferentes paradigmas de controle.

Outros componentes desejáveis nessa arquitetura são: *mecanismos de explicação* para elucidar a linha de raciocínio, *métodos para adquirir novos conhecimentos* e codificá-los na base de conhecimento e *interface de entrada e saída* que pode variar de simples diálogos de telas até um completo processador de linguagem natural e/ou processador de imagens (Walker et alii, 1987), (Gottgtroy, 1990), (Waterman, 1986), (Luger e Stubblefield, 1989).

Capturar o conhecimento, representá-lo e utilizá-lo através do raciocínio são questões chaves na construção desses sistemas (Davis e Lenat, 1982), (Falcão, 1992), (Tanik e Yeh, 1988). A *aquisição do conhecimento* consiste no processo de transformação do conhecimento em uma representação de maneira que possa ser usado

pelo computador. Essa é uma área crucial do paradigma baseado em conhecimento, pois o poder de um programa inteligente está intimamente relacionado com a qualidade e completude de sua base de conhecimento. A *representação do conhecimento* é um conjunto de regras sintáticas e semânticas que permitem descrever o conhecimento em termos de estruturas simbólicas de dados usadas pelo computador e que devem ter flexibilidade para que o conhecimento possa ser adicionado e modificado ao longo do desenvolvimento da base de conhecimento. A *utilização do conhecimento* requer um conjunto combinado de métodos de raciocínio e mecanismos de inferência de forma a conquistar, com eficiência, bem como com acurácia e desempenho o espaço de solução do problema.

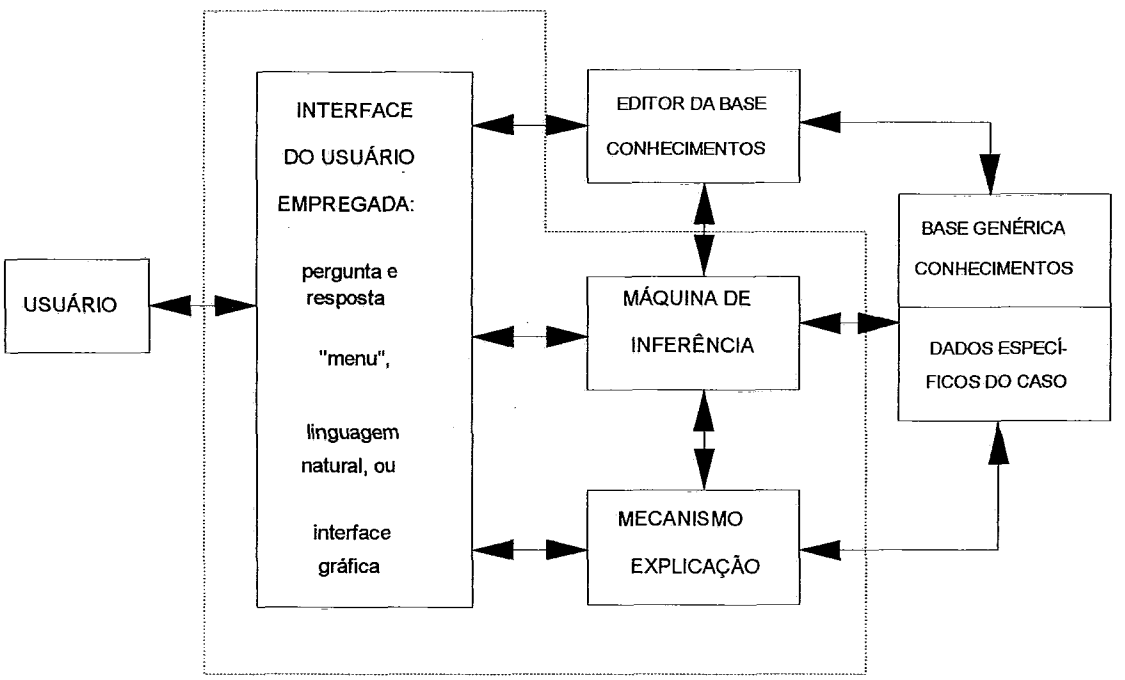


Figura II.1 - Arquitetura Geral de um Sistema Baseado em Conhecimento
(Luger e Stubblefield 1989)

II.5. O Conhecimento e sua Aquisição

Nesta seção discutimos aspectos relacionados ao conhecimento, sua origem e aquisição, participantes nessa aquisição e algumas técnicas de elicitação.

O conhecimento é o ponto chave dos sistemas baseados em conhecimento. Sendo uma área de pesquisa bastante complexa e polêmica, envolve diferentes áreas, como filosofia, psicologia, pedagogia e ciências exatas. Entretanto, o estudo do conhecimento

humano não está no contexto deste trabalho, o que não impede que seja possível se desenvolver sistemas baseados em conhecimento. Esses sistemas são geralmente aplicados para automatizar alguns aspectos do comportamento humano, na solução de determinados problemas ou na realização de tarefas específicas (diagnóstico, planejamento, predição, controle, prescrição, projeto, etc...).

O conhecimento tratado em sistemas baseados em conhecimento é, normalmente, composto de vários elementos que permitem a realização de tarefas com habilidades específicas e podem ser vistos de uma forma geral como: *fatos* (declarações sobre o domínio), *regras procedurais* (regras precisas que descrevem as seqüências dos eventos e as relações relativas ao domínio) e *regras heurísticas* (regras gerais intuitivas ou regras simples e práticas que sugerem procedimentos a serem seguidos quando regras procedurais invariantes não são eficientes. São regras aproximadas e a qualidade desse conhecimento determina o grau de especialização). Um modelo conceitual geral do domínio e um esquema global para achar a solução complementam esses tipos de conhecimento e representam o conhecimento de um especialista (Assis, 1992), (Rolston, 1988).

Os SBC se justificam, principalmente, por serem uma forma de registrar, transferir e multiplicar o conhecimento especializado de um domínio. É fundamental para o sucesso da utilização da tecnologia baseada no conhecimento a escolha de domínios apropriados e aplicação em situações específicas. Waterman (1986) e Laufmann et alii (1990) discutem aspectos sobre a aplicação desta tecnologia.

A obtenção do conhecimento de um SBC pode ser feita de diferentes maneiras através da aquisição que transforma o conhecimento coletado em uma representação que o computador entenda, isto é, que seja possível de ser implementada. Esta atividade pode ser realizada diretamente pelo especialista no domínio do problema, pelo engenheiro do conhecimento ou, em conjunto, pelo especialista e pelo engenheiro de conhecimento.

A elicitação do conhecimento num SBC é a etapa da aquisição do conhecimento que trabalha diretamente com o conhecimento, permitindo elaborar um modelo do mesmo, e cujo foco é a coleta dos dados. A obtenção do conhecimento pode ser efetuada através de (Assis, 1992), (Gottgtroy, 1990), (Patterson, 1990), (Walker et alii, 1987), (Forsyth, 1987):

- **Meio Bibliográfico**
  O conhecimento é extraído de material bibliográfico sobre o domínio pelo engenheiro de conhecimento.

- **Especialista**

  O conhecimento é obtido de um ou mais especialistas no domínio pelo engenheiro de conhecimento. Nesse caso é necessário utilizar técnicas para elicitação do conhecimento, para garantir a sua efetiva aquisição. Algumas técnicas de elicitação do conhecimento que efetivam a sua aquisição podem ser encontradas em (Werneck, 1993), (Assis, 1992), (Gottgtroy, 1990), (Scott et alii, 1991), tais como: entrevistas, análise de protocolo, cenários, questionário, fatores críticos de sucesso, caso típico, ordenação conceitual, protocolo por telefone, diagrama e hipóteses terminais.

- **Protótipo**

  É construído um protótipo do sistema a partir da leitura de documentos relevantes ao domínio. O especialista avalia o sistema apontando problemas e estratégias alternativas de solução.

- **Ferramentas de Edição do Conhecimento**

  Os especialistas do domínio, com ajuda do engenheiro de conhecimento, interagem com um programa de edição do conhecimento que incorpora conhecimento do domínio de várias formas.

- **Ferramentas de Aquisição do Conhecimento Automatizadas e Semi-Automatizadas**

  O especialista do domínio para incorporar o conhecimento no computador interage com o sistema de aquisição do conhecimento.

- **Programas de Entendimento de Texto**

  Um sistema desse tipo lê e entende documentos representando o conhecimento através de estruturas apropriadas.

- **Mista**

  Neste caso, os programas de entendimento de texto são seguidos pela apreciação de especialistas ou pela utilização de ferramentas de edição do conhecimento ou de aquisição do conhecimento.

- **Aprendizado Automático**

  Aprendizado no contexto de sistemas baseados em conhecimento é a

habilidade que o sistema apresenta de se adaptar a mudanças no ambiente, realizando as tarefas de forma mais efetiva e eficiente numa próxima execução. Os programas de aprendizado automático automatizam o processo de aquisição do conhecimento permitindo que os sistemas baseados em conhecimento tenham capacidade de aprendizado. Esses programas podem ser classificados considerando sua capacidade de inferência em: *aprendizado por instrução* (o usuário ou ambiente fornece conhecimento muito abstrato ou geral e desse conhecimento é inferido o conhecimento específico), *aprendizado por analogia*, (o sistema capaz de gerar novos conceitos do domínio, partindo de episódios de resoluções de problemas passados), *aprendizado por descoberta* (o sistema descobre novos fatos e teorias a partir de informações já existentes e sem a ajuda do usuário no fornecimento de exemplos) e *aprendizado por exemplos* (o programa de aprendizado, partindo de exemplos específicos, é capaz de criar suposições de estruturas gerais do conhecimento para serem aplicadas em outras situações, que pode ser de um dos seguintes tipos: *aprendizado indutivo*, onde o programa gera, através de um processo indutivo e mediante um conjunto de exemplos, uma descrição do conceito que explica todos os exemplos positivos e nenhum negativo; *aprendizado dedutivo ou analítico* onde o programa de aprendizado é apresentado através de um exemplo de um conceito do domínio e a descrição funcional desse exemplo e os conceitos gerais ou específicos do domínio são gerados a partir desse exemplo juntamente com a teoria existente, *aprendizado pelo caminho da solução* onde o programa faz uma análise do caminho da solução e gera heurísticas a partir dessa análise).

II.6. Representação do Conhecimento

No desenvolvimento dos sistemas baseados em conhecimento a construção da base de conhecimento é um elemento chave para a transferência do conhecimento. Na representação do conhecimento temos, basicamente, dois tipos de entidades envolvidas: *fatos* e *representação dos fatos* (Rich, 1985), (Duarte, 1991). Fatos são verdades relevantes a algum domínio do problema e são o que se quer representar, enquanto a representação dos fatos é um formalismo expresso em estruturas de conhecimento, sendo efetivamente o que seremos capazes de manipular.

Algumas características do problema devem ser consideradas para a escolha dos esquemas de representação. Assim sendo, deve-se considerar os seguintes fatores (Woods, 1986):

- *adequação*, que é a capacidade de representação do método ser a mais natural possível, sempre que for necessário representar e inferir um determinado conhecimento;
- *eficiência*, que é a facilidade de implementação da representação e a velocidade e simplicidade dos recursos computacionais necessários para obter resultados ou processar inferências;
- *granularidade*, que diz respeito ao nível de detalhe do conhecimento numa representação e o nível de abstração da estrutura de dados, eliminando detalhes;
- *extensão* ou tamanho do problema que diz respeito ao fato da representação do espaço de solução ser grande ou pequeno e da representação e inferência do sistema poderem trabalhar com os recursos computacionais disponíveis;
- *inferência associada*, dado que qualquer mecanismo de representação deve estar associado a um ou mais mecanismos de inferência, a operação desse mecanismo deve ser facilitada pela representação do conhecimento utilizada;
- *formal ou informal*, onde "formal" significa seguir as idéias de consistência e completude da lógica formal e "informal" os outros métodos de representação;
- *Determinístico ou não*, considerando-se que o método de representação é determinístico se permite um único caminho (quando este existe) e não determinístico quando atinge o objetivo através de vários caminhos.

Mylopoulos e Levesque classificaram as estruturas de representações em quatro categorias (Luger e Stubblefield, 1989):

- *lógica*, que usa expressões da lógica formal para representar o conhecimento, aplicando regras de inferência e procedimentos de prova formal a instâncias do problema. A lógica de cálculo de predicados de primeira ordem é a estrutura lógica mais usada;
- *procedural*, que representa o conhecimento como um conjunto de instruções para solução do problema. As regras de produção são um exemplo dessa classe;

- *redes*, que capturam o conhecimento como um grafo onde os nós representam objetos ou conceitos do domínio do problema e os arcos representam as relações e associações entre os nós. Exemplos desse esquema são redes semânticas, dependência conceitual e grafos conceituais;
- *estruturada*, que estende o esquema de redes permitindo que cada nó seja uma estrutura de dados complexa chamada *slots*, associada com seus valores que podem ser dados numéricos ou simbólicos, ponteiros para outros esquemas ou procedimentos de uma tarefa específica. Exemplos: *frames*, *scripts* e objetos.

Os sistemas lógicos têm uma série de vantagens, tais como, expressão natural de certas idéias, força e expressividade da lógica e modularidade, pois permite que outras declarações sejam colocadas na base de conhecimento independente das existentes. Outra vantagem é a precisão no sentido de que possui regras de inferência com vínculo semântico, base de conhecimento logicamente consistente (na teoria) e todas as conclusões, garantidamente, corretas (Lucena, 1987), (Luger e Stubblefield, 1989), (Rich, 1985), (Walker et alii, 1987), (Gottgtroy, 1990). A representação lógica é, também, flexível pois os fatos são representados de uma forma que permite uma interpretação global e sua utilização para diferentes objetivos. Entretanto, existem alguns problemas resultantes dessa forma de representação, como a excessiva generalidade e a ausência de facilidades para definição de sintaxes mais complexas. Outra dificuldade é o tempo excessivo que pode levar a construção da prova para algum objetivo e também a necessidade de se ter um grande conhecimento de lógica. PROLOG é uma linguagem que implementa esquemas de representação lógica e muitos refinamentos foram feitos ao princípio de resolução, buscando otimizar o tempo de busca a informações importantes.

As regras de produção são a representação mais popular e efetiva para descrições declarativas do conhecimento em sistemas baseados em conhecimento, tendo sido desenvolvidas por Newell e Simon em 1972, como uma arquitetura cognitiva humana (Waterman, 1986), (Patterson, 1990), (Duarte, 1991), (Lucena, 1987), (Gottgtroy, 1990), (Luger e Stubblefield, 1989), (Rich, 1988), (Walker et alii, 1987). As regras podem ser representadas no cálculo de predicados com um elemento prescritivo que indica como a informação contida nas regras deve ser usada durante o raciocínio. Regras de produção são representações flexíveis do conhecimento, fáceis de serem criadas e entendidas pelas pessoas. Outra vantagem é terem poder de expressão suficiente para representar as regras de inferência do domínio e especificações de conduta gerais. A representação em regras é mais adequada a problemas com assertivas ou atividades

independentes, tais como diagnóstico, sendo inadequada à definição de termos, à descrição de objetos do domínio e ao estabelecimento de relações estáticas entre os objetos.

Entretanto, apesar destas vantagens, o emprego de regras de produção apresenta uma série de limitações, como busca seqüencial repetida na base de conhecimento o que compromete a eficiência do sistema. Outra dificuldade está relacionada à ordenação das regras que impõe uma diminuição de sua independência. Novas regras, também, podem significar problemas pois, muitas vezes, repetem o conhecimento contido na base e essa redundância compromete a eficiência e dificulta a manutenção. O conhecimento procedural representado por regras de produção é, ainda, difícil de manipular, pois fica misturado na base de conhecimento, não havendo a separação explícita do conhecimento específico do domínio e do controle. O conhecimento impreciso e os conceitos quantificáveis são, também, difíceis de serem manipulados nessa estrutura.

Redes semânticas são um esquema de representação definido por Quilan em 1968 e que consistem numa estrutura simples e fácil de ser entendida. Sua notação representa o conhecimento através de um grafo com nós, representando os fatos ou estados de problemas, e arcos que representam as inferências. As conclusões são obtidas através da busca dos passos de inferência entre os nós de início e os de objetivo (Waterman, 1986), (Patterson, 1990), (Ribeiro, 1989), (Duarte, 1991), (Lucena, 1987), (Gottgtroy, 1990), (Luger e Stubblefield, 1989), (Rich, 1985). É um método bastante flexível com poucas restrições. Permite a definição de uma grande variedade de objetos, atributos, conceitos e relacionamentos.

A implementação de redes semânticas pode ser incorporada facilmente, seja numa linguagem procedural ou lógica, pois esta representação tem a capacidade de representar relações da lógica de predicados. Os tipos de inferência mais usados na representação de redes semânticas são propagação, busca e herança de propriedades. A herança acarreta que esta seja uma representação natural para domínios onde existe uma taxonomia complicada a representar. Entretanto, embora esta característica seja admitida como importante para implementação, ela não expressa uma superioridade comunicativa ou expressiva das redes semânticas

Frames foram propostas em 1975 por Minsky, como sendo uma estrutura para a representação do conhecimento existente de objetos, eventos e situações estereotipadas em geral. Permite um nível grande de detalhes do domínio (Haynes, 1980), (Lucena, 1987), (Luger e Stubblefield, 1989), (Rich, 1985), (Buchanam e Shortliffe, 1985),

(Waterman, 1986), (Patterson, 1990), (Walker et alii, 1987). Uma *frame* é composta por *slots*, espaços a serem preenchidos por um determinado objeto e relações que são inerentes a uma certa situação, sendo que associadas a cada frame encontram-se as seguintes informações: como usar a *frame*, o que fazer se ocorrer algo inesperado, valores padrão para os *slots* e um conjunto de condições que devem ser satisfeitas pelo objeto que vai preencher os valores dos *slots*. A representação por esse método tem sido útil, principalmente, para domínios do problema em que o conhecimento é hierárquico ou contextual ou para conhecimento conflitante ou incompleto. São freqüentemente utilizadas em conjunto com outros formalismos, como por exemplo uso de *frames* ligadas a uma base de regras, através da ligação de regras a *slots*. A orientação a objetos pode ser uma forma de implementação de frames, pois sua representação tem conceitos similares (Patterson, 1990).

A elaboração e uso de *frames*, entretanto, também, tem algumas complicações pois sua estrutura pode ser interpretada de diferentes maneiras podendo gerar ambiguidade principalmente de interpretações factuais e de definição.

A orientação a objetos é outra representação que tem sido utilizada em vários sistemas baseados em conhecimento (Duff e Carlson, 1991), (Atabakhsh e Chan, 1994), (Dai e Hughes, 1993), (Saake et alii, 1993), principalmente combinada com regras.

As metas-regras são representações procedurais que estruturam o conhecimento em diversos níveis para expressar propriedades de outras regras e também para usar essas propriedades à invocação de uma regra. As meta-regras estão fundamentadas no conceito de meta-conhecimento (Davis e Lenat, 1982), (Buchanan e Shortliffe, 1985), (Davis, 1980), (Patterson, 1990), que é o conhecimento sobre o conhecimento.

As vantagens encontradas na representação das meta-regras são permitir uma representação independente, separando o conhecimento dependente do domínio da aplicação, do dependente das estratégias de solução do problema e representar explícitamente as estratégias de controle. Suas desvantagens estão relacionadas na dificuldade de representar o conhecimento a nível meta e objeto e no tratamento de conflitos no nível objeto.

A aplicabilidade da representação de meta-regras é admitida, principalmente, para problemas com baixo grau de estruturação, ou domínios centrados em tarefas. Em grandes sistemas onde a construção é feita, incrementalmente, essa representação, por sua característica de transparência, torna explícita as informações de controle e permite uma facilidade de manutenção e expansão da base de conhecimento.

Com relação às estruturas de representação, a experiência tem mostrado que o uso combinado de diferentes técnicas tem obtido sucesso pois aproveita as vantagens de cada um dos componentes da representação, embora permaneça o problema da escolha do formalismo adequado para um determinado tipo de conhecimento e inferência.

II.7. Mecanismos de Utilização do Conhecimento (Scott et alii, 1991), (Hu, 1987), (Patterson, 1990), (Forsyth, 1987), (Harmon e King, 1988), (Buchanan e Shortliffe, 1985), (Gottgtroy, 1990), (Waterman, 1986), (Lucena, 1987)

A utilização da informação dinâmica, com o conhecimento estático da base de conhecimento pela máquina de inferência, deriva conclusões sobre o caso corrente fornecido pelo usuário ao ambiente, através da interface de entrada e saída.

A máquina de inferência é o componente do sistema baseado em conhecimento que provê mecanismos de inferência e de controle para a aplicação do conhecimento. Ao aplicar o conhecimento associado à base de conhecimento o sistema estará executando uma tarefa. O mecanismo de inferência possibilita ao sistema raciocinar a partir dos dados de entrada para obter os resultados de saída, enquanto o mecanismo de controle governa a ordem em que o sistema deve desempenhar os passos do raciocínio, aceitar as respostas e produzir as saídas.

O comportamento inteligente pode ser reconhecido através da identificação da solução do problema, do processo de raciocínio e do controle, pois essa solução pode ser vista com freqüência como uma busca entre as alternativas existentes.

A separação do conhecimento dos programas que interpretam e aplicam o conhecimento é importante, pois facilita o desenvolvimento, a extensão e a manutenção desses programas que contém lógicas complexas, repercutindo também na explicação fornecida pelo sistema sobre o seu comportamento.

A representação do conhecimento influencia o paradigma de controle da máquina de inferência. Por isso esses componentes não são totalmente independentes. Assim sendo, os sistemas baseados em conhecimento podem ser caracterizados pelas diferentes técnicas utilizadas na representação do conhecimento. Sistemas baseados em regras, sistemas baseados em *frames*, sistemas de árvore de decisão, são alguns sistemas identificados por sua representação do conhecimento.

Outra forma de caracterizar os sistemas baseados em conhecimento tem relação com estrutura de controle: direção da busca, técnicas de controle empregada e transformações aplicadas no espaço de busca.

II.7.1. Mecanismos de Raciocínio (Hu, 1987), (Patterson, 1990), (Forsyth, 1984), (Harmon e King, 1988), (Lugger e Stubblefield, 1989), (Buchanan e Shortliffe, 1985), (Felix, 1991), (Ribeiro, 1989), (Haynes, 1980), (Waterman, 1986)

Os mecanismos de raciocínio mais utilizados em sistemas baseados em conhecimento são a aplicação de uma regra lógica (*modus ponens*), resolução e raciocínio em condições de incerteza.

A resolução é utilizada principalmente em sistemas lógicos, representados pelo cálculo de predicados de primeira ordem. O processo consiste em negar a "fórmula bem definida" que representa o objetivo, incluir a negação desta à base, converter toda a base em cláusulas e através da resolução tentar chegar à cláusula vazia. Com a chegada à cláusula vazia se prova que o objetivo é conseqüência lógica da base de conhecimento, confirmando a veracidade deste.

Os sistemas baseados em regras, também conhecidos como sistemas de produção, têm três partes distintas: a base de conhecimento, a base de dados ou fatos ou memória de trabalho e o interpretador ou máquina de inferência. A base de conhecimento utiliza a representação de regras de produção. Uma regra é aplicada sobre a base de dados sempre que o antecedente (situação) desta combinar ou seja, for verdadeira. A máquina de inferência é responsável pela escolha da regra a ser aplicada e a execução desta. A base de dados representa o estado atual do processo de solução do problema, sendo esta modificada pelo conseqüente da regra (ação) decorrente da ativação de uma determinada regra.

Os sistemas de produção operam em ciclos. A regra ou regras que combinam com o estado atual da base de dados são selecionadas e aplicadas. Ao se aplicar uma produção, suas ações modificam a base de dados fazendo que novas produções combinem. Este ciclo continua até não existir mais nenhuma regra que combine ou que se encontre um comando de parada.

O processo de inferência dos sistemas de produção é executado recursivamente em três estágios: comparação (*match*), seleção e execução. O estágio de comparação é um

processo que compara duas ou mais estruturas de conhecimento para descobrir suas semelhanças ou diferenças. É uma operação essencial em programas de reconhecimento, compreensão de linguagens naturais, planejamento, programação automática e sistemas especialistas.

A comparação pode ser exata utilizando ou não variáveis padrão ou parcial, dado que muitas vezes a comportamento exato é inapropriado. Nesse caso procura-se obter o melhor comportamento entre pares de estruturas, ou comparam-se elementos chaves do conhecimento ignorando outros menos relevantes. Alguns problemas necessitam uma comparação nebulosa, onde os conceitos não são delimitados claramente.

As métricas para comparação podem ser diferentes tipos de medidas usadas para determinar a similaridade ou proximidade de dois objetos, por exemplo métricas de distância como a distância Euclidiana, medidas probabilísticas, medidas qualitativas, diferentes medidas de similaridade e medidas nebulosas.

Outras estruturas de conhecimento utilizam também o processo de raciocínio por comparação. Em *frames* é possível comparar uma instância de um conceito com uma instância de outro, visualizando assim novas relações através de abstrações feitas nesse processo.

No caso de *frames* é possível ter outros mecanismos de raciocínio como a já mencionada lógica, incerteza, raciocínio por padrão, por suposições ou por probabilidade, que serão descritos a seguir.

II.7.2. Raciocínio por Incerteza (Hu, 1987), (Patterson, 1990), (Forsyth, 1987), (Harmon e King, 1988), (Lugger e Stubblefield, 1989), (Buchanan e Shortliffe, 1985), (Monat, 1989), (Felix, 1991), (Avanzato, 1991), (Rich, 1985)

Aplicações orientadas para o processamento do conhecimento requerem a habilidade de lidar adequadamente com informações incertas ou com problemas complexos onde se necessita de representação por abstrações. Incertezas podem surgir por diferentes razões, tais como, informação disponível incompleta ou altamente volátil, muitos fatos imprecisos, vagos ou nebulosos ou ainda informações disponíveis contraditórias ou inacreditáveis. Seres humanos, milagrosamente, lidam diariamente com incertezas. Usualmente chegam a soluções racionais mudando as situações de seu mundo.

Alguns métodos foram propostos com o objetivo de tratar inconsistências, incertezas, possibilidades e verdades, isto é, de raciocinar com incertezas. Nos processos de raciocínio por *modus ponens* e resolução, a lógica utilizada é conhecida como lógica monotônica. Os valores deduzidos permanecem verdadeiros e novos conhecimentos são adicionados à base de conhecimento aumentando-a monotonicamente. Esta só pode crescer, nunca diminuir. O conhecimento introduzido à base não contraria o conteúdo desta.

Na lógica não monotônica novos fatos ficam sendo conhecidos, contradizem e invalidam antigos conhecimentos. Estes podem ainda invalidar outros conhecimentos dependentes, causando uma retração da base de conhecimento com o crescimento contraído ou não monotônico.

Vários esquemas têm sido experimentados para permitir o uso de informações fragmentadas e incertas na estimação da verdade. Como sistemas de manutenção da verdade, raciocínio por padrão, suposição de um mundo fechado (CWA, *closed world assumption*), lógica modal e temporal, lógica nebulosa, fatores de certeza, teoria Bayesiana, teoria da evidência, métodos de raciocínio heurístico .

Os *sistemas de manutenção da verdade* permitem uma forma de raciocínio não-monotônico através da manutenção de um conjunto verdade mesmo com a adição de mudanças na base de conhecimento. O objetivo principal desses sistemas é manter a consistência do conhecimento usado pelo solucionador do problema. A máquina de inferência soluciona os problemas do domínio baseada no conjunto atual de verdade enquanto o sistema de manutenção da verdade mantém esse conjunto corrente verdadeiro. Para cada inferência, esses dois componentes trocam informações. A máquina de inferência informa que deduções foram feitas e o sistema de manutenção da verdade pergunta ao mecanismo de inferência sobre as verdades correntes e raciocina, identificando falhas e mantendo consistente a base de conhecimento.

Raciocínio por padrão é outro exemplo de raciocínio não-monotônico. É utilizado quando se tem um conhecimento incompleto. Suposições padronizadas de conhecimentos ou verdades que ainda não foram diretamente provadas são feitas e consideradas válidas, até que uma nova informação que contrarie essas hipóteses seja obtida. O raciocínio por padrão foi desenvolvido por Reiter em 1980, no contexto da lógica tradicional. Consiste de um conjunto de axiomas e um conjunto de regras padrão, sendo que essas são particularmente úteis em bases de conhecimento hieráquicas. Seu objetivo é suprir os vazios do conhecimento de um determinado domínio do problema,

permitindo que se realizem inferências e assim fornecer uma resposta ou executar uma ação.

A *lógica modal* foi definida para estender o poder das lógicas tradicionais, acrescentando a habilidade de expressar a necessidade, possibilidade, obrigatoriedade, credibilidade, verdades conhecidas e situações temporais. São encontrados vários tipos de lógicas modal, que podem ser classificados pela modalidade que expressam: lógica alética (*alethic*) (necessidade e possibilidade), lógica *deontic* (obrigatoriedade e permissibilidade), lógica epstêmica (verdade e conhecido) e lógica temporal (modificadores do tempo, tais como, sempre, muitas vezes, o que aconteceu, o que acontecerá ou o que é).

A *lógica nebulosa* também foi introduzida para generalizar e estender a expressividade das lógicas tradicionais. Está baseada na teoria de conjunto nebuloso que permite capturar significados de conceitos lingüísticos inerentemente vagos. O propósito da lógica nebulosa é lidar com conceitos que geram conjuntos de contorno não claramente distinguíveis, sendo que nestes um elemento pertencerá ou não a um conjunto, dentro de uma escala contínua de pertinência.

No tratamento de incertezas, muitos sistemas utilizam *métodos probabilísticos* com níveis de confiança ou computação de probabilidade de inferência. Cada opinião ou crença é admitida com um grau de verdade ou plausibilidade e a evidência é associada a funções combinadas onde é calculada a probabilidade de uma opinião ser verdadeira. O método de probabilidade bayesiano foi utilizado por vários sistemas baseados em conhecimento, tais como o PROSPECTOR e está fundamentado no teorema de Bayes proposto no século dezoito. Esta forma de raciocínio depende do uso de probabilidade em caráter condicional de eventos específicos, a partir do conhecimento da ocorrência de outros eventos.

A *teoria de evidência,* conhecida como teoria de Dempster-Shafer, foi proposta em 1968 por Dempster e estendida em 1986 por Shafer. Esse método utiliza o acúmulo de evidência para direcionar o processo de escolha das hipóteses mais prováveis. Surgiu como uma alternativa para casos onde não era possível utilizar o método de Bayes pela falta de dados estatísticos. Pode ser considerada como uma generalização da teoria geral da probabilidade que permite atribuir um grande número de probabilidades a todos os sub-conjuntos do universo e não somente aos elementos básicos. A medida de certeza da verdade para uma assertiva é então computada como sub-intervalos de 0 a 1, onde o tamanho do intervalo mede a incerteza.

Os *métodos de raciocínio heurístico* têm por base o uso de procedimentos, regras e outras formas de codificação do conhecimento para atingir objetivos específicos em situação de incerteza. O uso combinado de heurísticas gerais e específicas do domínio pode escolher uma das várias alternativas de solução, através do poder das evidências negativas e positivas apresentadas em formas de justificativas ou endossamentos (sem responsabilidade ou sem vínculo). Os pesos dos endossamentos empregados nos sistemas não necessitam ser numéricos e alguma forma de seleção de preferência ou ordenação deve ser empregada .

Os endossamentos são usados para controlar o processo de raciocínio de duas formas diferentes. Na primeira a preferência é dada às regras que são mais fortemente suportadas, enquanto a outra permite que uma condição ou premissa de uma regra seja satisfeita ou rejeitada sem *match* exato.

Em Avanzato (1991) é feita uma classificação das técnicas de tratamento de incerteza apresentadas acima. A escolha do gerenciamento de incertezas é analisado através de uma comparação desses métodos considerando as seguintes propriedades: clareza (proposições bem definidas), continuidade escalar, completude, dependência contextual, condicionamento hipotético, complementariedade e consistência. O autor conclui afirmando a importância da escolha do mecanismo de incerteza no início do desenvolvimento do sistema, embora não exista uma maneira simples para guiar essa escolha. O domínio da aplicação, a natureza do conhecimento e o requisito de desempenho são fatores, que deverão ser analisados em cada um dos métodos existentes, para se definir o mecanismo a ser empregado.

II.7.3. Estratégias de Controle (Patterson, 1990), (Luger e Stubblefield, 1989), (Harmon e King, 1988), (Waterman, 1986), (Lucena, 1987), (Rich, 1985), (Davis, 1980), (Ribeiro, 1989)

O controle dos mecanismos de busca faz-se necessário para minimizar o tempo de raciocínio, pois a base de conhecimento pode ter um conjunto muito grande de estruturas de conhecimento aplicáveis à solução do problema. O sistema deve ter meios de decidir onde começar o processo de raciocínio e escolher como resolver conflitos quando estes surgirem.

Os problemas resolvidos por sistemas baseados em conhecimento podem ser representados por um grafo e, assim, a solução do problema passa a ser uma busca a um

determinado nó do grafo através da descoberta do caminho para chegar à solução nesta representação. As buscas podem ser de vários tipos dependendo da informação usada. A busca em profundidade e em largura são exemplos de busca desinformada. A busca heurística é uma busca informada que expande um nó no grafo de solução, a partir de uma heurística que é uma forma de agir numa situação nova mas semelhante a uma outra.

As heurísticas têm um papel importante nas estratégias de busca pois muitos problemas são de natureza exponencial. Ajudam a reduzir o número de alternativas produzindo uma solução numa quantidade tolerável de tempo. Embora, nesse tipo de busca não se tenha garantia de conseguir a melhor solução, em muitos casos uma solução boa ou aceitável é obtida. As heurísticas são mais eficientes quando se tem continuidade e estabilidade no ambiente e quando esse conhecimento for possível de ser adquirido, formalizado e avaliado.

Nos sistemas baseados em regras, as estratégias que direcionam o processo de busca podem ser dirigidas para o objetivo (*top-down*) ou dirigidas para os dados (*bottom-up*), sendo conhecidas, também, como *encadeamento regressivo* e *encadeamento progressivo*. Essas estratégias são maneiras de escolher uma determinada regra.

É possível a aplicação combinada dessas estratégias de direcionamento de busca das regras. Como benefício obtém-se uma maior otimização da resolução do problema pois dependendo do tipo de busca empregada, existe a possibilidade de faltar alternativas importantes no espaço de soluções.

Outro mecanismo de controle são as *regras de controle*, que explicitam o controle representando-o de forma semelhante às regras dependentes do domínio do problema e utilizando, também, o mesmo mecanismo de inferência. Esse processo de controle através de regras tem, entretanto, algumas desvantagens, como a dificuldade de representação, pois o conhecimento sobre o controle não é necessariamente o mesmo do seu domínio. Outra dificuldade está relacionada à estrutura de trabalho para organização e uso do conhecimento que é potencialmente frágil. Sua eficiência ainda não pode ser comprovada, pois não existe muita experiência na aplicação desse mecanismo. Entretanto, uma grande vantagem dessa representação do controle em regras é tornar explícito esse conhecimento, facilitando o processo de explicação do raciocínio que será descrito a seguir.

II.7.4. Explicação do Raciocínio (Buchanan e Shortliffe, 1985), (Davis e Lenat, 1982), (Wick, 1992), (Basu e Ahad, 1992), (Jao e Hier, 1993), (Kim, 1993)

Sistemas baseados em conhecimento devem ter, como uma de suas principais características, a facilidade de explicar o seu comportamento que se define na capacidade do sistema mostrar de forma compreensível os motivos de suas ações.

Os motivos para que os sistemas baseados em conhecimento expliquem o seu raciocínio são variados. Tornar compreensível o conteúdo da base de conhecimento e sua linha de raciocínio é um dos maiores objetivos da explicação. Pode, também, racionalizar o processo de depuração do sistema, especialmente devido ao fato da base de conhecimento ser desenvolvida incrementalmente. A educação também é uma razão importante pois o aprendizado é um fator que torna interessante o uso do sistema fomentando a exploração e sedimentação do conhecimento. A explicação também ajuda na persuasão e aceitação do sistema, pois mostra que as conclusões são corretas ou razoáveis explicitando o conhecimento e a linha de raciocínio.

O ideal é que seja fácil para o usuário obter respostas compreensíveis e completas para qualquer pergunta sobre o conhecimento e a operação do sistema. Para isso, o módulo de explicação deve ter as seguintes características: *acurácia* (no sentido de prover uma visão precisa do que está acontecendo com o desempenho do sistema), *compreensibilidade* (no sentido de ser facilmente entendido por usuários, o que implica em restrições ao conteúdo e vocabulário enfatizando a brevidade), *engenharia humana* (no sentido de ser fácil de usar, ser poderoso e rápido).

A explicação pode ser *comportamental* ou *estrutural*. A explicação comportamental está relacionada com perguntas do porque uma informação é necessária para resolver um problema ou como uma solução foi obtida ou porque uma solução em particular não foi obtida. A explicação estrutural é caracterizada por possuir mecanismos mais abrangentes do que a explicação comportamental. A explicação estrutural tem como propósito uma maior compreensão da base de conhecimento, sendo que o uso de tutores associados com a tecnologia de hipermídia podem auxiliar numa maior compreensão do conhecimento do sistema.

O desenvolvimento de mecanismos de explicação está relacionado aos problemas de representação do conhecimento e de utilização de diferentes fontes de conhecimento. As pesquisas levam ao desenvolvimento de métodos dinâmicos para determinar a complexidade e importância do conhecimento, descobrir e melhorar as técnicas para uso

do diálogo de contexto para guiar a formação da explicação, uso de métodos lingüísticos ou psicológicos para determinar a razão do usuário ter feito uma pergunta, permitindo uma resposta específica para aquele usuário, e desenvolvimento de técnicas para o gerenciamento de níveis de complexidade e detalhes inerentes ao mecanismo enfatizando os processos psicológicos.

II.8. Tendências Atuais nos Sistemas Baseados em Conhecimento

Sistemas baseados em conhecimento tornaram-se um grande sucesso comercial nos Estados Unidos, Europa e Ásia. Vários sistemas estão sendo utilizados com ganhos econômicos significativos no que se refere ao aumento de produtividade do trabalho profissional, diminuição dos custos internos de operação, retorno do investimento, melhora na qualidade e consistência da tomada de decisão, aquisição do conhecimento organizacional, melhoria na forma de realização do próprio negócio das empresas, gerenciamento de crises e estímulo para inovações (Feigenbaum et alii, 1994).

Como tecnologia, os sistemas baseados em conhecimento passaram da etapa de experimentação e estão na fase de maturação. Por outro lado, os sistemas atuais são menos ambiciosos que os de 10 anos atrás e pode-se afirmar que eles realmente assessoram pessoas, provendo uma especialidade valiosa e economizando bastante (milhões de dólares ao ano por sistemas) (Munakata, 1994a). Haynes-Roth e Jacobstein (1994) forneceram uma visão geral da área de sistemas baseados em conhecimento, concluindo que esses alcançaram um papel permanente e seguro nas indústrias.

O uso de bases de conhecimento é revolucionário, encontrando barreiras culturais que muitas vezes aliam ao alto custo associado aos estágios iniciais de comercialização e do desenvolvimento do mercado. As dificuldades associadas ao mercado e ao desenvolvimento tecnológico criaram uma impressão de que a tecnologia falhou. Entretanto, os atuais resultados contradizem essa crença.

Os SBCs prometem uma radical mudança na distribuição da especialidade e em geral na prosperidade. Entretanto, experiências práticas mostram que os SBCs, raramente, representam mais do que 20% da solução completa de um problema no mundo real. O que acontece nas empresas hoje em dia, é que estas sofrem mais pela falta de integração do que de conhecimento. Assim sendo, para os SBCs se tornarem efetivos, devem ser integrados às operações do dia a dia.

O processo de aquisição do conhecimento continua sendo um desafio. Duas direções estão surgindo como promessas para essa área. A primeira provê uma linguagem específica para o domínio da especialidade e a segunda busca instalar o conhecimento e fornecer assessoramento mais do que emular o comportamento de um especialista. Hoje existem ferramentas que são úteis para a aquisição e instalação do conhecimento, tais como, sistemas de indução de regras a partir de dados e raciocínio baseado em casos.

O *raciocínio baseado em casos* (CBR) é uma estrutura para solução de problemas baseada na consulta e adaptação de casos ou descrições de episódios de problemas e suas soluções associadas. Atualmente existem alguns sistemas desenvolvidos utilizando CBR e algumas ferramentas de programação já dispõem desta estrutura de trabalho.(Allen, 1994)

Em Haynes-Roth e Jacobstein (1994) é identificado que as pesquisas de SBC estão na vanguarda de vários esforços que irão revolucionar a integração, tornando possível agregar aplicações através de componentes que interagem de forma inteligente. Os componentes inteligentes podem interagir cooperativamente, sendo que cada um poderá utilizar o paradigma que for mais conveniente para a solução do problema.

Experiências combinando as tecnologias existentes têm sido utilizadas com sucesso, o que é coerente, pois cada solução é melhor aplicada a um determinado tipo de problema. Os sistemas baseados em conhecimento em particular deverão desenvolver a interface, e o módulo de explicação, possibilitando uma interação em linguagem natural ou ao vivo com o usuário e melhorando o âmbito de abrangência da explicação.

O aprendizado automático deverá ser uma característica cada vez mais presente nos SBCs (Rubin, 1993). Neste sentido os sistemas híbridos podem dirigir o processo de solução, utilizando as vantagens e benefícios das diferentes representações e mecanismos de utilização do conhecimento e dos diferentes paradigmas da inteligência artificial, tais como, a combinação de redes neurais e sistemas baseados em conhecimento (Cho et alii, 1991), (Hall, 1992), (Asakawa e Takagi, 1994), (Mendonça et alii, 1994), (Leão e Reategui, 1993).

Os sistemas *fuzzy* utilizam a teoria de conjuntos nebulosos e a lógica nebulosa e têm sido desenvolvidos comercialmente com sucesso nos últimos anos, sendo também baseados em conhecimento ou especialistas. Esses sistemas são aplicáveis a problemas complexos ou aplicações que envolvem forma descritiva humana ou pensamento intuitivo. Alguns desses sistemas combinam os princípios dos conjuntos nebulosos com outras representações como, por exemplo, regras de produção (Munakata, 1994b).

Hoje as ferramentas existentes permitem o uso de diferentes modelos de representação e de inferência, o que possibilita experimentar diferentes soluções para um determinado problema. Futuramente as aplicações poderão ser criadas utilizando representações híbridas combinando casos, regras e modelos.

Entretanto o real poder desta tecnologia ainda precisa ser explorado. Isto pode ser evidenciado pela comparação feita a esta tecnologia com um "tigre na jaula", no painel JTEC (Feigenbaum et alii, 1994) realizado no Japão (Haynes-Roth e Jacobstein, 1994).

II.9. Conclusão

Este capítulo caracterizou os sistemas baseados em conhecimento, tratando aspectos relativos à sua arquitetura, sua evolução ao longo do tempo, sua aplicabilidade a diversos tipos de problemas e domínios de aplicação e as tendências atuais.

No próximo capítulo abordaremos aspectos relativos ao seu processo de desenvolvimento, isto é, à engenharia de software a ser utilizada em sua construção.

Capítulo III

Engenharia de Software no Desenvolvimento de Sistemas Baseados em Conhecimento

III.1. Introdução

No Capítulo II foram caracterizados os sistemas baseados em conhecimento: sua arquitetura, histórico, aquisição, representação, mecanismos de utilização e tendências atuais.

É fato conhecido que estes sistemas ainda não tem plena aceitação no mercado. Essa realidade deve-se, principalmente, ao fato de que os sistemas baseados em conhecimento não têm, em geral, o nível de qualidade desejado.

Sistemas baseados em conhecimento foram inicialmente construídos de forma "ad-hoc", sem uso de métodos e técnicas específicos. Embora isso sejam características dos momentos iniciais de introdução de uma nova tecnologia, este processo em geral conduz a produtos de baixa qualidade e com tempo de desenvolvimento e custos ultrapassando o inicialmente previsto.

Qualidade em software não se atinge de forma espontânea. É necessário ter-se um processo de desenvolvimento sistemático e adequado, isto é, uma engenharia de software adequada à construção de sistemas com esta tecnologia.

Neste capitulo faz-se um estudo do estado da arte no que se refere à engenharia de software adequada à construção de sistemas baseados em conhecimento.

III.2. Engenharia de Software e Sistemas Baseados em Conhecimento

A engenharia de software é definida como a "aplicação prática do conhecimento científico no projeto e construção de programas e da documentação requerida para desenvolver, operar e manter esses programas" (Boehm, 1980).

O estudo e sistematização do processo de desenvolvimento requerem que se conheçam as características do produto desejado e da tecnologia que será utilizada em seu desenvolvimento. Neste trabalho vamos considerar apenas aspectos relacionados à tecnologia baseada em conhecimento.

A tecnologia baseada no conhecimento pode ser dividida em dois componentes básicos: a engenharia do conhecimento e a programação simbólica (Haynes-Roth, 1984). A engenharia do conhecimento é o processo responsável pela construção do conhecimento no sistema. Procedimentos, estratégias, regras e heurísticas para solução do problema devem ser adquiridos. Além disso, as ferramentas de hardware e software, necessárias ao projeto, devem ser selecionadas para implementação do conhecimento de forma correta e eficiente na base de conhecimentos (Luger e Stubblefield, 1989), (Waterman, 1986), (Gottgtroy, 1990).

Na literatura podem-se encontrar propostas de sistematização do processo de desenvolvimento de sistemas baseados em conhecimento, utilizando o conhecimento atual da engenharia de software no que se refere a sistemas convencionais (Jong, 1988), (Connors, 1992), (Bader et alii, 1988), (Bochsler e Goodwin, 1989), (IEEE/ACM, 1991), (Guida e Tasso, 1994).

Entretanto, algumas diferenças básicas devem ser consideradas, principalmente no que se refere à aquisição permanente do conhecimento, atividade que deve ocorrer durante todo o ciclo de vida do sistema. Outra diferença importante é o fato de que os sistemas convencionais lidam com dados, enquanto os sistemas baseados em conhecimento manipulam conhecimento de forma não procedural.

Sistemas baseados em conhecimento são bem sucedidos na emulação de alguns aspectos do comportamento humano para solução de problemas. Isto se deve, principalmente, a três características básicas (Jong, 1988): a máquina de inferência não tem informação do domínio do problema, a base de conhecimentos guarda conhecimento de como a informação deve ser manuseada, e o conhecimento na base não possui uma ordem pré-determinada para ser armazenado.

A engenharia de software para sistemas baseados em conhecimento deve tentar solucionar alguns problemas provenientes dessas características, considerando a aquisição permanente de conhecimento e o potencial de ineficiência desses sistemas. O processo de busca conduz facilmente a uma explosão combinatória, podendo acarretar numa ineficiência do sistema em obter alguma solução.

A modelagem do conhecimento trata do conhecimento num nível alto de abstração, sendo importante a distinção das diferentes categorias e níveis de conhecimento e sua acomodação em estruturas apropriadas. Essa modelagem é um elemento chave para a engenharia de software da mesma forma que nos sistemas convencionais.

Em Bader et alii (1988) é concluído que o processo de desenvolvimento de sistemas baseados em conhecimento é substancialmente diferente dos sistemas de dados. Entretanto, é necessária, muitas vezes, uma integração entre sistemas destes dois tipos, pois a tendência é o desenvolvimento integrado ou até embutido de sistemas convencionais e inteligentes (Haynes-Roth e Jacobstein, 1994), (Mihaguti, 1995).

Bailor (1992) discute os papéis do engenheiro do domínio, engenheiro do conhecimento e engenheiro de software, ressaltando que o engenheiro do domínio e engenheiro do conhecimento parecem ter muito em comum, pois ambos analisam as áreas do problema, enquanto, o engenheiro de software deve ter conhecimento dessas duas áreas, em função das perspectivas do trabalho.

Num primeiro momento, o engenheiro de software aplica a tecnologia baseada no conhecimento para auxiliar o desenvolvimento das aplicações de software em geral. Numa outra fase, a aplicação dessa tecnologia pode ter a finalidade de desenvolver sistemas baseados em conhecimento capazes de auxiliarem na construção, gerência ou manutenção de software. Para isto podem ser desenvolvidas ferramentas inteligentes. A utilização de assistentes inteligentes para auxiliar os engenheiros de software no seu trabalho é uma perspectiva da área, pois estes assistentes permitem às ferramentas terem um outro tipo de raciocínio, baseado mais na semântica e não na forma estrutural (Bailor, 1992), (Selfridge, 1992), (Selfridge et alii, 1991).

Assim sendo na engenharia de software, ao se definir e propor ambientes de desenvolvimento de software é fundamental se considerar o uso de tecnologias de inteligência artificial. A área engenharia de software baseada no conhecimento trata exatamente da aplicação dessa tecnologia.

O objetivo deste capitulo é descrever o estado da arte relativo à Engenharia de Software para desenvolvimento de sistemas baseados em conhecimento. Deste modo, nas seções que se seguem, serão abordados vários aspectos relativos ao processo de desenvolvimento de sistemas com esta tecnologia: modelos de ciclo de vida, métodos de desenvolvimento, aspectos gerenciais e de avaliação da qualidade.

III.3. Modelos de Ciclo de Vida para Desenvolvimento de Sistemas Baseados no Conhecimento

Modelos de ciclo de vida descrevem as etapas do processo de desenvolvimento de sistemas e as atividades a serem realizadas em cada etapa. A definição dessas etapas e atividades possibilitam prover marcos e pontos de controle para avaliação da qualidade e gerência do projeto (Park et alii, 1991).

O estudo do processo de desenvolvimento provocou o surgimento de várias propostas de ciclo de vida que incluem desde o simples modelo de *codificar e consertar* (Connors, 1992), (Pressman, 1992) até o modelo espiral (Boehm, 1988).

Inicialmente, o desenvolvimento de software era algo feito em pequena escala com equipes pequenas ou, até mesmo, apenas com esforço individual. Neste momento, a ênfase do processo estava na etapa de programação. Neste escopo o desenvolvimento de software caracterizou-se pelo *codificar e consertar*, também chamado de desenvolvimento artesanal ou *ad-hoc*, que consiste em se partir diretamente para a codificação, seguida de correção, sendo esse ciclo repetido até se completar o projeto (Connors, 1992).

O aumento da complexidade e tamanho dos sistemas gerou algumas propostas de ciclo de vida levando em conta o desenvolvimento de sistemas em grande escala envolvendo grandes equipes. Isto acarretou uma mudança de enfoque com ênfase no desenvolvimento de sistemas e não apenas de programas.

A primeira proposta deu origem ao *modelo tradicional* ou cascata. Nesse modelo as fases são executadas sistematicamente de forma seqüencial.

O *modelo evolucionário* surgiu propondo um desenvolvimento que expandisse o sistema gradativamente, permitindo que se obtivessem modelos do comportamento do software antecipadamente, os denominados protótipos. A prototipagem é uma forma de desenvolvimento incremental e contém quatro tipos diferentes: ilustrativo (telas), simulado (acesso ao banco de dados é simulado), funcional (subconjunto limitado) e evolucionário (começa pequeno e cresce). Os três primeiros tipos são construídos para se atingir objetivos propostos a priori, sendo considerados protótipos descartáveis. O protótipo evolucionário se tornará ao final um produto operacional.

Outra forma de desenvolvimento é caracterizada pelo *modelo de síntese automática* que transforma as especificações em programas (Balzer, 1981), (Boehm, 1988), (Bersoff e Davis, 1991), (Connors, 1992).

O *modelo em espiral* proposto por Boehm (1988) mostrado na Figura III.1, fornece uma estrutura de trabalho para a produção de software baseada em processo e níveis de risco permitindo a análise dos riscos em várias etapas do desenvolvimento. Esse modelo pode ser considerado um meta-modelo pois pode abranger diferentes outros modelos, desde o modelo cascata até os vários tipos de protótipos. É um modelo cíclico.

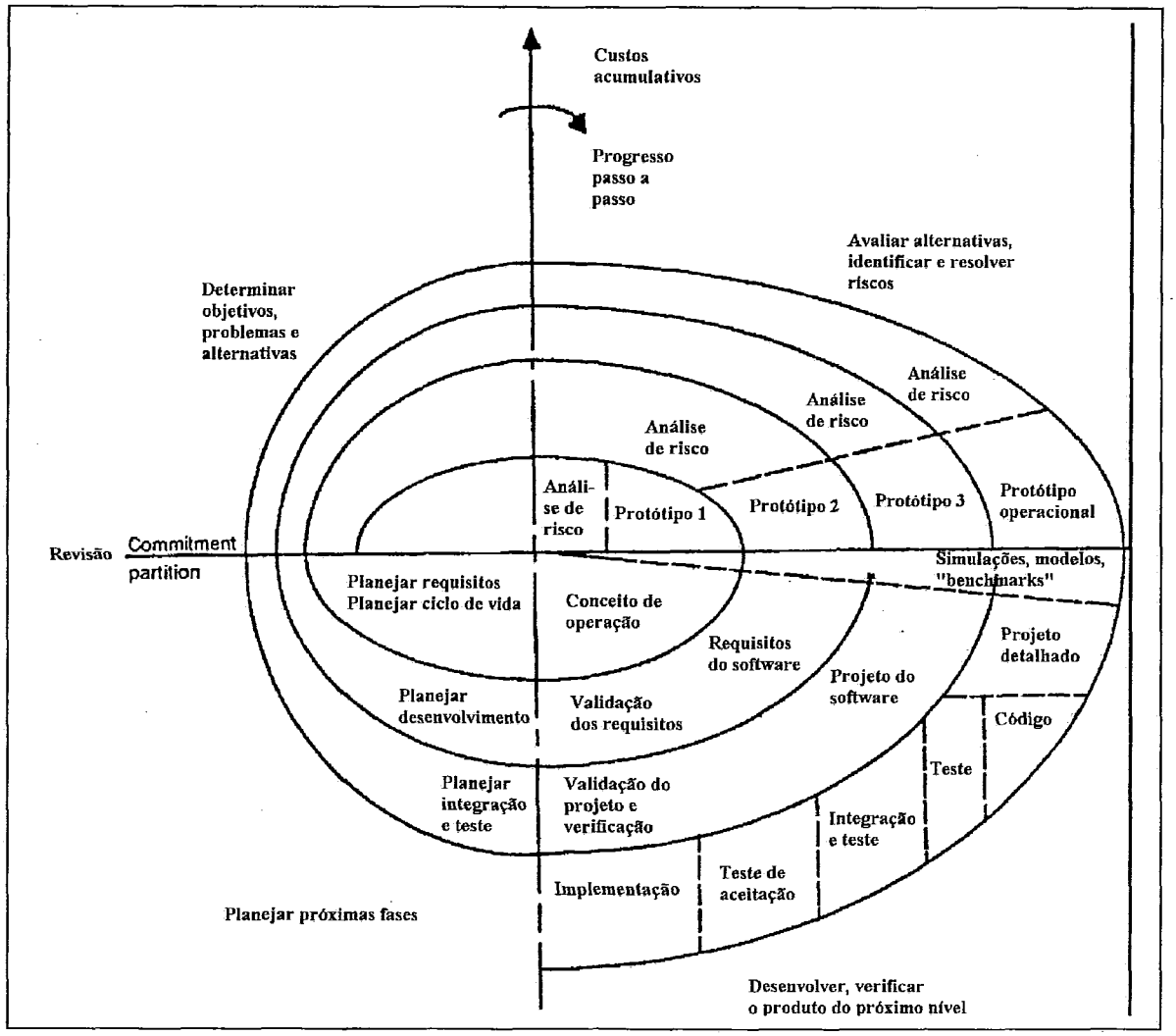


Figura III.1 - Modelo Espiral (Boehm, 1988)

A análise dos modelos de ciclo de vida pode ser feita considerando-se a identificação monolítica ou incremental do processo de desenvolvimento. Na estrutura de trabalho monolítica, os detalhes de cada fase são completados em sua totalidade, antes do início

da etapa seguinte. Sendo assim, o comportamento do software só pode ser avaliado no final do desenvolvimento. Na estrutura incremental, os detalhes podem ser retardados com o benefício de se produzir antecipadamente um protótipo mostrando o funcionamento do produto (Connors, 1992).

A literatura em Engenharia de Software para Sistemas Baseados em Conhecimento apresenta uma série de modelos de ciclo de vida, definindo etapas, produtos e estratégias gerenciais. Evidencia-se a necessidade de modelos adequados ao desenvolvimento de sistemas com esta tecnologia, pois as características dos sistemas baseados em conhecimento fazem com que o uso de alguns modelos de ciclo de vida tenham difícil aplicação. O modelo em cascata é um exemplo, pois é difícil de se obter a especificação completa do comportamento de um Sistema Baseado em Conhecimento no começo do projeto. A natureza interativa na obtenção do conhecimento adicionada à complexidade de validação e testes no comportamento inteligente desses sistemas são outros fatores que dificultam o uso do modelo em cascata para sistemas baseados em conhecimento.

Na literatura foram encontradas várias propostas de modelos de ciclo de vida (Connors, 1992), (Hull e Kay, 1991), (Fenn e Veren, 1991), (Park et alii, 1991), (Garrett, 1991), (Perot, 1991), (Arora e Cooke, 1991), (Bader et alii, 1988), (Jong, 1988), (Matsumoto, 1989), (Sacerdoti, 1991), (Weitzel e Kerschberg, 1989), (Weitz e Meyer, 1990), (Guida e Tasso, 1994), (Juristo e Pazos, 1993) podendo se verificar que a maioria utiliza a prototipagem e alguns usam, também, o modelo espiral.

Nessas propostas, normalmente, o uso de prototipagem é abordado, embora algumas considerem que o protótipo é apenas uma técnica para auxiliar na análise e projeto do sistema. O importante são as lições e o *feedback* obtidos através dele e não o protótipo em si. Por isso, em alguns ciclos de vida, essa fase não é especificada claramente, pois é somente uma forma de investigação.

Fenn e Veren (1991) fizeram uma pesquisa na indústria e departamentos governamentais americanos e verificaram que todas, exceto uma empresa, utilizavam até aquele momento, a prototipagem como modelo de ciclo de vida para desenvolvimento de sistemas baseados em conhecimento. Na maioria dos casos são protótipos incrementados ao longo do ciclo, onde o conhecimento e a funcionalidade são adicionados até se obter um sistema operacional satisfatório.

Nessa mesma pesquisa foram, ainda, identificadas quatro estruturas de trabalho. A primeira constroe, em poucas semanas, um pequeno protótipo inicial que apresenta

algumas funcionalidades úteis para os usuários e gerentes. O desenvolvimento completo é feito após essa avaliação.

A segunda estrutura é semelhante à primeira, embora envolva um protótipo maior, desenvolvido em alguns meses, que apresenta funcionalidade completa em algumas áreas, podendo ser avaliado, diretamente, pelos usuários. O sistema completo é desenvolvido com base no *feedback* do usuário e do especialista.

Na terceira são definidas algumas etapas, tais como, protótipo, sistema piloto e sistema funcional. Neste caso, a funcionalidade do sistema é expandida gradualmente e a sistemática consiste no desenvolvimento inicial de um protótipo restrito aos casos normais, seguindo-se do aumento da funcionalidade e sua possível expansão em até três protótipos. Essas novas versões tratam de casos de exceção até se obter um produto robusto e integrado.

A quarta estrutura desenvolve alguns protótipos em paralelo sendo que cada um investiga uma questão do problema.

Perot (1991) faz uma análise das propostas de estruturação do processo de desenvolvimento dos sistemas baseados em conhecimento caracterizando-as em três grupos. A primeira classe é baseada em protótipos e não considera questões de gerência do projeto, sendo que o desenvolvimento das aplicações depende totalmente do progresso dos protótipos. No segundo grupo, estão os modelos de ciclo de vida que combinam a etapa de aquisição do conhecimento com o modelo tradicional, como no KADS-(Knowledge Aquisition and Design Structure) (Vob e Karbach, 1993) e MIXAGE[1] (Perot, 1991). E por último as estruturas em que as fases do desenvolvimento estão baseadas somente no processo de aquisição do conhecimento, exemplo KOD de modelos de ciclo de vida (Perot, 1991) e a proposta de Scott et alii (1991).

Um dos primeiros modelos de ciclo de vida para Sistemas Baseados em Conhecimento que serviu de base para diversas outras propostas foi o modelo proposto por Buchanan at alii (1983). Esse ciclo consiste de cinco fases altamente interdependentes e que se sobrepõem, não tendo uma seqüência bem definida. O produto é desenvolvido de forma incremental. A figura III.2 mostra este modelo de ciclo de vida.

[1]MIXAGE é uma metodologia que combina duas outras: MERISE e KADS.

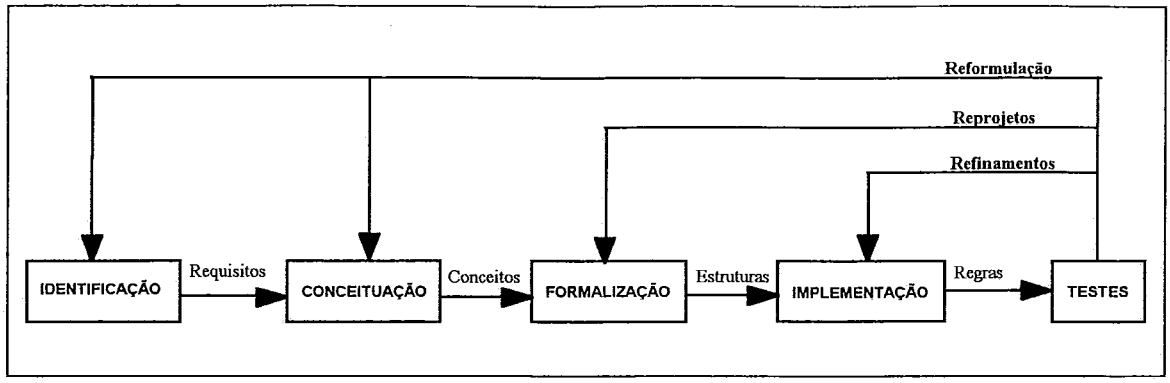


Figura III.2 - Modelo de Ciclo de Buchanan et alii (1983)

Em Guida e Tasso (1994) são definidas e discutidas fases, atividades e tarefas do ciclo de vida de sistemas baseados em conhecimento. A proposta para este modelo está baseada em três fases: análise, desenvolvimento e operação. A fase de análise corresponde à análise de oportunidade e de viabilidade. O desenvolvimento engloba as fases de construção de um protótipo demonstrativo, de desenvolvimento de outros protótipos e construção do sistema final. A fase de operação consiste na manutenção e geração de novas versões do sistema. Esta proposta é interessante, pois discute aspectos gerais de construção, avaliação da qualidade e gerência de sistemas baseados em conhecimento, servindo como um guia para o desenvolvimento desses sistemas.

Em Park et alii (1991) é definido um modelo de ciclo de vida estruturado baseado em prototipagem. Nele combinam-se o modelo tradicional denominado estruturado e o uso de protótipos através da definição de uma atividade de prototipagem no lugar da fase de análise de requisitos. Essa etapa de prototipagem é subdividida e sua saída pode ser para diferentes etapas do ciclo de vida. Essa proposta é bem simples e utiliza prototipagem para definir os requisitos do sistema.

O ciclo de vida do ESDM ("Expert System Development Methodology"), metodologia para desenvolvimento de sistemas especialistas desenvolvida no centro *Goddard Space Flight* da NASA, combina o modelo proposto por Buchanan et alii (1983) com o modelo espiral de Boehm (1988). As cinco fases do modelo de Buchanan formam um ciclo da espiral sendo que o produto gerado em cada ciclo corresponde a um tipo diferente de protótipo conforme mostra a Figura III.3 (Hull e Kay, 1991).

O modelo proposto pelo ESDM prevê, assim, um ciclo incremental de desenvolvimento do sistema com cinco estágios de prototipagem: protótipo de demonstração de viabilidade, protótipo de pesquisa, protótipo de campo, protótipo de

produção e protótipo operacional. A partir desses ciclos é definida a transição para o modelo tradicional de desenvolvimento, quando a viabilidade e os riscos de uso ficarem reduzidos a níveis aceitáveis.

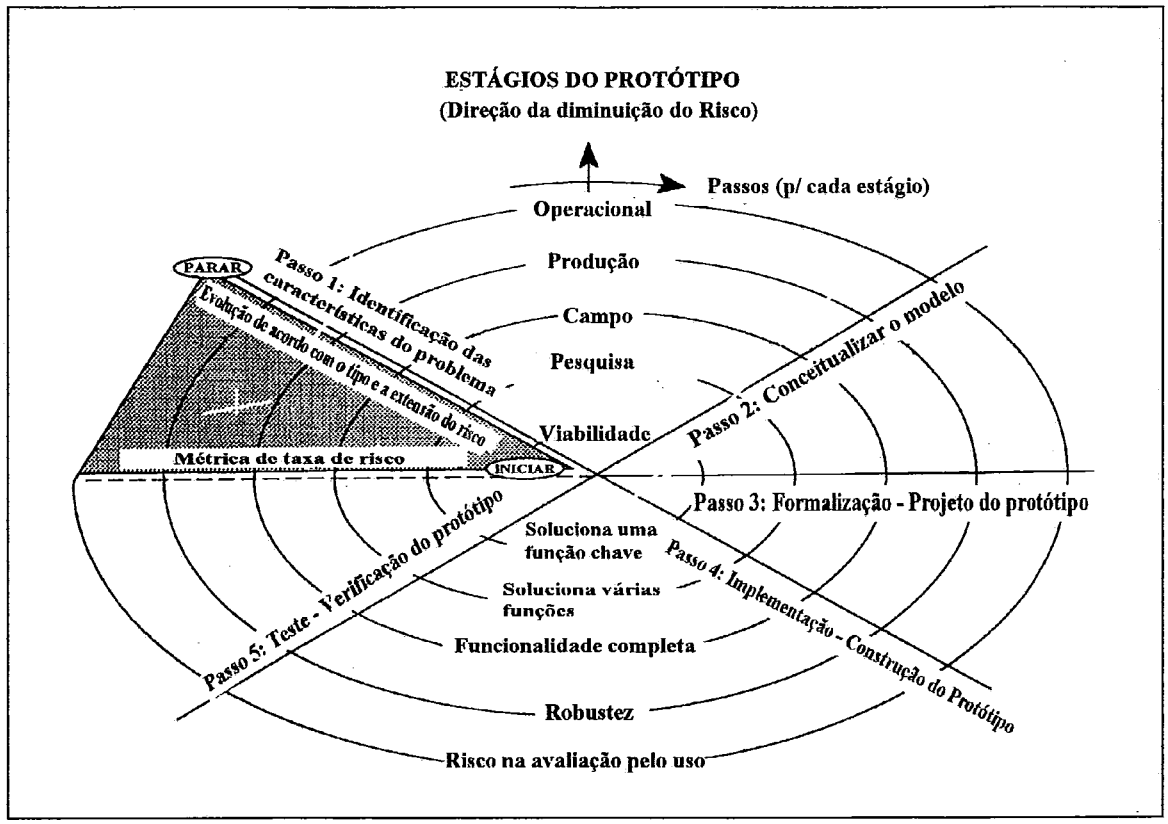


Figura III.3 - Ciclo de Vida do ESDM (Hull e Kay, 1991)

Um outro enfoque é o paradigma proposto pelo modelo POLITE que está baseado em dois outros ciclos: o modelo tradicional e o modelo RUDE. Considerando que o desenvolvimento de sistemas em Inteligência Artificial partem do modelo RUDE *Run-Understand-Debug-Edit*, ou seja, rodar o sistema, entendê-lo, depurá-lo e editar uma nova versão, é proposto o ciclo de vida POLITE, mostrado na Figura III.4. Esse modelo utiliza em suas fases o modelo RUDE combinado ao modelo tradicional (Bader et alii, 1988).

POLITE é dividido em seis fases, sendo estas fases também sub-divididas. De um lado tem-se o desenvolvimento tradicional e de outro, os elementos da tecnologia baseada no conhecimento. Esse modelo de ciclo de vida visa atender aos objetivos de desempenho desde o início. Estes servem de base para avaliação e validação de cada estágio do ciclo onde o desenvolvimento RUDE pode ser apropriado ou não.

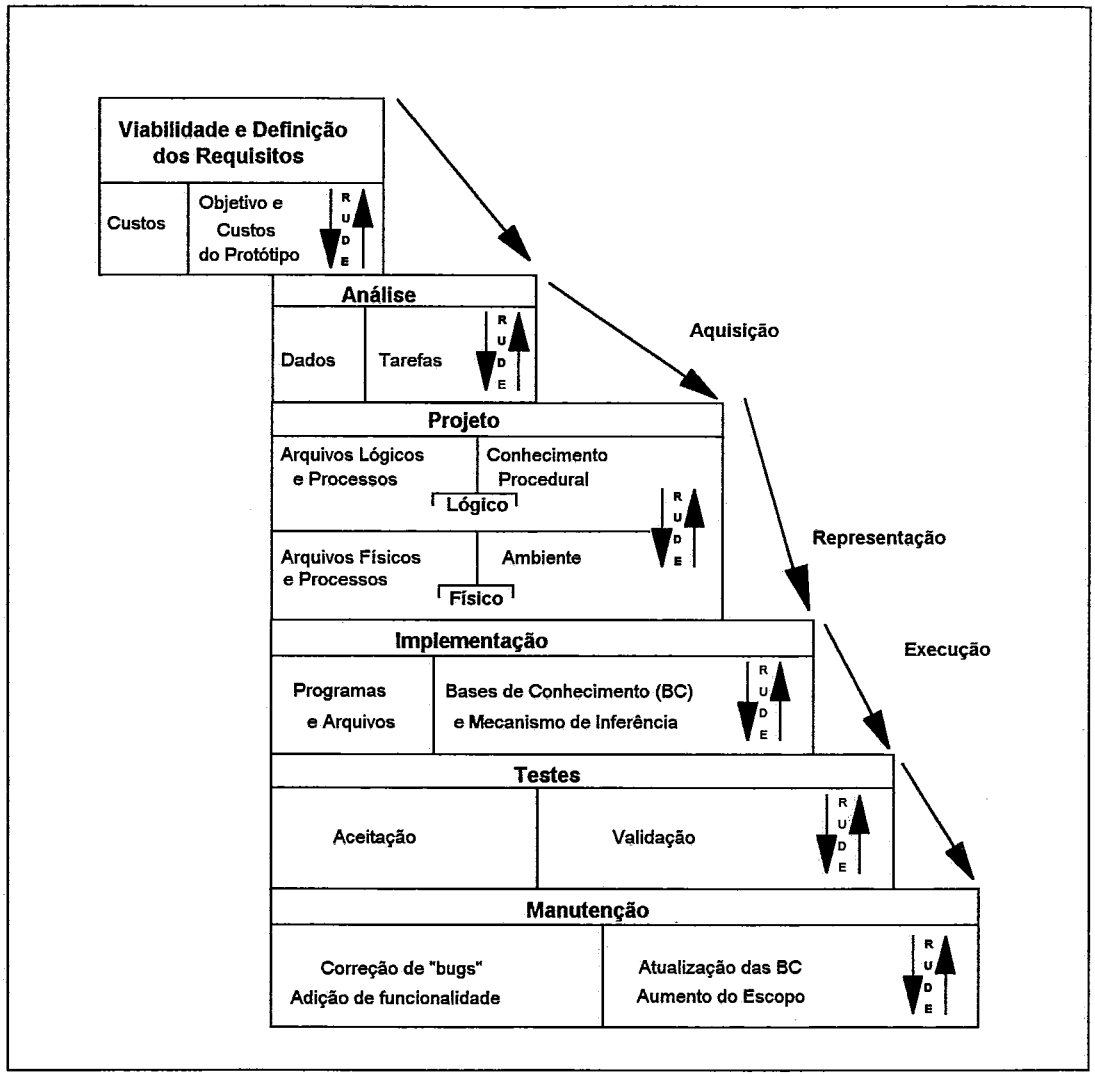


Figura III.4 - Modelo de Ciclo Vida POLITE (Bader et alii, 1988)

A metodologia KADS, que será descrita na seção III.4, é uma proposta para o desenvolvimento de sistemas baseados em conhecimento. O ciclo de vida proposto no KADS é um enfoque baseado em estágios que incorpora aspectos da engenharia de software ao especificar o modelo de ciclo de vida com ferramentas e técnicas para engenharia do conhecimento (Breuker e Wielinga, 1988), (Wielinga et alii, 1988), (Schreiber, 1992), (Hickman et alii, 1989). As etapas do ciclo de vida do KADS estão baseadas nas transformações e nos níveis de abstração do processo de desenvolvimento de um sistema baseado em conhecimento (Figura III.5), considerando que o sistema passa por cinco níveis: lingüístico, conceitual, epistemológico, lógico e

implementacional. A prototipagem, no KADS, é considerada fora do ciclo de desenvolvimento. É vista como uma técnica para auxiliar a análise e projeto do sistema. O desenvolvimento de protótipos permite um procedimento cíclico dentro de um mesmo ponto no ciclo de vida (Hickman et alii, 1989).

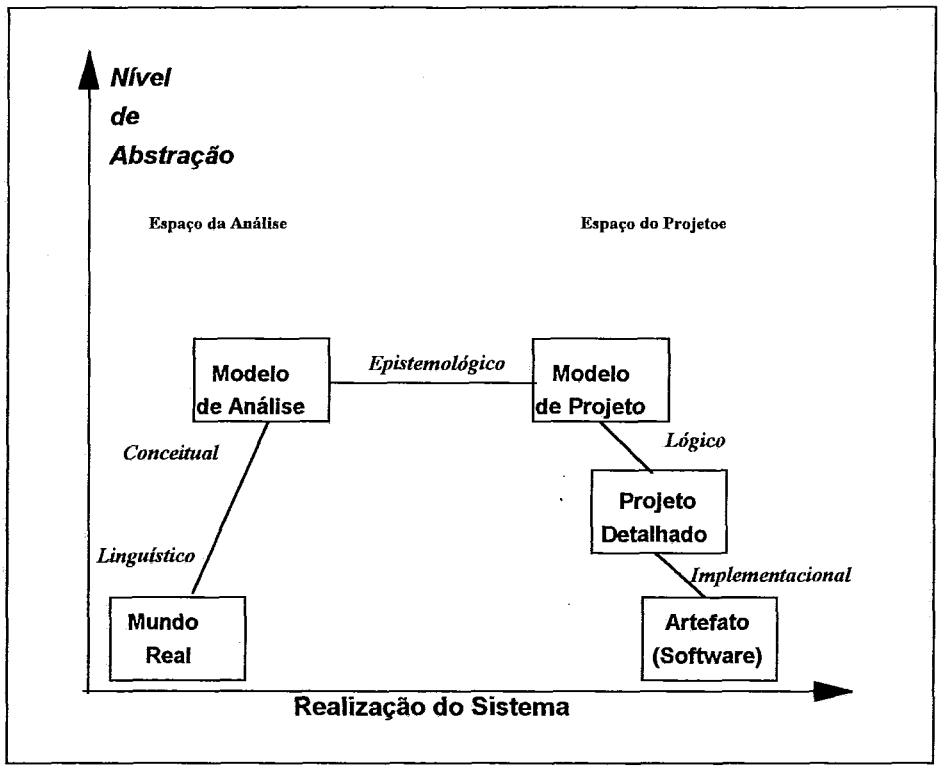


Figura III.5 - Ciclo de Vida do KADS (Breuker e Wielinga, 1988)

O modelo CRYSTAL, proposto em Perot (1991), busca unir os benefícios do modelo espiral e da prototipagem. Do modelo espiral é ressaltada a importância da análise de riscos. Além disso, as estruturas de prototipagem horizontal e vertical[1] aliadas às técnicas de aquisição do conhecimento, fornecem um desenvolvimento flexível com especificações executáveis (Perot, 1991).

No modelo CRYSTAL é possível iniciar e desenvolver, em paralelo, diferentes módulos definidos na fase de análise dos requisitos, que serão progressivamente,

[1]A prototipagem horizontal tem como objetivo prototipar o nível geral do sistema, enquanto que a prototipagem vertical detalha uma parte específica do sistema.

integrados até se chegar à aplicação final. A idéia deste processo de desenvolvimento é definir a parte central da base de conhecimento e depois imaginar o sistema como um cristal num ambiente sem muitas limitações, permitindo abordar diferentes questões, tais como, o despacho de tarefas em diferentes linguagens e tecnologias (sistema especialista, multimídia, banco de dados, etc...). A Figura III.6 apresenta a arquitetura do SBC neste modelo.

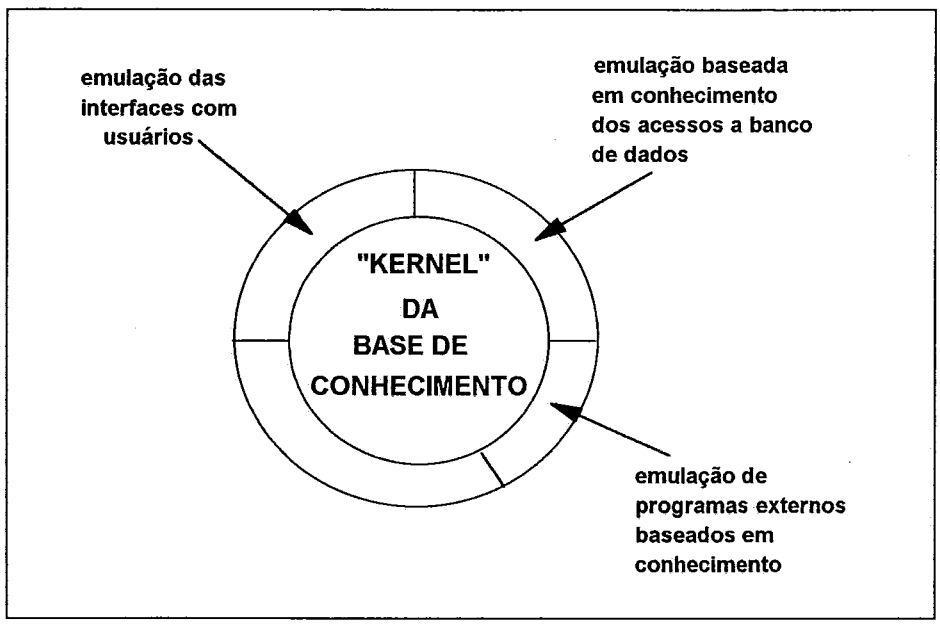


Figura III.6 - Arquitetura do SBC no Modelo CRYSTAL (Perot, 1991)

O modelo de ciclo de vida para sistemas baseados em conhecimento proposto por Arora e Cooke (1991) se compõe de sete fases principais que tem a possibilidade de serem aplicadas de forma interativa e incremental. Prevê um desenvolvimento cíclico, onde novas versões do sistema surgem como uma evolução, permitindo assim um desenvolvimento incremental. Por sua vez, cada uma das sete fases pode ser executada de forma cíclica, constituindo um mini-ciclo, como mostra a Figura III.7. Cada mini-ciclo consiste num desenvolvimento interativo até se atingir com sucesso o desenvolvimento completo da etapa.

Os mini-ciclos do modelo de Arora e Cooke (1991) podem ser executados várias vezes, pois cada fase pode seguir ou retornar às fases anteriores e o ciclo maior, que é constituido pela sequência de todas as fases, pode ser dividido em três grupos ou marcos no desenvolvimento. O primeiro grupo compreende as quatro primeiras fases e cada uma pode voltar no máximo até duas fases anteriores, sendo sua conclusão marcada pela

obtenção de módulos codificados e testados individualmente. Outro grupo é constituido pelas fases de integração e testes de aceitação, que têm como marco a implantação de uma nova versão do sistema. Concluído esse conjunto de etapas o sistema vai para a fase de operação onde a mudança para outra atividade só pode ser efetuada através de um novo grande ciclo com o desenvolvimento de uma nova versão do sistema.

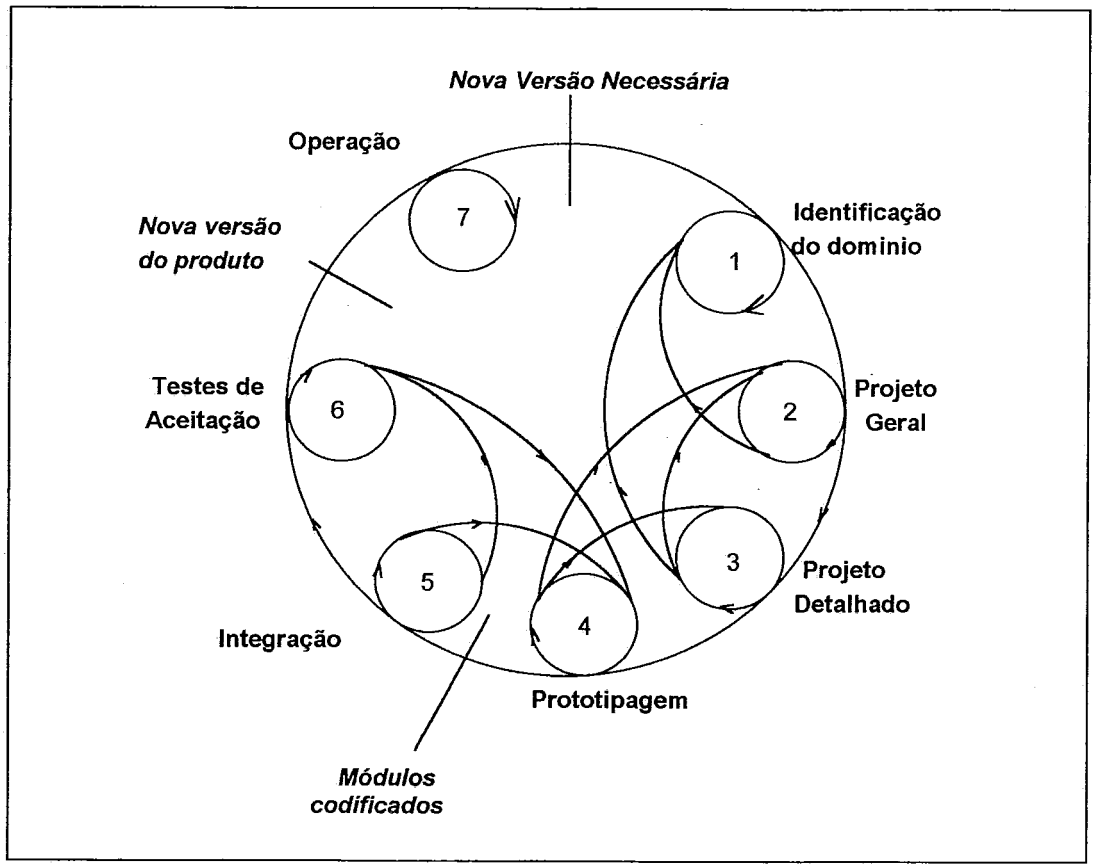


Figura III.7 - Modelo de Ciclo de Vida de Arora e Cooke (1991)

Juristo e Pazos (1993) propõem uma modificação no modelo de ciclo de vida espiral para o desenvolvimento de sistemas baseados em conhecimento, introduzindo mais uma dimensão a este. O modelo denominado espiral cônico é o modelo espiral em três dimensões, onde a terceira dimensão corresponde à manutenção adaptativa, ou seja, à incorporação de novos conhecimentos ao sistema. Este modelo parte do princípio que conhecimento gera sempre novos conhecimentos, sendo estes adaptados ao sistema na forma de um cone, pois o conhecimento tende a aumentar. Neste ciclo são previstos o desenvolvimento de diferentes protótipos até se obter a versão operacional, sendo que

em cada uma dessas versões podem ser realizadas a manutenção adaptativa para incorporar novos níveis de conhecimento.

III.3.1. Análise dos Modelos de Ciclo de Vida para SBC

Nos modelos de ciclo de vida descritos anteriormente pode-se verificar que cada um introduz aspectos que são bastante proveitosos no desenvolvimento de sistemas baseados em conhecimento.

Nesta seção serão analisados os modelos de ciclo de vida descritos anteriormente, agrupando-os nas seguintes categorias: combinação do modelo cascata com algumas propostas específicas consideradas para atender as características dos sistemas baseados em conhecimento, modelo espiral com alterações, que enfatizam os atributos desses sistemas, e protótipos paralelos.

A primeira categoria agrupa modelos que se caracterizam por ser uma modificação do modelo cascata. Dos ciclos de vida estudados, estão neste grupo os modelos POLITE/RUDE e KADS.

O modelo de ciclo de vida POLITE fornece um padrão de desenvolvimento de sistemas baseados em conhecimento utilizando modelos de ciclo de vida já consagrados. Com a combinação dos dois modelos, RUDE e cascata, o POLITE controla o desenvolvimento incremental através do emprego de um modelo determinístico e um modelo interativo. Esta combinação é interessante, pois admite o desenvolvimento em conjunto de sistemas, utilizando a tecnologia convencional e a tecnologia baseada no conhecimento.

O modelo de ciclo de vida do KADS é importante, principalmente, por ser esta uma metodologia bastante referenciada e de grande aceitação para o desenvolvimento de sistemas baseados em conhecimento e também por ser uma adaptação do modelo em cascata. Enfatiza as características da engenharia do conhecimento. Outro fator importante é a adaptação dos níveis do modelo de ciclo de vida do KADS ao modelo espiral, que está sendo proposto na nova versão desta metodologia.

O segundo grupo de modelos agrupa ciclos de vida que podem ser caracterizados como espiral modificado. Estão neste caso o modelo ESDM, a proposta de Arora e Cooke (1991) e o modelo espiral cônico.

O aspecto interessante do modelo de ciclo de vida proposto no método ESDM é a flexibilidade e a definição da transição dos produtos durante cada etapa. A combinação dos dois modelos, identificando os diferentes protótipos, enfatiza o gerenciamento do processo de desenvolvimento, através da gerência de riscos, do direcionamento do desenvolvimento e da validação dos requisitos do sistema. A transição para o modelo tradicional de ciclo de vida pode ser feita ao se obter níveis de viabilidade e riscos aceitáveis.

Analisando a sistemática proposta por Arora e Cooke (1991) verifica-se que concluídas as fases de identificação do domínio e projeto geral, o processo segue numa seqüência de desenvolvimentos incrementais próprios das fases de projeto detalhado e prototipagem. A fase de integração pode ser realizada também incrementalmente. Já as etapas de operação e testes de aceitação, por sua natureza, não podem ser feitas incrementalmente e tratam do sistema como um todo.

No modelo de ciclo de vida de Arora e Cooke (1991) é claramente definida a distinção de desenvolvimento incremental e interativo pois o primeiro envolve uma tarefa diferente e não a repetição da mesma. Essa proposta de Arora e Cooke (1991) é interessante pois sistematiza o desenvolvimento de sistemas baseados em conhecimento, adotando um modelo cíclico e definindo as atividades e seus produtos e estabelecendo, também, formas de avaliação em cada uma das fases. Este ciclo corresponde à uma descrição mais detalhada que as demais. Embora a avaliação seja definida de forma simplificada, ela fornece uma noção dos pontos de verificação e validação e dos produtos a avaliar em cada fase. São, também, esboçadas métricas para avaliação, que são discutidas em Oliveira (1995).

O modelo espiral cônico é uma adaptação do modelo espiral, sendo que no nível plano (duas dimensões) é similar ao modelo proposto no método ESDM. A novidade é a terceira dimensão, que trata do problema de se introduzir novos conhecimentos ao sistema, sendo algo que não pode ser esquecido em sistemas baseados em conhecimento, pois a aquisição do conhecimento é permanente ao longo de toda vida do sistema.

A terceira classe de modelos se caracteriza pela construção de protótipos múltiplos. É o caso do modelo CRYSTAL.

A filosofia adotada no modelo de ciclo de vida CRYSTAL tem como principal vantagem iniciar e desenvolver em paralelo diferentes módulos. Esses módulos são definidos e desenvolvidos durante a fase de análise dos requisitos e integrados, progressivamente, até se obter a aplicação final.

O modelo de ciclo de vida CRYSTAL permite a pesquisa separada de diferentes questões, podendo estimular o uso de diferentes paradigmas e centra o desenvolvimento no núcleo da base de conhecimentos. Esse modelo, entretanto, não especifica claramente as etapas de desenvolvimento, podendo ser considerado mais uma filosofia de desenvolvimento, que pode ser utilizada com aplicação das etapas de outros modelos de ciclo de vida. A ênfase é colocada no desenvolvimento em paralelo de protótipos, a partir da obtenção de uma parte fundamental da base de conhecimento.

A pesquisa desenvolvida por Fenn e Veren (1991) encontrou esta estrutura de desenvolvimento de protótipos em paralelo, nas empresas, mostrando que esta pode ser uma solução para o desenvolvimento de sistemas baseados em conhecimento, principalmente quando este está integrado a sistemas convencionais. O desenvolvimento baseado no modelo CRYSTAL pode ser bastante útil, principalmente, em domínios de aplicação pouco explorados pela tecnologia baseada no conhecimento.

III.4. Métodos de Desenvolvimento para SBC

Métodos em Engenharia de Software, segundo Charette (1986) são: "prescrições explícitas para a realização de uma atividade ou conjunto de atividades do modelo de ciclo de vida". Eles devem refletir um conjunto de diretivas para a aplicação sistemática de técnicas e instrumentos. As técnicas podem ser definidas como um conjunto de princípios para execução de uma tarefa específica do processo de desenvolvimento do software. Os instrumentos tornam possível o uso de métodos e técnicas (Rocha et alii, 1987), (Crispim e Rocha, 1992) .

Na literatura de engenharia de software (Ross, 1977), (Yourdon e Constantine, 1978), (Page-Jones, 1980), (Chandersekaran e Linder, 1981), (Jackson, 1983), (Ward e Mellor, 1985), (Booch, 1986), (Warnier, 1986), (Chen, 1987), (De Marco, 1989), (Jones, 1990), (Yourdon, 1990), (Hatley e Pirbhai, 1991), (Coad, 1992), encontramos diversas propostas de métodos que podem atender diferentes paradigmas (dados, funções, objetos), para diversos domínios de aplicação, com rigor de expressão diferente e aplicação em fases específicas.

Nesta seção serão considerados, apenas, métodos que auxiliam no processo de desenvolvimento para o caso particular dos sistemas baseados em conhecimento, objeto deste trabalho. Na literatura técnica existem algumas propostas de metodologias de desenvolvimento, tais como KADS (Schreiber, 1993), VITAL (Shadbolt et alii, 1993),

TBSM (Yen e Lee, 1993), ESDM (Hull, 1991), ES/SDEM (Matsumoto, 1989), POLITE (Bader et alii, 1988).

Em Guida e Tasso (1994) são identificadas também algumas metodologias desenvolvidas internamente em empresas ou organizações governamentais (Tecktronix, DEC, Departamento de Defesa Americano, ..). Entretanto, muitas dessas metodologias descritas se restringem a definir um modelo de ciclo de vida com uma descrição suscinta de modelos, não detalhando os conceitos e métodos envolvidos nem o processo de construção desses modelos. Muitas dessas propostas estão ainda em desenvolvimento, o que acarreta numa dificuldade na aplicação e entendimento do processo de modelagem.

Foram encontrados, também, métodos especializados na construção de sistemas baseados em conhecimento que, em geral, enfatizam engenharia do conhecimento, pois esta é fundamental nesse desenvolvimento. O método de especialidade da metodologia KADS, o método estruturado proposto por Keller (1987), o desenvolvimento de caixas estruturadas (Basu e Hevner, 1991), a forma estruturada de formalizar o conhecimento (Merlevede, 1991), os modelos baseados em tarefas (Yen e Lee, 1993), (Musen e Tu, 1993), (Vestli et alii, 1994) são exemplos de propostas de métodos com o intuíto de orientar o processo de aquisição, análise e especificação de sistemas baseados em conhecimento.

Foram encontradas, também, experiências e propostas de utilização de orientação a objetos como estrutura de trabalho para esse desenvolvimento ou sua aplicação a determinada fase do ciclo de vida (Duff e Carlson, 1991), (Schaschinger, 1992), (Czedjo et alii, 1993), (Sierhuis, 1991), (Ito, 1991), (Cuena, 1993).

O método desenvolvido por Keller (1987) é aplicado nas fases iniciais do processo de desenvolvimento de sistemas baseados em conhecimento, enfatizando a definição da base de conhecimentos no processo de aquisição. Este utiliza um diagrama de fluxo de dados, que auxilia a esquematizar o processo de decisão de um especialista. A partir dos possíveis resultados, define as decisões e os fatores que influenciam a decisão. Esses diagramas são detalhados num dicionário de dados sendo especificados, para cada decisão, os valores dos possíveis resultados e dos fatores que influenciam a decisão. A partir desse dicionário de dados são definidas as regras pelas quais se chegam aos resultados, partindo dos fatores.

O método Keller utiliza a técnica *top-down* para decompor o problema inicial em sub-problemas, através do detalhamento sistemático dos diagramas de fluxo de dados, de forma análoga ao utilizado pelo método de Análise Estruturada (DeMarco, 1989). Em

Werneck et alii (1989), Werneck e Rocha (1994), Passos (1994) e Ribeiro (1989) encontram-se exemplos de modelagens de sistemas utilizando esse método.

A metodologia KADS é o resultado de uma pesquisa desenvolvida em um projeto ESPRIT da comunidade européia, com número de contrato P1098. É uma proposta para o desenvolvimento e construção de sistemas baseados em conhecimento, prevendo a geração dos seguintes modelos: (Schreiber, 1992), (Schreiber, Wielinga e Breuker, 1993):

- *Modelo Organizacional*: analisa o ambiente sócio-organizacional em que o sistema baseado em conhecimento irá funcionar;
- *Modelo da Aplicação*: define o escopo do problema a ser tratado pelo sistema, suas funções e restrições externas;
- Modelo de Tarefas: especifica como a função do sistema é realizada através das tarefas executadas pelo sistema;
- Modelo de Cooperação: descreve as tarefas do modelo de tarefas que necessitam um esforço cooperativo, isto é, define como deverão ser distribuídas as tarefas entre o sistema e os agentes externos (usuários) e como as tarefas com envolvimento conjunto serão executadas;
- *Modelo de Especialidade*: atividade central do desenvolvimento de um sistema baseado em conhecimento e tem como objetivo especificar o conhecimento necessário para executar as tarefas associadas ao sistema.
- *Modelo Conceitual:* é composto do *modelo de cooperação* e *modelo de especialidade*, sendo estes independentes da implementação. Este modelo resulta no modelo de resolução do problema, agregando esses dois modelos;
- *Modelo de Projeto:* descreve o sistema baseado nas técnicas computacionais e de representação do conhecimento.

Para apoiar o desenvolvimento de sistemas baseados em conhecimento, KADS baseia-se em cinco princípios: múltiplos modelos, modelagem do conhecimento em quatro camadas através do Modelo de Especialidade, reutilização de componentes genéricos, diferenciação do conhecimento através do refinamento de modelos (a partir de um modelo genérico simples pode-se construir modelos mais complexos) e projeto com preservação da estrutura do Modelo de Especialidade. A Figura III.8 apresenta a estrutura dos múltiplos modelos do KADS.

O modelo de especialidade do KADS está definido em detalhes no Anexo VI.

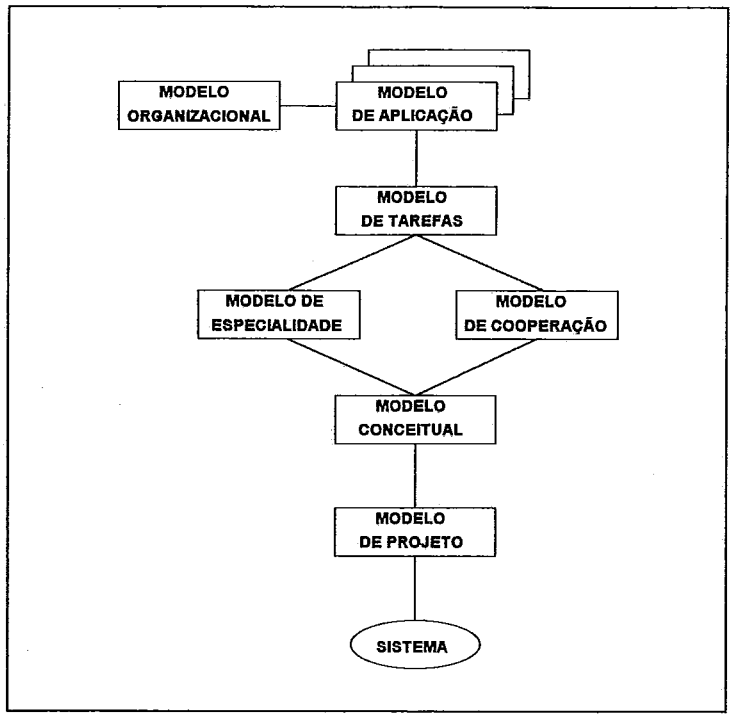


Figura II. 8 - Estrutura dos Modelos na Metodologia KADS (Screiber et alii, 1993)

O método de projeto orientado a objetos proposto por Sierhuis (1991) é baseado em dois paradigmas conhecidos: a orientação a objetos e as quatro camadas de conhecimento do Modelo de Especialidade KADS.

O paradigma de orientação a objetos é, facilmente, introduzido nas três primeiras camadas do conhecimento, isto é, na de domínio, inferência e tarefas. A camada de estratégia, na prática, é bastante discutida, pois sua utilidade não é muito clara e a maioria das aplicações que utilizaram o Modelo de Especialidade não a usaram. O benefício de se modelar as quatro camadas em objetos físicos é transformar o modelo de conhecimento num modelo fácil de ser implementado. A orientação a objetos é utilizada para modelagem de projeto. Para isso define-se uma proposta de mapeamento em objetos, do modelo de conhecimento dos diferentes níveis. Entretanto, algumas relações no nível de domínio não têm formas de expressão em objetos, o que significa que este paradigma tem problemas de expressividade na representação do conhecimento.

Este método proposto por Sierhuis (1991) fornece um modelo geral em orientação a objetos, dos conceitos tratados pelo paradigma de quatro camadas. A ferramenta CAKE ("Computer Aided Knowledge Engineering") para auxiliar o engenheiro do conhecimento no uso desse método está sendo desenvolvida. Essa ferramenta faz parte

de um ambiente de engenharia do conhecimento, o CAKEE ("Computer Aided Knowledge Engineering Environment").

A metodologia TBDM (Yen e Lee, 1993) define um modelo de sistemas baseados em conhecimento através de diferentes níveis de abstração, utilizando modelos de especificação e processos de especificação. Os modelos de especificação incluem o modelo dos objetos do domínio e modelo de estado relativo ao processo de solução de problemas. O processo de especificação é, também, composto de duas partes: especificação funcional que descreve a função da tarefa e especificação comportamental que utiliza os estados das tarefas para descrever a seqüência de tarefas e a interação entre elas em diferentes níveis. Esta metodologia é baseada em especificações formais, que permitem a verificação da consistência e completude desta modelagem.

A metodologia VITAL (Shadbolt et alii, 1993), também, faz parte de uma pesquisa desenvolvida em um projeto ESPRIT da comunidade européia, sendo baseada na metodologia KADS. Entretanto esta prevê, somente, quatro produtos:

- *Especificação de Requisitos*: descreve as funções esperadas pela aplicação em desenvolvimento e sua limitações;
- *Modelo Conceitual*: descreve as entidades do domínio do problema, estruturas de tarefas e comportamento especialista para a solução do problema;
- *Modelo de Projeto*: contém o modelo funcional do projeto, independente da implementação e o mapeamento técnico do projeto que transforma o modelo funcional num código executável;
- *Código Excutável*: é o produto final, os componentes do software.

Além desses produtos, a metodologia VITAL prevê os seguintes componentes: metodologia de engenharia do conhecimento, modelo de ciclo de vida e configuração de ciclo de vida. Atualmente este projeto está em fase de testes, sendo desenvolvido um sistema com esta proposta e ferramentas de suporte.

A estrutura de trabalho MIKE ("Model-based and Incremental Knowledge Engineering") prevê métodos e princípios para o desenvolvimento de sistemas baseados em conhecimento (Angele et alii, 1993). MIKE baseia-se em quatro princípios: engenharia do conhecimento baseada em modelos, engenharia do conhecimento como um processo incremental, ponte para evitar a perda de representação e reutilização de modelos. Esta estrutura utiliza o modelo de especialidade do KADS definido em três

camadas (domínio, inferência e tarefas). A especificação completa deste modelo é feita na linguagem formal KARL que permite operacionalizar o sistema.

A metodologia CommonKADS (Velde, 1994), (Schreiber et alii, 1994) é uma nova versão da metodologia KADS, prevendo o gerenciamento do projeto, análise organizacional, engenharia do conhecimento e engenharia de software para sistemas baseados em conhecimento. Utiliza o modelo espiral proposto por Boehm (1988) para o gerenciamento do projeto, definindo os seguintes modelos: modelo de organização, modelo de tarefas, modelo de especialidade, modelo de comunicação, modelo de agentes e modelo de projeto. Em termos de reutilização são utilizadas três estruturas de *frameworks*: componentes do modelo, modelos genéricos e operadores de modelagem.

Esta nova versão prevê algumas mudanças na definição dos modelos. O modelo organizacional descreve a organização em termos de funções, estrutura, processos, relações de poder e recursos. O modelo de tarefas define o escopo do projeto através da especificação de tarefas e funções. O modelo de agentes define os agentes que executam as tarefas definidas no modelo de tarefas enquanto que o modelo de comunicação descreve a interação entre esses agentes, podendo ser definidas tarefas adicionais. O modelo de especialidade é definido somente em termo das três primeiras camadas do conhecimento e utilizando esta nova abordagem da biblioteca para reutilização de modelos.

III.4.1. Análise dos Métodos de Desenvolvimento

Ao analisar e aplicar o método proposto por Keller (1987) evidencia-se sua tendência voltada para estruturas do conhecimento que utilizam regras, embora seja prevista, na fase de análise do novo sistema, a escolha da representação do conhecimento mais adequada. Pode-se considerar, também, que a utilização do método Keller no início do desenvolvimento é bastante útil. Sua representação permite uma implementação rápida com o uso de *shells* o que torna possível uma visão antecipada do comportamento do sistema. Entretanto, o uso explícito de regras pode impor uma limitação no emprego do método.

A metodologia KADS é mais abrangente e o Método de Especialidade direciona melhor a fase de análise através de uma teoria consistente. Entretanto, esta metodologia não foi totalmente desenvolvida, pois não existe uma definição detalhada de todos os modelos. O Modelo de Especialidade é o principal modelo, sendo definido em maiores

detalhes, encontrando-se vários exemplos (Schreiber, 1992), (Schreiber, Wielinga e Breuker, 1993). O Modelo de Cooperação foi também definido encontrando-se, porém, somente um exemplo na literatura (Schreiber, Wielinga e Breuker, 1993).

A facilidade do Método de Especialidade do KADS é a reutilização de modelos. Na nova versão do KADS (Velde, 1994), (Schreiber et alii, 1994) nota-se que foram realizados estudos definindo melhor o modelo organizacional, embora a ênfase continue no Modelo de Especialidade. No exemplo fornecido, o Modelo de Tarefas, de Agentes e de Comunicação parecem ser um único modelo com diferentes aspectos, necessitando de uma melhor definição.

O método de orientação a objetos de Sierhuis (1991) é uma proposta que traduz, facilmente, os conceitos das camadas definidas no método de análise do KADS para a orientação a objetos. Esse paradigma é utilizado para a fase de projeto, trazendo os benefícios dessa abordagem para o desenvolvimento de sistemas baseados em conhecimento. Considerando a característica incremental do desenvolvimento de sistemas baseados em conhecimento, essa proposta mostra-se bastante vantajosa, pois fornece um método que facilita a manutenção e a reutilização, dois fatores preponderantes para o desenvolvimento desses sistemas.

MIKE e TDBM mostram uma das tendências da área de desenvolvimento de sistemas baseados em conhecimento, que é a utilização de métodos formais para a especificação.

Considerando as propostas de MIKE e VITAL conclue-se que o modelo de especialidade tem sido bastante utilizado e adaptado por outros métodos.

III.5. Avaliação da Qualidade

A tarefa de controlar a qualidade de software não é algo simples pois está relacionada com a gradativa incorporação da qualidade no software no decorrer de seu processo de desenvolvimento (Arthur, 1985). Esse controle é complexo sendo esta uma atividade abrangente, pois a qualidade do software relaciona os diferentes produtos e as formas do processo com métricas específicas (Pressman, 1992).

A qualidade do software pode ser definida como um conjunto de propriedades a serem satisfeitas em um determinado grau, de modo a atender as necessidades de seus usuários (Rocha, 1987).

Um software de qualidade deve permitir ao usuário um alto grau de satisfação, que pode ser definido em termos de atributos externos de qualidade do software (Denning, 1992). Entretanto, isso não é suficiente para se atingir a qualidade, existindo fatores internos não claramente visíveis, tais como a modularidade, que não pode ser avaliada diretamente pelo usuário mas cuja ausência implicará numa dificuldade de manutenibilidade (Ghezzi, 1991).

Construir software de boa qualidade é um objetivo perseguido pelos que desenvolvem software. O uso de métodos ao longo do desenvolvimento pode ser visto como uma forma de buscar essa qualidade e avaliar se a meta está sendo atingida. As normas ISO 9000-3 e ISO 9126 cobrem os aspectos de qualidade a nível do processo e do produto.

Sistemas baseados em conhecimento, por sua característica de permitir respostas incorretas ou com determinado grau de incerteza e também pelo seu comportamento, muitas vezes em desacordo com as expectativas do usuário, geraram uma série de estudos sobre o controle da qualidade. Foram, então, definidos alguns procedimentos gerais com tarefas e métricas específicas para esses sistemas.

A necessidade de serem feitos estudos e experimentos específicos para verificação, validação e testes de sistemas baseados em conhecimento possibilitou o surgimento de um novo campo de pesquisa. As várias estruturas propostas de verificação e validação e os diversos sistemas desenvolvidos com esse propósito são um exemplo de pesquisas na área. Entretanto, trata-se de uma área ainda imatura, pois os termos verificação, validação, avaliação e testes têm diversas definições muitas vêzes conflitantes além de ignorarem a terminologia empregada na engenharia de software (Hoope, 1993).

No glossário de terminologia padrão da IEEE (Culbert et alii, 1987), verificação é definida como o processo de determinar se os produtos desta fase de desenvolvimento de software atingiram todos os requisitos pré-estabelecidos na fase anterior. A validação se refere ao processo de avaliar o software no final do processo de desenvolvimento para garantir o atendimento às necessidades que motivaram sua construção. Boehm simplifica essa definição em duas perguntas: Estamos construindo corretamente o produto? (verificação); Estamos construindo o produto correto? (validação).

Na literatura foram encontradas algumas propostas gerais (Green et alii, 1987), (Culbert et alii, 1987), (Naser, 1988), (Geissman e Schultz, 1989), (O'Leary, 1989), (Botten, 1989), (Arora e Cooke, 1991), (Bench-Cupon, 1993), (Ducassé e Emde, 1993) de validação e verificação dos componentes desses sistemas e também uma série de

ferramentas e técnicas (Krisnamurthy et alii, 1987), (Gupta, 1991), (Kiper, 1992), (Vinson et alii, 1992) , (Jafar e Bahill, 1993) que permitem validar e verificar os sistemas baseados em conhecimento. Entretanto, este assunto é objeto de uma tese complementar a este trabalho desenvolvida por Oliveira (1995), onde a revisão bibliográfica da avaliação da qualidade de sistemas baseados em conhecimento é descrita em detalhes.

III.6. Aspectos Gerênciais

Na gerência são desempenhadas atividades muito importantes para o sucesso do desenvolvimento de um software, pois estas controlam os recursos e as atividades técnicas do processo. O objetivo principal da gerência é garantir que o software seja entregue dentro do prazo estimado, de acordo com os custos estabelecidos e que este apresente os atributos funcionais e de qualidade requisitados pelo usuário (Fairley, 1986).

A gerência controla as atividades de construção, visando estabelecer um plano de desenvolvimento que possa ser executado. O objetivo é construir um sistema que atenda às necessidades do usuário, de acordo com as restrições e os requisitos estabelecidos. Suas atividades compreendem o planejamento do projeto, com a estimativa de custos, cronograma e identificação dos recursos necessários e acompanhamento, com o controle e gerência dos mesmos recursos, qualidade e produtividade.

Na literatura foi encontrado muito pouco material sobre a gerência dos sistemas baseados em conhecimento, existindo artigos que tratam principalmente do gerenciamento do processo de desenvolvimento através de propostas de modelos de ciclo de vida. No sistema CANASTA (Rewari, 1991) e em Guida e Tasso (1994) são relatadas experiências sobre a gerência de sistemas baseados em conhecimento, sendo identificadas uma lista de questões a considerar na gerência de sistemas baseados em conhecimento: envolver gerentes, usuários e especialistas, estabelecer e gerenciar a equipe, considerar treinamento necessário, administrar espectativas gerenciais, selecionar ferramentas apropriadas, entre outras.

Arora e Cooke (1991) ressaltam as diferenças da estimativa de custo para os sistemas baseados em conhecimento. Em Nemecek e Bemley (1993) é discutida uma adaptação do método de avaliação de custos por ponto por funções para sistemas baseados em conhecimento. Esta proposta encontra-se ainda em fase de experiência.

Em Hillmer et alii (1991) é descrita uma pesquisa para estabelecer uma estrutura quantitativa para apoiar gerentes a identificar riscos em projetos de sistemas especialistas.

Moulin (1990) apresenta uma experiência na definição de um planejamento estratégico para introdução da tecnologia de sistemas especialistas.

A área de gerência de sistemas baseados em conhecimento foi pouco explorada sendo isto razoável, pois os sistemas baseados em conhecimento sairam do ambiente exploratório para o desenvolvimento em produção, há mais ou menos uma década. A preocupação inicial é provar a utilidade e viabilidade de desenvolvimento de produtos com essa tecnologia. Depois disso o essencial é estabelecer o processo, os métodos e as ferramentas de construção e as normas e ferramentas de avaliação. Assim, uma vez estabelecido o desenvolvimento e o ambiente de construção, em termos produtivos, deverão surgir propostas e adaptações de métodos tradicionais de gerência da engenharia de software para esses sistemas.

A integração de sistemas convencionais com módulos inteligentes deve ser um ponto preponderante na análise do problema e da solução proposta, pois atualmente a maioria das organizações espera usar os sistemas baseados em conhecimento com sistemas de suporte à decisão, sistemas de informação executiva e uso de informação estratégica.

III.7. Métodos de Manutenção de SBC's

A manutenção dos sistemas baseados em conhecimento tem sido apontada na literatura como um ponto crucial no ciclo de vida, principalmente, pela natureza interativa na obtenção do conhecimento e também pela complexidade de validação e testes no comportamento inteligente desses sistemas (Coats, 1988).

Essas características acarretam a necessidade de se ter um processo interativo de desenvolvimento, gerando maior atenção na fase de manutenção.

A base de conhecimentos é um dos componentes onde ocorrem as maiores mudanças. O conhecimento, por sua natureza mutável, deve ser constantemente aperfeiçoado na base. Por isso, a modelagem do conhecimento num nível alto de abstração é importante, para se obter uma maior manutenibilidade dos produtos.

Na literatura foi encontrada uma proposta de método de manutenção da base de conhecimentos. Este organiza a base, facilitando a manutenção de sistemas representados por regras de produção. A manutenção é facilitada pela divisão das regras em grupos e na concentração dos fatos que carregam informação entre os diferentes grupos de regras. Seu objetivo é tornar a base de conhecimento mais compreensível, manutenível, eficiente e adequada para o processamento paralelo, e com isso desenvolver sistemas baseados em conhecimento mais fáceis de serem modificados (Jacob e Froscher, 1990).

Este método proposto por Jacob e Froscher (1990) pode ser aplicado desde o início do projeto ou pode ser introduzido depois que uma primeira versão do protótipo tenha sido construída, mas sempre antes do desenvolvimento do produto final. Em trabalhos de exploração de um novo domínio de problema, onde os requisitos são fracamente definidos, a aplicação do método desde o início pode acarretar em mais custos do que benefícios e a introdução mais tarde não traz nenhum efeito na precisão ou no tempo de execução do sistema.

Foram realizados testes com este método para verificar a sua viabilidade, constatando-se que a manutenção foi feita mais rapidamente e as mudanças foram de melhor qualidade. A base do método é o rastreamento do fluxo de dados entre regras e grupos através do significado dos nomes do conjunto de variáveis ou pelo uso das regras. O método suporta a automação, tendo sido desenvolvida uma ferramenta que separa as regras em grupos, gerando um gráfico de relacionamento entre elas (Jacob e Froscher, 1990).

III.8. Ferramentas

O aumento da produtividade no desenvolvimento dos sistemas baseados em conhecimento está relacionado com o surgimento de ferramentas que auxiliam o processo de construção e avaliação, principalmente os ambientes de programação (Luger, 1989).

Na pesquisa realizada por Philip e Hilbert (1990) constatou-se que a maioria dos sistemas desenvolvidos utilizam *shells* que são produtos de software que provêm uma estrutura de trabalho onde é possível se implementar um sistema baseado em conhecimento, pois contêm uma máquina de inferência e suportam alguns formalismos de representação do conhecimento (Scott et alii, 1991).

A utilização de *shells* acarreta numa série de benefícios incluindo um aumento de produtividade e facilidade no desenvolvimento e maior eficiência na manutenção. Entretanto, o seu uso possui limitações acarretando com que a maioria das organizações utilize uma interface com programas externos escritos em linguagens de programação convencionais.

Pesquisa realizada por Mihagutti (1995) constatou que no Brasil as ferramentas de programação mais utilizadas são Nexpert, Art, Aion-Ds, GURU, TIRS, KT enquanto que no mercado americano a mais utilizada é o ambiente de programação KEE.

Na literatura foram encontradas ferramentas que podem ser classificadas como ferramentas de construção que têm como objetivo dar apoio à implementação, à análise e ao projeto, e as ferramentas de avaliação da qualidade, que auxiliam na verificação e na validação desses sistemas.

Desde o desenvolvimento com sucesso do primeiro sistema especialista até hoje um grande número de ferramentas de construção foi introduzido tanto na comunidade acadêmica como na indústria. Essas ferramentas se estendem desde linguagens de programação de alto nível até editores inteligentes que completam as ferramentas *shells* (Patterson, 1990).

Analisando algumas propostas (Patterson, 1990), (Waterman, 1986), (Guida e Tasso, 1994), (Mylopoulus et alii, 1993), (Cherubin et alii, 1989) (Czejdo et alii, 1993), (Blackman, 1990), as ferramentas de construção podem ser divididas em ferramentas de implementação, ferramentas ou ambientes que auxiliam a engenharia do conhecimento, oferecendo suporte à aquisição, à representação e à implementação da base de conhecimentos e máquina de inferência e ferramentas CASE para sistemas baseados em conhecimento que integram as etapas do ciclo de vida.

Por sua vez, as ferramentas de implementação podem ser classificadas nos seguintes grupos: *shells* executadas em computadores pessoais ou de grande porte, ambientes de inteligência artificial, isto é, conjunto de ferramentas executadas em estações de trabalho bastante poderosas, linguagens de programação de inteligência artificial tais como, Lisp e Prolog e linguagens de programação convencional (C, Pascal e Fortran) (Bader et alii, 1988).

As linguagens de programação, tanto as convencionais como as de inteligência artificial, oferecem a vantagem de poderem executar em vários ambientes de hardware, enquanto que as outras ferramentas, *shells* e conjunto de ferramentas são específicas

para algumas configurações de hardware, formalismos de representação do conhecimento e mecanismos de controle.

As ferramentas próprias para os sistemas baseados em conhecimento são mais rápidas do que as linguagens de programação em termos de desenvolvimento, mas normalmente oferecem um menor desempenho na execução e uma flexibilidade menor. Ferramentas com orientação a objetos e regras tem sido introduzidas oferecendo uma abordagem flexível (Rasmus, 1994).

A escolha das ferramentas de programação é um aspecto muito importante no desenvolvimento, sendo sugerido por vários autores critérios para sua escolha (Bader et alii, 1988), (Patterson, 1990), (Gevarter, 1987), (Stylianou et alii, 1992). No Anexo II temos um exemplo de formulário para ***Identificação dos Critérios Relevantes para Definição do Ambiente de Programação***.

III.9. Conclusão

Os sistemas baseados em conhecimento têm sido descritos ao longo da literatura, passando dos sistemas simples aos complexos, dos inovativos aos comuns. No entanto, pouco se descreveu sobre como esses sistemas passaram das fases experimentais para a fase operacional. As lições aprendidas no desenvolvimento, testes e gerência de sistemas têm sido perdidas, por não serem convenientemente relatadas (Gupta, 1991).

O processo de desenvolvimento de sistemas baseados em conhecimento é algo que ainda está sendo amadurecido (Jong, 1988). O desenvolvimento de sistemas, a nível de produção, necessita de conhecimento sobre o processo de desenvolvimento, os produtos de cada etapa e os métodos para obtenção destes produtos. Na literatura sobre sistemas baseados em conhecimento podem ser encontrados vários modelos de ciclo de vida. Entretanto, somente alguns métodos de desenvolvimento estão definidos suficientemente de forma a possibilitar sua aplicação.

Este capítulo mostrou algumas propostas de modelos de ciclo de vida, métodos de desenvolvimento e alguns aspectos de controle da qualidade, gerência e manutenção desses sistemas. Percebe-se que existe carência de uma estrutura de desenvolvimento madura e principalmente de métodos de avaliação da qualidade e de gerência. Entretanto, estão surgindo várias ferramentas e nota-se uma preocupação com o processo de desenvolvimento, modelagem conceitual, verificação e validação de sistemas

baseados em conhecimento. Assinala-se, principalmente, os ambientes de programação, que facilitam a fase de implementação e depuração dos sistemas.

O histórico da engenharia de software tradicional fala, a priori, das técnicas e conceitos aplicados à programação, para depois se concentrar nas fases iniciais do desenvolvimento, isto é, na fase de análise e no projeto. Analogamente, o processo de desenvolvimento de sistemas baseados em conhecimento está ainda na fase de sedimentação das aplicações na construção de programas a nível comercial mas já com o enfoque na sistematização do processo e métodos de análise. Em muitas organizações, a ênfase está ainda na fase de implementação e validação desses sistemas.

As perspectivas futuras do desenvolvimento de sistemas baseados em conhecimento deve ser o amadurecimento do processo de desenvolvimento e dos próprios sistemas que deverão evoluir como ocorreu com os sistemas de informação cujas bases de dados servem para mais de uma aplicação. Os sistemas baseados em conhecimento terão bases de conhecimentos compartilhadas e desenvolvimento de sistemas integrados e de tecnologia híbrida. Robison (1990) considera que os sistemas gerenciadores das bases de conhecimentos têm como origem as áreas de inteligência artificial e de sistemas gerenciadores de banco de dados, interagindo também com a engenharia de software.

Por outro lado, os sistemas gerenciadores de bases de conhecimento, no futuro devem suportar vários tipos de representação e vários tipos de interface, tais como, linguagem natural, linguagem gráfica, imagens e voz. Esses sistemas devem prover também facilidades para o suporte de sistemas múltiplos e cooperativos baseados em conhecimento e ter funções análogas aos sistemas gerenciadores de banco de dados existentes hoje.

Capítulo IV

Definição de um Processo de Desenvolvimento para Sistemas Baseados em Conhecimento

IV.1. Introdução

No Capítulo II, foram caracterizados os sistemas baseados em conhecimento, suas peculiaridades, semelhanças e diferenças em relação a outros tipos de sistemas. Tais características os diferem dos demais sistemas, acarretando a necessidade de definição de um processo específico de desenvolvimento.

No Capítulo III foram apresentadas abordagens para o desenvolvimento de sistemas baseados em conhecimento, sendo identificados ciclos de vida, métodos e ferramentas para a construção, gerência e controle da qualidade desses sistemas.

O processo de desenvolvimento, para ser eficaz e conduzir à construção de produtos de boa qualidade, deve ser adequado ao domínio de aplicação e ao projeto específico que será desenvolvido. Assim sendo, o processo de desenvolvimento e avaliação da qualidade deve ser definido caso a caso, consideradas as especificidades da aplicação, da tecnologia a ser adotada na construção do produto, da organização, do grupo de desenvolvimento e dos futuros usuários do produto (Rocha, 1983), (Belchior, 1992), (Campos e Rocha, 1993), (Rocha et alii, 1993).

Os sistemas baseados em conhecimento utilizam a tecnologia baseada no conhecimento e seu desenvolvimento caracteriza-se, principalmente, pela natureza interativa e pela aquisição gradativa de conhecimento heurístico e especializado.

Neste capítulo será definido um processo de desenvolvimento para sistemas baseados em conhecimento, considerando suas especificidades, segundo os princípios estabelecidos na norma ISO 9000-3 (ISO, 1990). Esta norma, definida em 1990, significa um reconhecimento de que o processo de desenvolvimento e manutenção de software é diferente do de outros produtos industriais, sendo seu objetivo facilitar a aplicação da

norma ISO 9001 (ISO, 1987) no contexto de software. A norma ISO 9000-3 possui três partes:

- *Estrutura*, que descreve os aspectos organizacionais, a serem considerados na produção de software;
- *Atividades do ciclo de vida*, que define as ações necessárias para conduzir o desenvolvimento, e,
- *Atividades de apoio*, que define as atividades de suporte à produção, entrega e manutenção de software.

O processo de desenvolvimento para sistemas baseados em conhecimento que propomos, nesta tese, advém da evolução de definições anteriores de processos de desenvolvimento (Werneck, 1994a), (Werneck et alii, 1994a), (Werneck et alii, 1994b),(Rocha et alii, 1994), (Rocha e Werneck, 1993) e de experiências práticas de desenvolvimento realizadas utilizando essas propostas (Rabelo et alii, 1994a), (Rocha et alii, 1994).

Após um estudo da literatura especializada foi proposto em 1993, um primeiro esboço de um processo de desenvolvimento específico para Sistemas Baseados em Conhecimento. Este processo foi utilizado no desenvolvimento do sistema CRIANDO, concluído em março de 1994. CRIANDO auxilia o esclarecimento dos direitos das Crianças e Adolescentes segundo a legislação vingente.

Esta experiência possibilitou uma reformulação desta 1ª versão do processo, em maio de 1994, através da elaboração de uma proposta completa e detalhada de Processo de Desenvolvimento para Sistemas Baseados em Conhecimento para o projeto *"Desenvolvimento de Sistemas Especialistas para Cardiologia"* (SEC) desenvolvido na Unidade de Cardiologia e Cirurgia Cardiovascular do Hospital Universitário Prof. Edgard Santos da UFBA/Fundação Bahiana de Cardiologia (UCCV/FBC). O sistema SEC tem como objetivo o diagnóstico de eventos coronarianos agudos.

A primeira versão do sistema SEC foi concluída com sucesso, em setembro de 1994, utilizando o processo de desenvolvimento proposto. A avaliação deste processo foi realizada ao se dar concluída a 1ª versão do projeto. O resultado desta avaliação e experiência adquirida acompanhando as diversas fases de construção da versão nos levaram à reformulação final do processo.

As próximas seções deste capítulo detalham cada uma destas etapas. Finalmente, descreve-se, em detalhes, o processo de desenvolvimento para sistemas baseados em conhecimento em sua reformulação final.

IV.2. Primeira Definição do Processo de Desenvolvimento

No segundo semestre de 1993, após um estudo da literatura especializada, foi proposto o primeiro esboço de um processo de desenvolvimento adequado a Sistemas Baseados em Conhecimento (Werneck, 1994). Esta proposta contém a definição de um modelo de ciclo de vida com sua estrutura básica, atividades e documentação necessária. O modelo de ciclo de vida foi baseado nas propostas de Arora e Cooke (1991), Hull e Kay (1991), na prototipagem evolutiva e em conceitos definidos no modelo POLITE (Bader et alii, 1988) e em Scott et alii (1991).

O desenvolvimento de sistemas baseados em conhecimento com este modelo de ciclo de vida é realizado através de um processo cíclico, onde novas versões surgem como uma evolução, permitindo um ciclo com vários estágios de prototipagem: protótipo de demonstração de viabilidade, protótipo de pesquisa, protótipo de pesquisa em campo, protótipo de produção e protótipo operacional. Durante cada estágio de desenvolvimento são realizadas as seguintes fases: *análise do domínio, análise do conhecimento, projeto, implementação do protótipo, integração, avaliação e operação*. A Figura IV.1 apresenta este modelo de ciclo de vida.

O método de construção adotado na fase de análise do conhecimento foi o Modelo de Especialidade do KADS, descrito no Anexo VI. O controle da qualidade previa revisões informais ao longo de todo o processo e principalmente ao se ter concluído o produto de uma determinada fase. Em relação à gerência, foi previsto um planejamento e acompanhamento através do *Plano do Projeto* e elaboração de um *Relatório Histórico do Projeto*.

IV.3. Primeira Experiência de Uso do Processo: Desenvolvimento do Sistema CRIANDO

Esta primeira proposta serviu de base para o desenvolvimento de um sistema baseado no conhecimento jurídico, abordando o "Estatuto da Criança e do Adolescente",

o sistema CRIANDO (Werneck, 1994b). O objetivo principal do sistema CRIANDO é auxiliar os processos vinculados ao Direito das Crianças e dos Adolescentes, de forma a ser utilizado como um jogo e/ou tutor, simulando o decorrer de um processo na Justiça da Infância e da Juventude e também no esclarecimento dos Direitos das Crianças e dos Adolescentes.

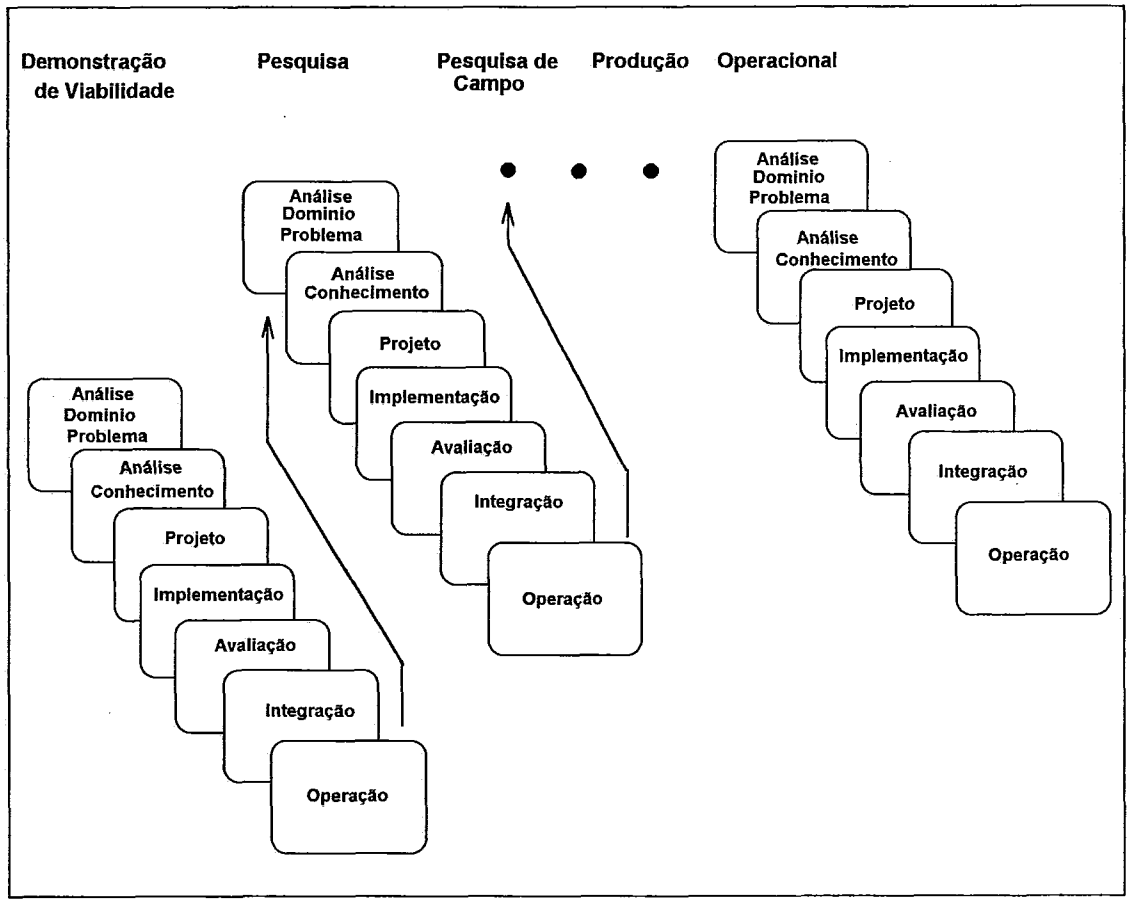


Figura IV.1 - Modelo de Ciclo de Vida da Primeira Definição do Processo de Desenvolvimento

No primeiro estágio de desenvolvimento do sistema CRIANDO foram realizadas as seguintes fases: análise do domínio, análise do conhecimento, projeto, implementação do protótipo e avaliação. Em Werneck (1994b) encontram-se especificados os produtos desenvolvidos, no Estágio de Demonstração da Viabilidade: Especificação do Domínio do Problema e Plano Geral do Projeto, Especificação de Requisitos, Especificação de Projeto, Relatório Histórico do Projeto, Relatório de Avaliação e Testes e Listagem do Programa.

O sistema CRIANDO é um sistema *backward* baseado em regras, desenvolvido em Prolog para ser executado em microcomputadores da linha PC. O desenvolvimento deste sistema teve o apoio de um advogado especialista na área de família, consultado nas fases de análise do domínio e avaliação do sistema. O papel desempenhado pelo consultor especialista foi fundamental para que o sistema CRIANDO pudesse ser construído com êxito em dois meses com um único desenvolvedor. O consultor validou o documento "Especificação do Domínio do Problema e Plano Geral do Projeto" e sugeriu um livro para aquisição do conhecimento nas fases posteriores.

O método de especialidade do KADS, utilizado na modelagem conceitual do CRIANDO, atendeu às expectativas, garantindo uma aquisição e compreensão dirigida do conhecimento. Entretanto, houve uma dificuldade inicial na utilização do método, pois este não está totalmente definido, sendo necessárias adaptações. No momento da modelagem do CRIANDO, por exemplo, não estava disponível a linguagem de definição da camada de domínio. Foi, também, identificada a necessidade de expressar o modelo gerado de uma forma mais próxima ao domínio do problema.

O uso de uma *shell* simplificou bastante a descrição dos predicados. A construção do protótipo do CRIANDO foi bastante simples, principalmente devido ao uso de um grafo hierárquico de regras. A programação deste protótipo foi gradativa, com a realização de testes ao longo da construção de cada caminho definido no grafo hierárquico de regras.

Analisando o sistema na fase de avaliação, o especialista considerou inadequada a terminologia jurídica em alguns pontos. Assim sendo, verificou-se ser recomendável que o consultor especialista avalie, sempre, todo o conhecimento definido ao longo do processo, inclusive na fase de projeto, para garantir uma implementação coerente com os termos usados na área do domínio do problema.

IV.4. Reformulação do Processo: Experiência de Desenvolvimento do Sistema SEC

Em março de 1994, a equipe de Engenharia de Software da COPPE/UFRJ foi convidada a participar do projeto "*Desenvolvimento de Sistemas Especialistas para Cardiologia*" (SEC) da UCCV/FBC (Projeto FBC-FINEP, convênio nº 66940058-00). O sistema SEC tem como objetivo apoiar médicos, não especialistas em cardiologia, no

diagnóstico de eventos coronarianos agudos. O sistema justifica-se por ser esta a enfermidade cardíaca de maior incidência e pelo fato de, no momento da crise, o paciente normalmente procurar um hospital onde, nem sempre, está disponível um cardiologista.

A equipe da COPPE apoia o desenvolvimento do sistema SEC no que se refere a aspectos de Engenharia de Software para Sistemas Baseados em Conhecimento: definição/acompanhamento do processo de desenvolvimento e garantia da qualidade a nível do processo e do produto.

Em abril de 1994, foi iniciado o trabalho com uma primeira visita da equipe da COPPE à UCCV/FBC com o objetivo de conhecer o ambiente e o projeto a ser desenvolvido utilizando a tecnologia baseada no conhecimento. Nesta visita foi definido um primeiro esboço do processo de desenvolvimento, avaliação da qualidade e um plano de ação para o projeto SEC. Neste momento foi decidida a elaboração detalhada do processo de desenvolvimento e avaliação da qualidade do SEC que seria apresentado na FBC um mês após, iniciando-se, então, o projeto com uma semana de treinamento da equipe.

No início de junho, foi realizada uma nova visita à UCCV/FBC de duas semanas para apresentar o processo de desenvolvimento e treinar a equipe (engenheiros do conhecimento e especialistas) no uso deste processo. Além do treinamento no processo, os engenheiros do conhecimento foram treinados no método KADS e nas técnicas de aquisição do conhecimento a serem adotadas. Os especialistas (cardiologistas), por sua vez, receberam uma palestra sobre sistemas baseados em conhecimento, enfatizando os sistemas especialistas, suas características e técnicas empregadas para sua construção.

Na segunda semana foi iniciado o desenvolvimento do SEC com a elicitação e documentação do conhecimento. A partir desse momento, o acompanhamento e orientação na realização das atividades foi feito, em geral, por contatos telefônicos, correio e correio eletrônico através do envio do material e das avaliações realizadas ao longo do processo.

A definição e acompanhamento do processo de avaliação da qualidade do SEC foram realizados em trabalho complementar a este (Oliveira et alii, 1994a), (Oliveira et alii, 1994b), (Oliveira, 1995). Neste caso tratava-se de um trabalho a nível de mestrado, sob nossa orientação. A mestranda permaneceu em Salvador, participando ativamente de todo o desenvolvimento da 1ª versão do sistema (junho a setembro 1994). Assim, foi

possível se ter uma orientação e um acompanhamento de perto das atividades, analisando as dificuldades e facilidades encontradas. A coordenadora da equipe da COPPE também realizou várias visitas à UCCV/FBC, neste período, para acompanhamento do trabalho.

O processo de desenvolvimento e avaliação da qualidade adotado no SEC, está definido com detalhes no documento "*Processo de Desenvolvimento do SEC*" (Werneck et alii, 1994a). Este documento define o modelo de ciclo de vida adotado no projeto, os métodos e ferramentas para construção do produto, a documentação a ser produzida, os procedimentos e métodos para controle da qualidade e gerência do processo de desenvolvimento.

O primeiro passo na definição do processo de desenvolvimento do SEC foi a definição do modelo de ciclo de vida a ser adotado no projeto. Este ciclo de vida é uma evolução do modelo proposto na primeira definição do processo, a partir da experiência adquirida no desenvolvimento do sistema CRIANDO.

O ciclo de vida adotado para o SEC, como o anterior, é composto de cinco estágios de desenvolvimento, de forma que em cada estágio, é construída uma versão do sistema. O modelo evolutivo é uma alternativa de desenvolvimento que expande o sistema gradativamente, permitindo que se obtenham modelos do comportamento do software antecipadamente. Além disso, o desenvolvimento gradativo leva a uma maior segurança no desenvolvimento, evitando riscos desnecessários. O objetivo desses estágios foram determinados, sendo caracterizadas as versões por eles construídas:

1º estágio: ***Demonstração da Viabilidade***

Neste estágio é desenvolvida uma versão do sistema que manipula uma parte do problema referindo, apenas, a identificação de evento coronariano agudo com análise eletrocardiográfica simples. O tempo de desenvolvimento foi estimado em três meses.

2º estágio: ***Pesquisa***

Neste estágio é desenvolvida uma versão do sistema, de médio porte, que determine qual o evento coronariano agudo do paciente, com análise eletrocardiográfica mais detalhada e com tempo de desenvolvimento previsto de seis meses.

3º estágio: ***Pesquisa em Campo***

Neste estágio é desenvolvida uma versão do sistema de médio a grande porte, com o objetivo de ser amplamente avaliada por vários usuários, com

análise eletrocardiográfica completa. Este estágio tem duração prevista de seis meses.

4º estágio: *Produção*

Neste estágio é desenvolvida uma versão do sistema, de grande porte, determinando a gravidade do evento agudo. O produto deste estágio é uma versão do SEC operando de forma controlada pela UCCV/FBC com o objetivo de avaliar o sistema antes de torná-lo operacional em diversos locais. Este estágio tem duração prevista de seis meses.

5º estágio: *Operacional*

Neste estágio é desenvolvida uma versão do sistema, de grande porte, robusta, com um alto grau de confiabilidade e de fácil e eficiente utilização pelos seus usuários (não especialistas em cardiologia) em diferentes locais. A construção desta versão está prevista para ter duração de seis meses.

O ciclo de vida definido para o SEC, divide os estágios de desenvolvimento em fases. Aqui, também, foram feitas alterações com base na experiência de desenvolvimento do CRIANDO. Os seguintes aspectos foram identificados:

- A fase de *Análise de Domínio do Problema* só deve ser realizada no início do projeto com o objetivo de definir o problema e o escopo a ser tratado. Esta fase se assemelha à definição da *Proposta de Desenvolvimento* de um sistema convencional (Rocha, 1987a);
- A fase de *Integração* é realizada como uma estratégia de implementação e antes da fase de *Avaliação*, devendo estar contida na fase de *Construção* de cada versão do sistema;
- A fase de *Operação* não é realizada nos primeiros estágios do desenvolvimento;
- O *Planejamento do Projeto* é uma fase fundamental no desenvolvimento de sistemas, devendo ser destacada. Na primeira proposta de ciclo de vida o planejamento era uma atividade realizada na fase de *Análise do Domínio do Problema*. Nesta reformulação esta atividade é transformada em uma fase com o objetivo de planejar o projeto, definindo o escopo de cada estágio e o planejamento detalhado do estágio corrente;
- No final de cada estágio, uma avaliação deve ser realizada, que será a base para a realização de melhoramentos no processo de desenvolvimento e elaboração do

plano do próximo estágio. Por isso, nesta reformulação, foi criada a fase de *Avaliação do Processo*.

Assim, o ciclo de vida definido para o SEC inicia-se no primeiro estágio (*Demonstração da Viabilidade*) com a realização da fase de *Análise do Domínio do Problema*. Para a construção de cada versão do SEC, o modelo de ciclo de vida é organizado em até sete fases (Figura IV.2):

- Planejamento do Projeto;
- Análise do Conhecimento;
- Projeto da Versão;
- Construção da Versão;
- Avaliação do Produto;
- Operação da Versão;
- Avaliação do Processo.

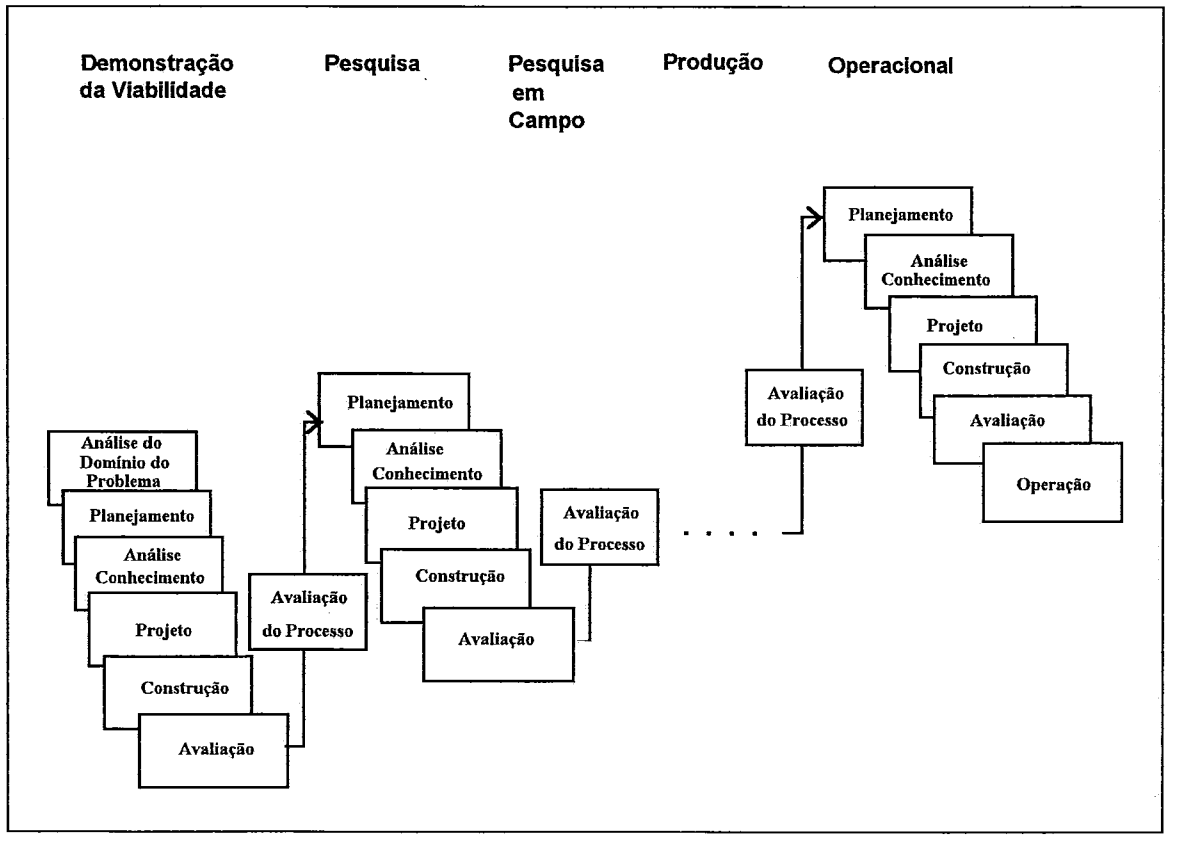


Figura IV.2 - Modelo de ciclo de vida do SEC

A fase *Operação da Versão* só é realizada nos estágios mais avançados do

desenvolvimento (*Pesquisa em Campo, Produção e Operacional*).

A partir das fases do modelo de ciclo de vida foram definidos os procedimentos e as atividades realizadas em cada uma dessas fases, os participantes em cada atividade e a documentação a ser produzida. Estas fases, bem como, os métodos e técnicas adotados no desenvolvimento, controle da qualidade e a organização da equipe do projeto serão descritos em detalhes a seguir.

Para facilitar a gerência e acompanhamento do projeto, foi definido um documento, o Relatório Histórico do Projeto. Este documento contém o registro de todas as atividades realizadas no âmbito do projeto, bem como os responsáveis e participantes na realização destas atividades. A Figura IV.3 mostra um trecho do roteiro para produção deste documento.

IV.4.1 Análise do Domínio do Problema

A ***análise do domínio do problema*** é realizada com o objetivo de definir uma visão geral do problema e o escopo do projeto. Esta fase tem como produto final a ***Especificação do Domínio do Problema***.

Nesta fase são realizadas as seguintes atividades:

- pesquisar sistemas baseados em conhecimento no domínio do problema,
- adquirir o conhecimento geral,
- descrever o sistema proposto, e,
- avaliar a Especificação do Domínio do Problema.

A atividade ***pesquisar sistemas baseados em conhecimento*** implica no estudo de sistemas relatados na literatura técnica com o propósito de obter uma visão geral do estado da arte desses sistemas no domínio do problema do projeto em questão. É responsabilidade dos engenheiros do conhecimento.

A ***aquisição do conhecimento geral*** implica na realização das seguintes sub-atividades:

- elicitar conhecimento geral,
- documentar o conhecimento na *Especificação do Domínio do Problema*, e,
- validar o conhecimento geral através de uma reunião de inspeção.

1. IDENTIFICAÇÃO DO ESTÁGIO

ESTÁGIO: DEMONSTRAÇÃO DA VIABILIDADE

2. ANÁLISE DO DOMÍNIO DO PROBLEMA

2.1 PESQUISA DE SISTEMAS BASEADOS EM CONHECIMENTO

2.1.1 Data de início:_____

2.1.2 Data de término:_____

2.1.3 Participantes:

Nome: _____

2.2 AQUISIÇÃO DO CONHECIMENTO GERAL

2.2.1 ELICITAÇÃO DO CONHECIMENTO GERAL

2.2.1.1 Data de início:_____

2.2.1.2 Participantes

Especialistas:

Nome: _____ Especialidade : _____

Equipe Técnica de Desenvolvimento:

Nome: _____

2.2.1.3 Responsabilidades

Nome: _____

2.2.1.4. Data de término:_____

2.2.2 DOCUMENTAR O CONHECIMENTO GERAL

2.2.2.1 Data de início:_____

2.2.2.2 Data de término:_____

2.2.2.3 Responsabilidades

Nome: _____

2.2.3 VALIDAÇÃO DO CONHECIMENTO GERAL

2.2.3.1 Responsável pela avaliação

Nome:_____

Assinatura:_____

2.2.3.2 Inspeções individuais

Avaliador:_____

Data início: _____

Data término: _____

Avaliador:_____

Data início: _____

Data término: _____

2.2.3.3 Reunião de Inspeção

Data: _____

Resultado

( ) Aprovado

( ) Aprovado com Alterações

Data da aprovação final: _____

Responsável (Assinatura): _____

( ) Rejeitado

2.2.3.3 Dados de Reinspeção

Data: _____

Resultado

( ) Aprovado

( ) Aprovado com Alterações

Data da aprovação final: _____

Responsável (Assinatura): _____

2.3 DOCUMENTAR O SISTEMA PROPOSTO

2.3.1 Data de início:_____

2.3.2 Data de término:_____

2.3.3 Responsabilidades

Nome: _____

2.4 AVALIAÇÃO DA ESPECIFICAÇÃO DO DOMÍNIO DO PROBLEMA

•

•

•

Figura IV.3 - Roteiro para Relatório Histórico do Projeto

A ***elicitação do conhecimento geral*** é realizada através de técnicas de elicitação do conhecimento e da técnica DELPHI descritas na seção IV.4.9. O conhecimento adquirido é documentado através da identificação do domínio do problema com a descrição geral do conhecimento, a definição do domínio específico do projeto, a descrição das tarefas e uma taxonomia geral do domínio do problema. A elicitação do conhecimento é uma atividade de responsabilidade dos engenheiros do conhecimento com a participação efetiva e essencial de especialistas no domínio do problema.

A ***descrição do sistema proposto*** implica na identificação dos objetivos, perspectivas, usuários, especialistas e restrições do sistema proposto. O estilo do sistema (assistente, crítico, segunda opinião, consultor, tutor ou automático) e a categoria das tarefas (análise: classificação, diagnóstico, avaliação, monitoramento ou predição; modificação: reparo, conserto ou controle; ou síntese: projeto, configuração, planejamento ou modelagem) são informações importantes para se definir o tipo de sistema baseado em conhecimento que será construído e o seu grau de especialidade. Esta atividade deve ser realizada pelos engenheiros do conhecimento com o apoio da gerência do projeto e da organização.

A ***avaliação da Especificação do Domínio do Problema*** implica na realização de uma reunião de inspeção[1] onde é avaliado o documento final. Devem estar presentes nesta reunião, os participantes nas atividades desta fase e a equipe gerencial do projeto.

IV.4.2 Planejamento do Projeto

Na fase de ***planejamento do projeto*** são realizados e aprovados o planejamento do projeto e o planejamento detalhado do estágio corrente, produzindo-se, assim, o ***Plano do Projeto***. As atividades realizadas nessa fase são:

- planejamento do desenvolvimento do projeto,
- planejamento do estágio, e,
- avaliação do Plano do Projeto.

Na atividade ***planejar o desenvolvimento do projeto*** é realizado o planejamento com a identificação dos produtos gerados em cada versão do sistema. Após a primeira versão, esta atividade revisará o planejamento, adaptando-o quando for necessário.

[1] Os procedimentos para realização das reuniões de inspeção, bem como os critérios de avaliação estão descritos em (Werneck et alii, 1994a) e (Oliveira, 1995).

No ***planejamento do estágio*** é produzido o Plano Detalhado do Estágio, ou seja, o plano de desenvolvimento da próxima versão, o que é fundamental para que a nova versão do sistema atinja o seu propósito no projeto. A responsabilidade pela realização das atividades de planejamento é da gerência do projeto.

A ***avaliação do Plano do Projeto*** implica na realização de uma reunião de inspeção, onde será avaliado o documento final, *Plano do Projeto.* A responsabilidade pela realização desta atividade é do coordenador do projeto, devendo participar a equipe gerencial do projeto.

IV.4.3 Análise do Conhecimento

Nesta fase são determinados os aspectos-chave do sistema, seus requisitos e o conhecimento contido no produto em desenvolvimento. A ***análise do conhecimento*** começa com a aquisição e elicitação do conhecimento desta versão, gerando o projeto conceitual do sistema, definido no documento ***Especificação de Requisitos***. As atividades realizadas nesta fase são:

- análise do conhecimento específico,
- especificação do conhecimento, e,
- avaliação da Especificação de Requisitos.

A ***análise do conhecimento específico*** implica na realização das seguintes sub-atividades:

- elicitar conhecimento específico,
- documentar o conhecimento na Especificação de Requisitos, e,
- validar os requisitos, através da realização de uma reunião de inspeção .

A ***elicitação do conhecimento específico*** é realizada através de técnicas de elicitação do conhecimento e da técnica de DELPHI descritas na seção IV.4.9. O conhecimento adquirido é documentado através da descrição geral da versão a ser desenvolvida e dos requisitos do sistema. A descrição geral da versão em desenvolvimento deve conter os objetivos do produto, suas perspectivas, características dos usuários e especialistas/consultores, restrições gerais, suposições e dependências que afetam o produto. Esta atividade deve ter a participação efetiva de especialistas, podendo, também, ser apoiada por bibliografia indicada pelos mesmos.

A *Especificação de Requisitos* deve definir os requisitos de dados de entrada/saída,

requisitos de qualidade, requisitos de interface com o usuário e outros sistemas e requisitos de conhecimento.

A ***especificação do conhecimento*** implica na realização das seguintes sub-atividades:

- modelagem do conhecimento usando o método KADS,
- documentação da modelagem na Especificação de Requisitos, e,
- validação da modelagem, através de inspeções individuais e reuniões de inspeção.

A ***modelagem do conhecimento*** é responsabilidade dos engenheiros do conhecimento, devendo ser validada pelos especialistas que participaram da elicitação do conhecimento específico. Esta atividade, algumas vezes, necessita que sejam realizadas novas reuniões para aquisição do conhecimento, principalmente porque ao modelar o conhecimento surgem vários detalhes e algumas dúvidas de conteúdo.

Em geral as atividades de elicitação do conhecimento específico e de modelagem do conhecimento são realizadas de forma interativa e não sequencial. Entretanto, a ***documentação do conhecimento*** e a ***validação dos requisitos*** deve ser finalizada antes do término da modelagem do conhecimento. A documentação do conhecimento e da modelagem é responsabilidade dos engenheiros do conhecimento e devem ser realizadas junto com as atividades de elicitação e modelagem.

A ***avaliação da Especificação de Requisitos*** implica na realização de uma reunião de inspeção, onde será avaliado o documento final.

IV.4.4 Projeto da Versão

Nesta fase os requisitos do sistema e o modelo conceitual do sistema são transformados na ***Especificação de Projeto***.

A ***fase de projeto*** é fundamental, pois é nela que são efetivamente incorporados os requisitos definidos para o sistema, devendo ser fiel à modelagem realizada na fase anterior. Contém as seguintes atividades:

- definição da representação e inferência do conhecimento,
- projeto da base de conhecimento,
- projeto do mecanismo de inferência,
- projeto do mecanismo de explicação,

- projeto da interface com o usuário,
- projeto dos módulos do sistema, e,
- avaliação da *Especificação de Projeto.*

Definir a representação e inferência do conhecimento implica na escolha dos esquemas de representação do conhecimento e do mecanismo de inferência a serem utilizados na versão em desenvolvimento do sistema, considerando o conhecimento modelado anteriormente na *Especificação de Requisitos.* A escolha da ferramenta de programação a ser utilizada no projeto deve ser feita após esta atividade.

O ***projeto da base de conhecimento*** implica na realização das seguintes sub-atividades:

- representar o conhecimento,
- especificar os fatos a serem tratados na versão em desenvolvimento,
- especificar modelo físico do conhecimento, e,
- avaliar a base de conhecimento, através de inspeções individuais.

A ***representação do conhecimento*** é realizada considerando as informações contidas na *Especificação de Requisitos* e o paradigma de representação do conhecimento escolhido na atividade anterior. Assim os fatos relacionados a esse conhecimento podem ser especificados e definidos no modelo físico do conhecimento através do *Diagrama Estrutural do Conhecimento*, que foi definido para o sistema SEC, baseado na linguagem de programação adotada.

O ***projeto do mecanismo de inferência*** define as inferências do conhecimento de acordo com o mecanismo especificado anteriormente e da ferramenta de programação a ser utilizada na versão em desenvolvimento.

O ***projeto do mecanismo de explicação*** define o módulo de explicação que o sistema baseado em conhecimento fornecerá a seus usuários.

Na atividade ***projeto da interface com o usuário*** são identificadas a hierarquia dos comandos e a interação do sistema com o usuário, definindo como deverão ser realizadas as consultas, a entrada das informações e as respostas fornecidas pelo sistema.

O ***projeto dos módulos*** especifica, numa representação próxima à linguagem de programação adotada, os módulos que serão construídos na versão em desenvolvimento.

A ***avaliação da Especificação de Projeto*** implica na realização de uma reunião de inspeção, onde será avaliado o documento final.

IV.4.5 Construção da Versão

Nesta fase o sistema é construído, a partir das especificações de projeto, sendo realizadas as seguintes atividades:

- codificar os módulos,
- avaliar os módulos,
- integrar os módulos, e,
- testar sua integração.

A ***codificação dos módulos*** implica em sua documentação segundo um padrão, previamente, definido para a linguagem de programação adotada, considerando as informações contidas na *Especificação de Projeto*. É importante, para esta atividade, o estabelecimento prévio de normas de estilo de programação.

A ***avaliação de cada módulo*** pode ser realizada através de uma inspeção individual ou de reuniões de inspeção. Os módulos complexos ou críticos para o sistema devem ser avaliados através de reuniões. Deve ser feito, sempre, um teste individual dos módulos.

A ***integração dos módulos*** identifica os módulos desenvolvidos em separado de acordo com a estrutura definida na fase de projeto. Após a integração, os testes devem ser realizados de acordo com o Plano de Testes.

A atividade ***testar versão*** consiste no teste de cada nova versão como um todo, com os especialistas que participaram no desenvolvimento e com outros cardiologistas.

IV.4.6 Avaliação do Produto

Nesta fase, a versão construída do sistema é submetida a ***avaliação para aceitação do produto***, considerando-se a base de conhecimento e o comportamento esperado para o sistema. Esta avaliação é fundamental para se obter um *feedback* (para as primeiras versões) ou a aprovação final (na versão operacional).

A fase de avaliação implica na realização das seguintes atividades:

- testar a nova versão para aceitação,

- aceitar a versão, e,
- planejar a operação.

O ***teste do sistema para aceitação*** de uma nova versão deve ser realizado na presença do coordenador do projeto e dos especialistas envolvidos no desenvolvimento.

A ***aceitação de uma nova versão*** é feita a partir dos resultados de uma reunião de inspeção realizada com a participação do coordenador do projeto, especialistas, representantes dos desenvolvedores e outros avaliadores segundo os requisitos de qualidade do sistema definidos na *Especificação de Requisitos* (Rabelo et alii, 1994b), (FBC, 1994a), (Oliveira et alii, 1994a).

Nos primeiros estágios de *Demonstração da Viabilidade* e *Pesquisa* não é realizada a operação do sistema, portanto não é necessário o seu planejamento. Quando pertinente, o ***planejamento da operação*** é revisto e detalhado. O resultado deste planejamento deve ser submetido à apreciação do coordenador do projeto. Ao final desta atividade deve estar concluído o ***Manual do Usuário***, elaborado pelos desenvolvedores e engenheiros do conhecimento com o apoio dos especialistas.

IV.4.7 Operação da Versão

Nesta fase a nova versão aceita do sistema é colocada em operação, devendo o sistema permitir, em tempo de execução, o registro dos problemas e das modificações sugeridas pelos usuários. Esses registros serão, posteriormente, listados em um ***Relatório de Avaliação e Uso da Versão***, para ser analisado pela equipe de desenvolvimento.

Esta fase só deve ser realizada a partir do terceiro estágio do desenvolvimento e implica na realização das seguintes atividades:

- treinar o usuário,
- implantar a versão, e,
- usar a versão.

O ***treinamento*** preparará o usuário para operar com o sistema, de acordo com o Plano de Treinamento que deve estar contido no *Plano do Projeto*.

A ***implantação da nova versão*** do sistema consiste em tornar disponível esta versão nos locais, onde o sistema já está ou ficará operacional.

Através do ***uso do sistema***, o usuário pode criticar os seus resultados e a sua operação. O registro dos problemas e observações relativos a essa nova versão do sistema deverá ser feito em tempo de execução sem afetar o andamento do sistema, uma vez que permite uma maior fidelidade nos registro dos problemas. Periodicamente, os desenvolvedores devem ter acesso a esses registros, para análise das necessidades relativas a futuras versões do sistema.

IV.4.8 Avaliação do Processo

Esta fase consiste numa avaliação, quanto ao processo de desenvolvimento, do estágio de desenvolvimento realizado.

Inicialmente deve ser feita uma avaliação individual por parte do coordenador do projeto, especialistas, engenheiros do conhecimento e engenheiros de software, buscando identificar aspectos positivos e negativos do processo de desenvolvimento utilizado. Finalmente deve ser realizada uma reunião de avaliação onde se busca chegar a um consenso sobre os aspectos do processo que devem ser modificados. Esta avaliação é a base para a realização de aperfeiçoamentos no processo de desenvolvimento e para o planejamento do próximo estágio.

IV.4.9 Métodos e Técnicas de Desenvolvimento

Neste processo de desenvolvimento foi proposta uma sistemática com a utilização de técnicas específicas à elicitação do conhecimento. Esta sistemática (Figura IV.4) está baseada no processo de elicitação dos requisitos definido por Leite (1990). A efetiva aquisição do conhecimento é realizada pela coleta do conhecimento com o uso de técnicas específicas de elicitação do conhecimento, pela validação do conhecimento através de análise e identificação de diferentes pontos de vista e pela resolução dos pontos de vista através da técnica DELPHI adaptada (Turban, 1991) (Boehm, 1981).

A aquisição do conhecimento é uma tarefa chave no desenvolvimento de sistemas baseados em conhecimento e é realizada ao longo de todo o processo de desenvolvimento, principalmente, nas atividades de elicitação do conhecimento geral e específico e modelagem do conhecimento. No desenvolvimento do sistema SEC, a elicitação do conhecimento foi realizada com o auxílio de um grupo de três

especialistas[1], sendo utilizadas as seguintes técnicas de elicitação do conhecimento: entrevistas, ordenação conceitual, protocolo por telefone e casos típicos.

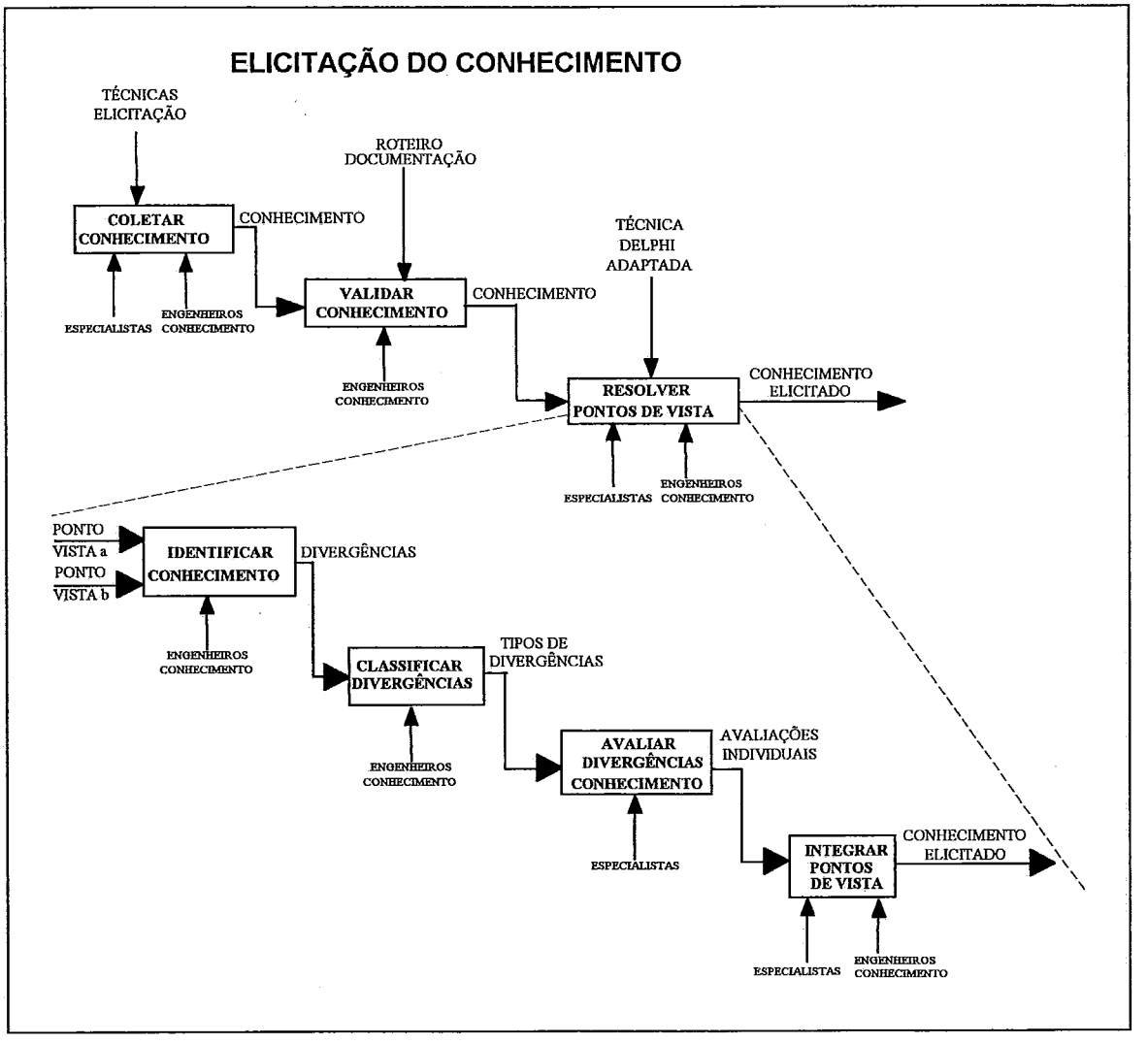


Figura IV.4 - Elicitação do Conhecimento

A aquisição do conhecimento geral, no início do projeto foi planejada utilizando *entrevistas*, que são encontros ou discussões entre o engenheiro do conhecimento e os especialistas, com o objetivo de se ter uma noção geral do domínio do problema. A primeira entrevista foi planejada para ser do estilo livre, pois num primeiro contato entre os membros da equipe é fundamental obter-se um ambiente descontraído para melhor entrosamento e comunicação. Como passo seguinte, planejou-se um tutorial dado pelos

[1] O número de especialistas na equipe é importante, pois este sempre que possível, deverá ser maior que um e em número ímpar para não haver problemas de impasses.

especialistas aos engenheiros do conhecimento para fornecer uma visão geral do conhecimento no domínio do problema, devendo ser recomendada uma bibliografia complementar para familiarização dos engenheiros do conhecimento com o assunto a ser tratado pelo sistema. Com o conhecimento básico do domínio do problema e dos termos empregados, as entrevistas devem, então, ser planejadas e estruturadas com o objetivo de revelar os objetos, a relação entre esses objetos, o processo de julgamento e a solução do problema.

A *ordenação conceitual* deve ser utilizada, principalmente, para organizar o conhecimento, pois esta técnica consiste na obtenção de um conjunto de conceitos do domínio, seguido da transferência de cada conceito para um cartão, afim de que o especialista classifique esses cartões, distribuindo-os em grupos com a descrição de suas afinidades. Esses grupos são interativamente combinados para formar uma hierarquia, se possivel. Esta técnica é útil para gerar a taxonomia geral do domínio do problema, que deverá considerar a classificação do problema sob diferentes aspectos (Assis, 1992), (Gottgtroy, 1990), (Scott, 1991).

O *protocolo por telefone* deve ser utilizado, também, na elicitação do conhecimento específico e na modelagem do conhecimento, pois esta técnica tem como objetivo obter fatos, relações e estratégias de solução. Este tipo de protocolo consiste em pedir ao especialista que solucione um problema sem vê-lo, como se estivesse falando no telefone com outra pessoa, o usuário, sendo muito útil para identificar o processo de raciocínio na solução do problema (Assis, 1992), (Gottgtroy, 1990).

Os *casos típicos* são fundamentais para verificação da modelagem e também para obter o processo de raciocínio da solução do problema. Uma análise de vários casos típicos, reais ou fictícios com o especialista é realizada, produzindo uma grande gama de informações seguras (Assis, 1992), (Gottgtroy, 1990), (Scott, 1991).

A *resolução de conflitos* é realizada através de uma adaptação da técnica DELPHI e do processo definido por Leite (1990). Para isto, são identificadas as discrepâncias entre os pontos de vista dos vários especialistas e classificados os tipos de discrepâncias, enviando-se a relação das mesmas para os especialistas que devem fazer uma análise individual das discrepâncias. A integração dos pontos de vista é realizada através de uma reunião, caso não haja consenso nas avaliações individuais.

Para modelagem conceitual do SEC foi utilizado o método KADS e para

representação gráfica da camada de domínio do problema redes semânticas, pois a linguagem definida para esta camada não estava relatada na bibliografia do KADS então disponível (Wielinga et alii, 1988), (Hickman et alii, 1989), (Schreiber, 1992).

Para a fase de projeto foi proposto um diagrama de representação da base de conhecimento, *Diagrama Estrutural do Conhecimento*, baseado na linguagem de programação adotada.

No Estágio de Demonstração da Viabilidade, foi utilizada a linguagem LIDIA (Ferreira, 1990), sendo definido o Diagrama Estrutural do Conhecimento para esta linguagem (Werneck et alii, 1994a). A escolha da ferramenta de programação definitiva a ser utilizada no projeto foi definida após a realização do 1º Estágio. No Anexo II temos um formulário para ***Identificação dos Critérios Relevantes para Definição do Ambiente de Programação***, utilizado no SEC e elaborado com base em Gevarter (1987) e Stylianou et alii (1992).

IV.4.10 Controle da Qualidade

Nesta seção abordaremos esta questão de forma suscinta, com o objetivo apenas de garantir a completude e o entendimento do processo de desenvolvimento como um todo. Aspectos relativos ao controle da qualidade do SEC foram objeto de uma tese de mestrado complementar a esta (Oliveira, 1995).

O controle da qualidade do SEC foi definido como um processo contínuo ao longo de todo o desenvolvimento. Foi proposta a realização de dois tipos de avaliação: *avaliações intermediárias* e *avaliações finais de produtos* gerados durante as várias fases do processo de desenvolvimento.

Avaliações intermediárias são realizadas ao se dar por concluída uma tarefa cujo resultado vai impactar fortemente a próxima tarefa. Esta avaliação tem como objetivo ter-se uma aprovação dos especialistas, usuários ou da equipe técnica sobre a adequação e completeza do produto da tarefa conforme pertinente a cada situação. ***Avaliações finais de produtos*** são sempre realizadas ao se dar por concluídas as tarefas de uma fase e antes de se passar à fase seguinte. O objetivo desta avaliação é ter-se a aprovação do coordenador do projeto e dos especialistas para o produto da fase. Para a realização das avaliações foi proposta a utilização da técnica de ***inspeção por fases*** (Knight e Myers, 1993).

Inspeção por fases é uma técnica disciplinada e rigorosa onde podem ser avaliados todos os produtos gerados ao longo do desenvolvimento. A avaliação do produto é feita através de uma série de avaliações parciais (fases). Cada avaliação parcial tem por objetivo verificar se o produto possui uma ou mais propriedades. As propriedades examinadas são organizadas na inspeção por fases, de forma que cada avaliação parcial pode assumir a existência das propriedades verificadas ou validadas nas avaliações parciais anteriores.

As *avaliações intermediárias* são realizadas através de inspeções com inspetores individuais ou por reuniões de inspeção. Entretanto, as *avaliações finais de produtos* incluem, obrigatoriamente, a realização de uma reunião de inspeção (Oliveira et alii, 1994a).

No contexto do processo de desenvolvimento do SEC, as características verificadas ou validadas foram os atributos identificados como imprescindíveis ou desejáveis para a qualidade do SEC. A identificação destes atributos foi feita pelos especialistas em cardiologia e pelos engenheiros de software envolvidos no projeto, a partir de um conjunto de atributos identificados como relacionados à qualidade de sistemas especialistas organizados segundo o método proposto em Rocha (1983) (Oliveira et alli, 1994a), (Oliveira et alli, 1994b).

O conjunto completo de critérios para avaliação, considerando os diferentes tipos de avaliadores (coordenador do projeto, especialistas, usuários) e os diversos produtos a serem avaliados ao longo do desenvolvimento, pode ser encontrado em Werneck et alii (1994a).

Na fase de avaliação, após a construção de cada versão, são realizados testes pelos especialistas que participam do desenvolvimento e por outros cardiologistas da FBC. Os procedimentos já utilizados no SEC incluem ***validação face a face***, onde os especialistas envolvidos no projeto compararam o desempenho do sistema com o seu próprio desempenho e ***análise de sensibilidade***, onde os especialistas sistematicamente mudavam o valor das variáveis de entrada do sistema, observando seu efeito sobre o desempenho do SEC (Rabelo et alii, 1994b), (Oliveira, 1995). O teste final para aceitação de cada versão deve ser realizado na presença do coordenador do projeto, também cardiologista, e dos especialistas.

IV.4.11 Organização da Equipe

A organização da equipe de desenvolvimento do SEC consiste de três grupos: gerência, desenvolvimento e controle da qualidade.

A gerência do projeto é responsabilidade do coordenador geral do projeto, que é o médico cardiologista, Presidente da FBC. Esta vinculação da coordenação do projeto ao nível mais alto de decisão na instituição facilitou o bom andamento do projeto, pois decisões importantes puderam ser tomadas por ele e com o seu apoio.

No SEC, a equipe de gerência tem também, a participação de um coordenador técnico, um coordenador da área médica e um coordenador administrativo. O coordenador técnico é responsável pela equipe técnica do projeto[1] e o acompanhamento de suas atividades. O coordenador da área médica é responsável pela equipe de especialistas e pode atuar como definidor dos impassses relativos ao conhecimento. Entretanto, não foi necessário em nenhum momento que este coordenador exercesse sua autoridade, sendo as decisões tomadas por consenso entre os especialistas. O coordenador administrativo é responsável pelo apoio administrativo e financeiro ao projeto.

A equipe de desenvolvimento do SEC é formada pela equipe técnica e especialistas em cardiologia da Fundação Bahiana de Cardiologia (FBC), sendo constituída pelos coordenadores técnico e da área médica, três engenheiros do conhecimento e mais dois especialistas em cardiologia. Os engenheiros do conhecimento são responsáveis pela elicitação do conhecimento, modelagem, projeto e programação do sistema. Os especialistas participam da elicitação do conhecimento e das atividades de avaliação da qualidade.

A equipe de controle de qualidade do projeto SEC é constituída pelo coordenador do projeto, pelo coordenador técnico, engenheiros de software da COPPE e avaliadores externos ao desenvolvimento do projeto, além da equipe de desenvolvimento.

IV.5 Avaliação do Processo de Desenvolvimento utilizado no SEC

O processo de desenvolvimento, descrito nas seções anteriores, foi utilizado na construção da versão 1.0 do SEC (junho a setembro de 1994), que utiliza uma

[1] Pessoal especializado em Informática (engenheiros do conhecimento e engenheiros de software).

representação em regras com tratamento de incerteza baseado nos coeficientes de incerteza do sistema MYCIN e mecanismo de controle híbrido (dirigido a dados e ao objetivo). Uma descrição da Versão 1.0 do SEC pode ser encontrada em (Rabelo et alii, 1995).

A avaliação do processo de desenvolvimento utilizado na construção desta primeira versão do SEC foi realizada através da aplicação dos formulários descritos no Anexo I. A experiência mostrou a importância e eficácia de se ter um processo de desenvolvimento bem definido, como base para o desenvolvimento e ponto de referência para coordenadores e desenvolvedores. A versão 1.0 do SEC foi concluída rigorosamente segundo o cronograma previsto e superou as expectativas. Em (Rabelo et alii, 1994a), (Rabelo et alii, 1994b) são feitas observações mais detalhadas sobre a construção da 1º versão do SEC.

A fase de avaliação da Versão 1.0 do SEC conduziu ao uso de prototipagem operacional (Davis, 1992), o que não tinha sido inicialmente previsto, mas que foi extremamente positivo para a qualidade do produto e a produtividade em geral, sendo incorporada ao processo de desenvolvimento para a avaliação de versões posteriores. A natureza interativa da aquisição e refinamento do conhecimento justificou o uso de prototipagem operacional.

Ao se planejar o estágio seguinte verificou-se que não eram necessários tantos estágios de desenvolvimento. Assim sendo, a 2ª Versão do SEC já será operacional, embora em ambiente restrito e controlado pela UCCV/FBC.

O método de especialidade do KADS, entretanto, não se mostrou totalmente adequado, pois os especialistas tiveram problemas no entendimento do modelo. Houve uma dificuldade de se entender a estrutura de inferência, principalmente quando esta era executada mais de uma vez, tendo sido sugeridas alterações que foram incorporadas ao modelo de especialidade do KADS, na proposta de extensão deste método descrita no próximo capítulo.

A equipe técnica, também, teve dificuldades com o método no que se refere à transformação do modelo de especialidade do KADS em um modelo implementável. Entretanto, o uso deste método foi fundamental para a direção e o entendimento do conhecimento do domínio do problema e do processo de raciocínio do especialista.

Na aquisição do conhecimento foi utilizada, também, a representação em grafo

proposta por Leão (1988). Esta representação implica em se agrupar evidências (sintomas, sinais e exames) que são importantes para o diagnóstico de uma doença, ordenando-as em graus de importância e atribuindo notas para cada um dos nós. Neste grafo podem ser criados nós intermediários que agrupam essas evidências relacionando-as através de operadores lógicos e notas.

Este grafo foi utilizado para extrair e representar o raciocínio do especialista quanto a associações de evidências e valorização dessas associações (Figura IV.5). A sequência de investigação das evidências pelo especialista é representada na direção horizontal, no sentido da esquerda para a direita. Um nodo representa uma associação de entradas (evidências ou outros nodos) com operadores lógicos. O valor numérico associado a cada nodo representa a probabilidade máxima ou mínima para aquele grupo de associações.

Esta representação do raciocínio em grafo foi bastante útil na fase de construção e avaliação do SEC. Ao se analisar este grafo e a estrutura de inferência do modelo de especialidade do KADS, verificou-se que ambos mostram o processo de raciocínio do especialista. Assim sendo, este grafo serviu de base para o Diagrama Heurístico do Raciocínio proposto na extensão do método KADS (Capítulo V), que possibilita uma transformação gradativa do modelo conceitual para o modelo implementável.

Os procedimentos utilizados para o controle da qualidade foram considerados um aspecto fundamental para o sucesso do 1º Estágio do SEC. Os resultados obtidos no processo de avaliação da qualidade podem ser encontrados em (Rabelo et alii, 1994a), (Rabelo et alii, 1994b) e (Oliveira, 1995).

IV.6. Reformulações Finais no Processo de Desenvolvimento

Nas seções anteriores descrevemos duas experiências de desenvolvimento de sistemas baseados em conhecimento. Estas experiências foram fundamentais para que fosse possível chegar à definição de um processo de desenvolvimento de sistemas baseados em conhecimento para Estação TABA (Rocha e Souza, 1988), (Rocha et alii, 1990), (Aguiar, 1992) e (Travassos, 1994). Nas experiências que relatamos, a ênfase do processo era o desenvolvimento como uma atividade isolada na organização. Entretanto, esta visão não é verdadeira, pois esses sistemas estão subordinados às práticas de trabalho existentes nas organizações e às habilidades e interesses dos usuários.

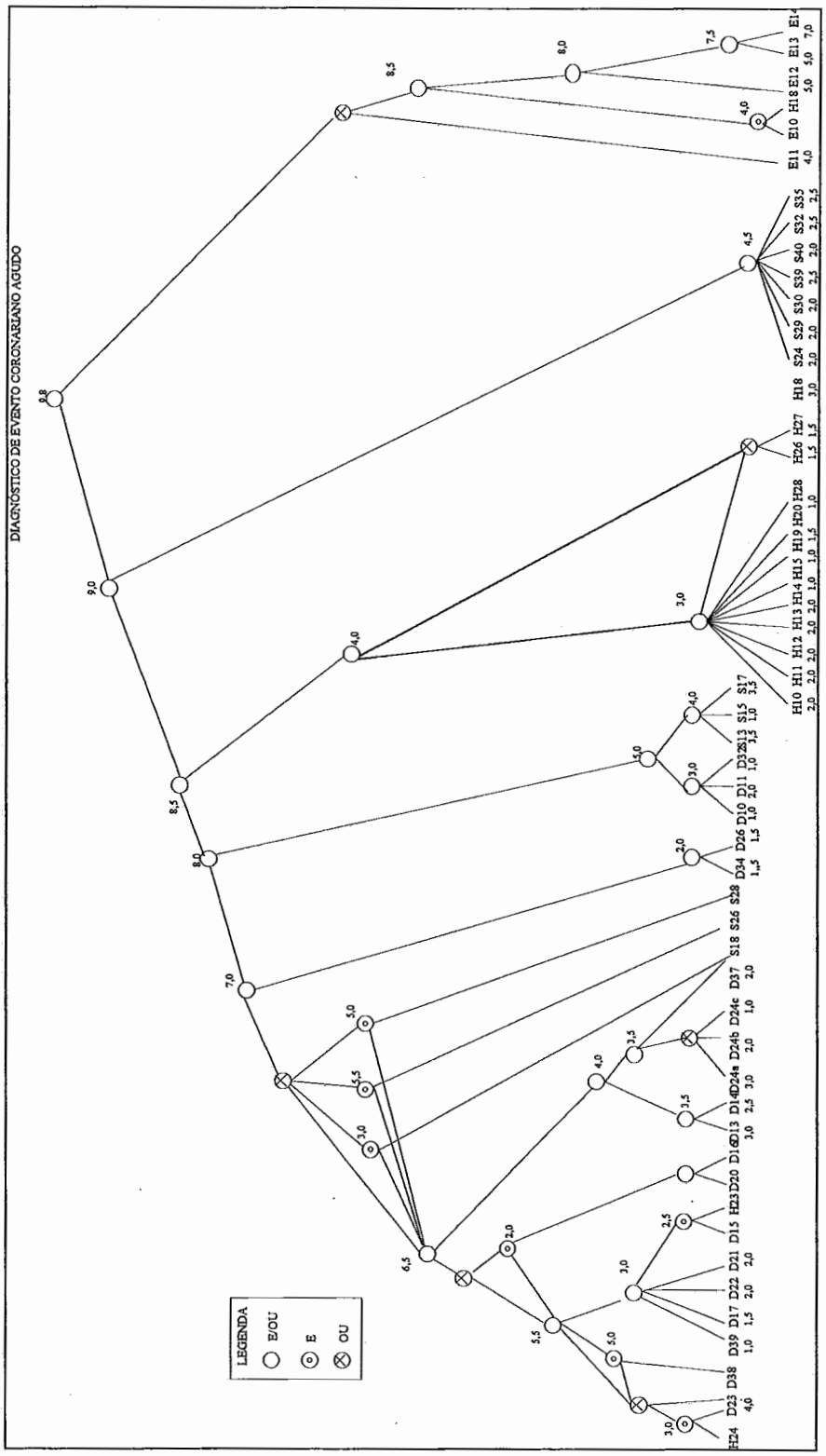


Figura IV.5 - Grafo de Representação do Raciocínio do Diagnóstico de Evento Coronariano Agudo

Os sistemas baseados em conhecimento, em geral, são desenvolvidos para interagirem com outros sistemas computacionais da organização.

No caso do SEC, embora esteja prevista sua futura integração ao SIGAH (Sistema de Informação Gerencial e Administração Hospitalar) já em operação na UCCV/FBC, ele deve operar também de forma isolada nos postos de saúde e inicialmente foi enfatizada esta utilização. Isto por várias razões, entre elas os objetivos principais do SEC e o grau de inovação que o seu desenvolvimento representava para a instituição, que com ele iniciou-se o desenvolvimento de sistemas baseados em conhecimento.

Pode-se fazer uma analogia entre o desenvolvimento de sistemas baseados em conhecimento e o de sistemas de informação, concluindo-se que a nível corporativo, ambos devem obedecer a um plano de ação estratégico. Este plano tem a finalidade de identificar novas oportunidades de negócios oferecidas pela introdução destas tecnologias (Martin, 1991), (Moulin, 1990). Assim sendo, o desenvolvimento de sistemas baseados em conhecimento deve ser realizado de forma integrada na organização, considerando novos projetos previstos e outros sistemas computacionais existentes.

Esta proposta pressupõe uma estrutura geral para o desenvolvimento integrado de sistemas computacionais, baseada em conceitos de domínio do problema e em analogia com a engenharia da informação (Martin, 1990).

O domínio do problema pode ser definido como um conjunto de classes de problemas caracterizados pela existência de uma comunidade dedicada à sua resolução (Arango e Prieto-Diaz, 1991). Essa comunidade tem acesso ao conhecimento dessas soluções e diferentes grupos poderão ter visões diferentes do que deve ser o domínio do problema. Entretanto, ao se realizar a análise de um domínio, normalmente, a comunidade estabelecerá, por consenso, a definição do escopo do problema.

A análise do domínio do problema tem sido alvo de estudos e práticas em reutilização de software e engenharia do conhecimento (Arango e Prieto-Díaz, 1991). Esta análise deverá identificar a estrutura da organização e suas metas, coletar e analisar os conhecimentos existentes neste domínio, caracterizando-o e classificando-o (Arango, 1994).

Outro conceito desta estrutura são as áreas de conhecimento que estão relacionadas a domínios de problema e deverão ser identificadas na análise do domínio. Após essa identificação, a organização fica dividida em áreas de conhecimento, devendo cada uma delas ser analisada em detalhes, com o propósito de planejar uma diretriz para a introdução das tecnologias computacionais. A identificação das áreas prioritárias é fundamental para a alocação de recursos, como também, para a elaboração de um *Plano de Desenvolvimento de Novas Aplicações Computacionais*. Desta forma, tem-se um processo integrado de desenvolvimento de sistemas computacionais.

A Figura IV.6 apresenta os componentes gerais desta estrutura de desenvolvimento, identificando os níveis de abstração desses componentes na organização. A identificação e definição desses componentes é realizada através da *Análise do Domínio do Problema*, da *Análise da Área de Conhecimento* e do *Projeto de Desenvolvimento de Sistemas*.

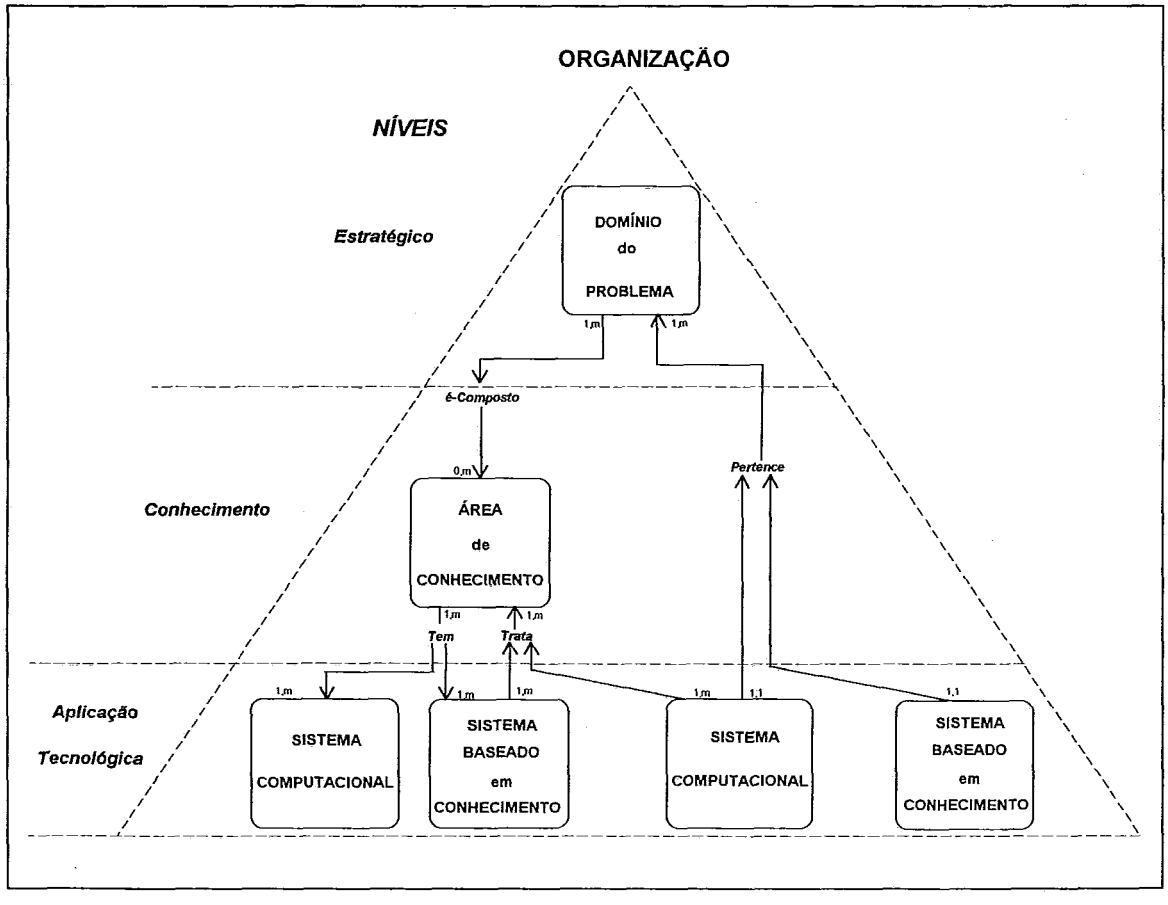


Figura IV.6: Componentes Gerais do Desenvolvimento Integrado de Sistemas na Linguagem de Definição do Domínio do KADS

A *Análise do Domínio do Problema* é realizada a nível estratégico enquanto a

Análise das Áreas de Conhecimento é feita no nível do próprio conhecimento que normalmente pode ser dividido em áreas de especialidades que estão concentradas em algumas áreas específicas da organização. O *Projeto de Desenvolvimento de Sistemas Computacionais* refere-se ao nível de aplicação tecnológica, envolvendo o desenvolvimento de novos sistemas.

A implantação desta estratégia deve ser flexível, possibilitando que cada organização escolha o momento e a forma de realização do *Plano de Ação para o Desenvolvimento Integrado de Sistemas Computacionais.* É possível, também, elaborar gradativamente este planejamento, ao se analisar em detalhes as áreas de conhecimento e os projetos de sistemas. A organização pode iniciar este planejamento enfocando uma determinada área do conhecimento que seja prioritária à instituição, ou o projeto de desenvolvimento de novos sistemas, realizando uma análise mais abrangente com o estudo das áreas de conhecimento abordadas pelo sistema.

Neste trabalho, enfoca-se o processo de desenvolvimento de sistemas baseados em conhecimento com esta filosofia. Esses sistemas tendem a fazer parte de um sistema computacional maior (Haynes-Roth e Jacobstein, 1994), (Mihaguti, 1995) e por isso é importante obter-se uma visão geral do domínio do problema e das áreas de conhecimento. A partir desta visão, é possível modularizar o sistema, definindo sub-sistemas em separado e, por conseqüência, a parte baseada no conhecimento passa a ser um projeto próprio, podendo se comunicar com os demais sistemas.

No desenvolvimento dos sistemas CRIANDO e SEC, foi realizada uma fase *de Análise de Domínio do Problema.* Nesta análise, no entanto, não foram consideradas as áreas de conhecimento tratadas pelo sistema, nem a possível interação com outros sistemas baseados em conhecimento ou não. No SEC, só foi considerada sua futura integração com o sistema SIGAH já existente. Nesta proposta reformulada, de processo de desenvolvimento de sistemas baseados em conhecimento, ampliamos o âmbito tratado por esta fase, de forma a possibilitar uma visão mais geral do projeto a ser desenvolvido no contexto da organização e o seu relacionamento com outros sistemas e áreas do conhecimento.

O modelo de ciclo de vida que agora propomos é, basicamente, o modelo de ciclo de vida utilizado para o desenvolvimento do sistema SEC no qual foram feitas pequenas alterações.

Na avaliação da Versão 1.0 do SEC ficou claro que o desenvolvimento de sistemas baseados em conhecimento deve seguir um processo evolutivo e que este é composto de *três tipos de estágios de desenvolvimento*, de forma que em cada estágio, é construída uma versão do sistema (figura IV.7). Estes estágios, caracterizam versões construídas de forma evolutiva e com objetivos bem determinados, sendo que o número de estágios depende do escopo do projeto e de seu andamento para atingir os objetivos previamente definidos:

Estágio 1: Análise da Viabilidade

Neste estágio é desenvolvida uma versão do sistema que manipula, apenas, uma parte do problema com o objetivo de verificar a viabilidade, demonstrar a utilidade do sistema e familiarizar a equipe com a potencialidade dessa tecnologia. Esta versão do sistema deve ser avaliada por diferentes especialistas na área do conhecimento.

Estágios de 2 a (n-1): Evolução

Nestes estágios são desenvolvidas novas versões do sistema num processo evolutivo, com o objetivo de acrescentar conhecimento e experimentar a potencialidade do sistema com diferentes usuários. O número de versões dependerá do projeto concreto a ser desenvolvido e deverão ser identificadas no respectivo *Plano do Projeto*. Cada versão deve ser amplamente avaliada por usuários e especialistas da área de conhecimento.

Estágio n: Maturidade (Versão Operacional)

O produto deste estágio é uma versão do sistema, contendo o nível de conhecimento julgado adequado por usuários e especialistas. Esta versão deve ser robusta, com um alto grau de confiabilidade, e de fácil e eficiente utilização por seus usuários.

As fases definidas para cada um desses estágios de desenvolvimento mantém-se inalteradas com relação ao ciclo de vida do SEC.

Neste processo, ao se iniciar um projeto (como foi feito no desenvolvimento do SEC) é prevista a realização da fase *Análise de Domínio do Problema*, no primeiro estágio de desenvolvimento (Análise da Viabilidade).

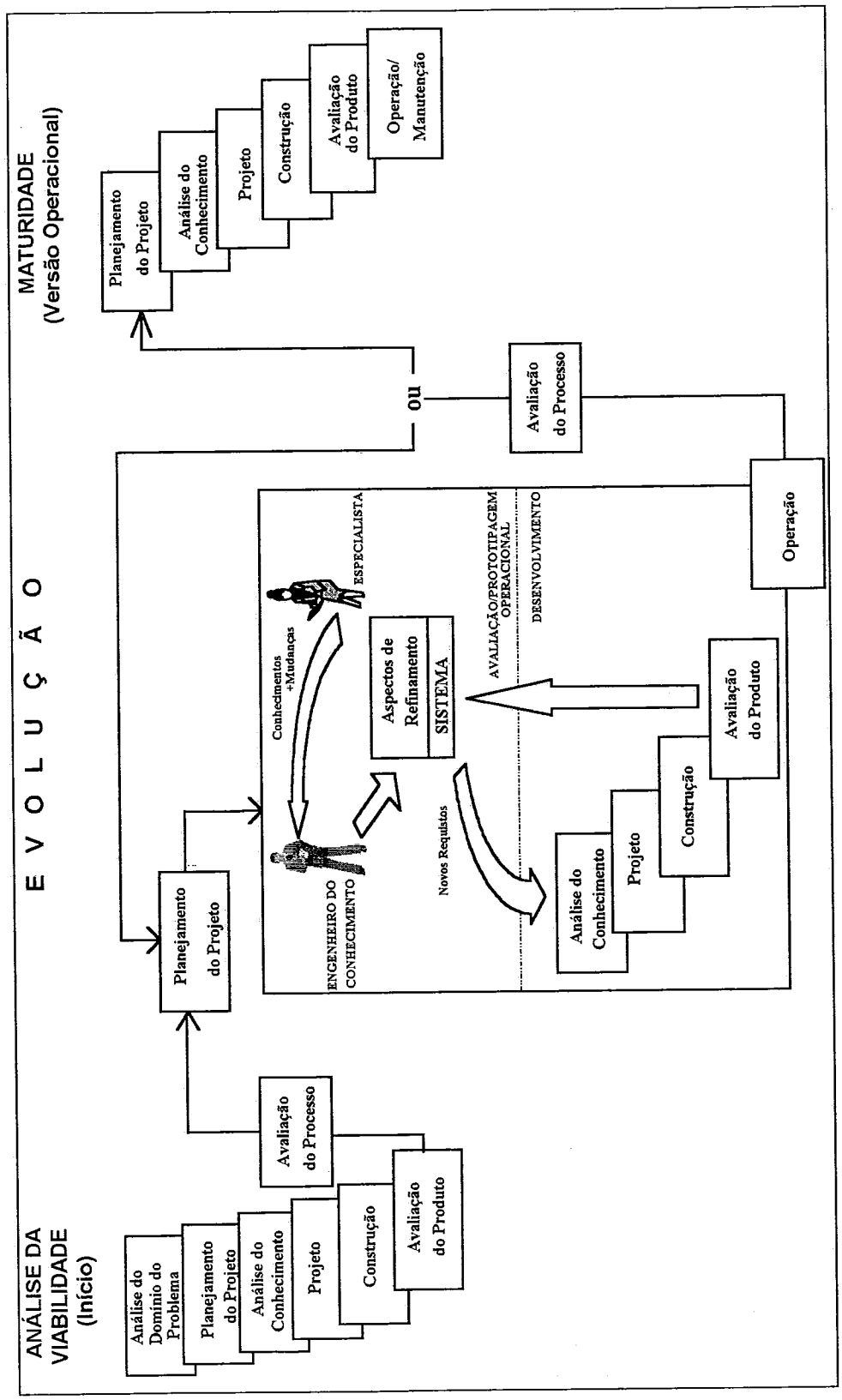


Figura IV.7 - Modelo de Ciclo de Vida para Sistemas Baseados em Conhecimento

No estágio de *Evolução* foi incorporada a prototipagem operacional (Davis, 1992), sendo que o desenvolvimento de uma nova versão pode começar com o refinamento da versão anterior até que sejam identificados os requisitos deste novo protótipo ou versão operacional. Este procedimento foi utilizado no SEC com excelentes resultados[1].

No estágio *Maturidade* (Versão Operacional) tem-se a *Operação/Manutenção* do sistema. Este trabalho enfatiza o processo de desenvolvimento e por isso não detalhará a atividade de manutenção de sistemas baseados em conhecimento.

Em Werneck et alli (1994a) encontram-se exemplos de roteiros dos documentos a serem gerados nas diversas fases deste ciclo de vida.

O método de desenvolvimento adotado neste processo de desenvolvimento para Sistemas Baseado em Conhecimento é uma extensão e adaptação da metodologia KADS (Schreiber et alii, 1993). Este método será descrito em detalhes no Capítulo V. A seguir são definidas com detalhes as alterações realizadas nas fases de Análise do Domínio do Problema, Análise do Conhecimento, Projeto e Avaliação do Produto.

IV.6.1 Análise do Domínio do Problema

O objetivo desta fase é obter a visão estratégica e global do conhecimento através de uma análise geral do domínio do problema e das áreas de conhecimento. Esta análise pode ser realizada de uma forma mais restrita do que no caso da elaboração do *Planejamento Estratégico de Sistemas*. Entretanto, deve ser mais abrangente do que a realizada nos processos utilizados no desenvolvimento dos sistemas CRIANDO e SEC. Deve-se, definir o problema dando uma visão geral do mesmo, definir as áreas de conhecimento e o escopo do projeto bem como o seu relacionamento na organização. Esta fase tem como produto final a ***Especificação do Domínio do Problema***.

Sistemas baseados em conhecimento, normalmente, estão relacionados a uma única área de conhecimento. Entretanto, sistemas complexos podem abranger mais de uma área. Por isso, a *Análise do Domínio do Problema* deve conter a identificação do escopo, áreas de conhecimento e funções do sistema.

[1] No SEC, o 2º estágio de desenvolvimento começou com o refinamento da versão 1.0 até a versão 1.3 que tem um desempenho de 80% de acerto, reduzindo pela metade o número de diagnósticos errados.

Nesta fase são realizadas as seguintes atividades[2]:

- pesquisar sistemas baseados em conhecimento no domínio do problema,
- **adquirir o conhecimento geral, identificar as áreas de conhecimento,**
- descrever o sistema proposto, e,
- avaliar a Especificação do Domínio do Problema.

Como no SEC, a *aquisição do conhecimento geral* implica na realização das seguintes sub-atividades:

- **elicitar conhecimento geral,**
- **documentar o conhecimento na Especificação do Domínio do Problema**, e,
- validar o conhecimento geral.

A ***elicitação do conhecimento*** é uma atividade que deverá ser responsabilidade dos engenheiros do conhecimento com o apoio de especialistas ou de consultores especialistas no domínio do problema, caso o projeto não tenha uma equipe de especialistas que participe do desenvolvimento. Esta atividade utiliza a sistemática e técnicas de elicitação do conhecimento propostas no SEC.

A ***documentação do conhecimento*** adquirido na elicitação do conhecimento geral utiliza o Modelo do Domínio do Problema proposto na extensão do método KADS (Capítulo V) .

As diferentes ***áreas de conhecimento relacionadas ao domínio do problema*** devem ser definidas, no caso de existir mais de uma área. Dessa forma devem ser identificados os perfis dos especialistas ou consultores que deverão participar do desenvolvimento. Esta atividade necessita de uma equipe multi-disciplinar, pois é fundamental a participação de especialistas ou consultores especialistas no domínio do problema, além dos engenheiros do conhecimento.

IV.6.2 Análise do Conhecimento

Através da ***Análise do Conhecimento*** gera-se o projeto conceitual do sistema, definido no documento ***Especificação de Requisitos.***

As atividades realizadas nesta fase são:

- **análise do conhecimento específico,**
- **especificação do conhecimento,** e,

[2] Em destaque estão as atividades/sub-atividades aumentadas ou alteradas nesta revisão do processo.

- avaliação da Especificação de Requisitos.

A ***análise do conhecimento específico*** implica na realização das seguintes sub-atividades:

- **elicitar conhecimento específico,**
- documentar o conhecimento na Especificação de Requisitos, e,
- validar os requisitos.

A atividade de ***elicitação do conhecimento específico*** deve ter a participação efetiva de especialistas ou ser realizada por outros meios de aquisição indicados pelos consultores especialistas (livros, filmes, questionários, observações, casos, ...). Na aquisição de conhecimento é importante ter-se alocado ao projeto vários especialistas no assunto e quando isto não for possível, deve-se ter pelo menos um consultor especialista à disposição para esclarecimento de dúvidas.

A *especificação do conhecimento* implica na realização das seguintes sub-atividades:

- **modelagem do conhecimento** usando a extensão do método KADS definido no Capítulo V,
- documentação da modelagem na Especificação de Requisitos, e,
- validação da modelagem.

IV.6.3 Projeto da Versão

Nesta fase os requisitos do sistema e o modelo conceitual do sistema são transformados na ***Especificação de Projeto***.

As maiores alterações com relação ao processo de desenvolvimento utilizado no SEC ocorreram nesta fase, pois na avaliação do processo realizada após a construção da versão 1.0 foi identificada uma dificuldade em se projetar o sistema, a partir do modelo de especialidade do KADS (Rabelo et alii, 1994a).

Esta fase implica na realização das seguintes atividades:

- definição da representação e inferência do conhecimento,
- **modelagem lógica,**
- **modelagem física,** e,
- avaliação da Especificação de Projeto.

A ***modelagem lógica*** do sistema consiste em se gerar uma visão interna da modelagem de especialidade do KADS, para o desenvolvedor, através da utilização do

método KADS-estendido. O objetivo desta modelagem é ser uma ponte entre o modelo de especialidade e o modelo físico. O modelo lógico dá origem ao ***Diagrama Heurístico do Raciocínio*** e ao ***Diagrama do Domínio do Problema***. Esta atividade implica na realização das sub-atividades:

- **construção do Diagrama Heurístico do Raciocínio,**
- **geração do Diagrama do Domínio do Problema,** e,
- **validação do modelo lógico** através da realização de inspeções individuais.

No Capítulo V, estão definidos, com detalhes, os passos para construção do Diagrama Heurístico do Raciocínio e do Diagrama do Domínio do Problema.

A ***modelagem física do sistema*** transforma o modelo lógico numa representação possível de ser implementada com a linguagem de programação escolhida. Esta modelagem, também, é realizada através do método KADS-estendido, sendo previstas as sub-atividades:

- **construção do Modelo de Implementação do Usuário,** e,
- **construção do Modelo de Implementação do Sistema**.

O ***Modelo de Implementação do Usuário*** consiste na elaboração dos Diagramas de Interface com o Usuário e de Explicação do Raciocínio. No ***Diagrama de Interface com o Usuário*** são identificadas a hierarquia dos comandos e a interação do sistema com o usuário, definindo como deverão ser realizadas as consultas, a entrada das informações e as respostas fornecidas pelo sistema. No ***Diagrama de Explicação do Raciocínio*** são definidas as informações de explicação e o momento que o sistema baseado em conhecimento poderá fornecer essas explicações a seus usuários. Estas atividades são desempenhadas pelos engenheiros do conhecimento com o apoio dos especialistas, consultores ou usuários.

O ***Modelo de Implementação do Sistema*** implica na realização das seguintes sub-atividades:

- **construção do Diagrama Estrutural da Base de Conhecimento,**
- **elaboração da Especificação da Base de Conhecimento,**
- **elaboração da Especificação da Memória de Trabalho,** e,
- elaboração **da Especificação dos Módulos**, conforme descrito no método KADS-estendido.

O ***Diagrama Estrutural da Base de Conhecimento*** contém a estrutura geral da base de conhecimento e das representações que serão utilizadas a nível genérico. O conteúdo

da base de conhecimento é definido nesta estrutura e seu conteúdo é descrito na ***Especificação da Base de Conhecimento***.

A ***Especificação da Memória de Trabalho*** implica na definição do conteúdo de cada caso executado no sistema que será armazenado em arquivos ou banco de dados, dependendo das facilidades da linguagem de programação, para posterior consulta ou nova execução.

A ***Especificação dos Módulos*** é realizada ao ser necessária a definição de alguma função especial, tais como tratamento de incerteza, cálculos ou métodos utilizados por uma classe de objetos em linguagens orientadas a objetos. Assim, a forma desta especificação é totalmente dependente do ambiente de programação utilizado na implementação do sistema.

A atividade ***construção do Modelo de Implementação do Sistema*** é realizada pelos desenvolvedores e engenheiros do conhecimento. Entretanto, a sub-atividade *Especificação da Base de Conhecimento* deve contar com o apoio dos especialistas ou do material utilizado na atividade *elicitação do conhecimento específico*.

IV.6.4 Avaliação do Produto

Nesta fase, a nova versão do sistema é submetida à avaliação considerando-se a base de conhecimento e o comportamento esperado para o sistema. Esta avaliação é fundamental para se obter um "feedback" e implica na realização das seguintes atividades:

- **testar versão para aceitação**,
- aceitar versão,e,
- planejar a operação.

No ***estágio de evolução***, a atividade ***testar versão para aceitação*** é realizada de forma interativa, utilizando a prototipagem operacional. Esta prototipagem prevê um mecanismo de refinamento da versão em teste. Este mecanismo baseia-se no uso de técnicas de validação qualitativa e quantitativa de sistemas baseados em conhecimento, análise dos resultados obtidos e reuniões de refinamento da versão.

Nas reuniões de refinamento da versão são analisadas as mudanças necessárias e seu impacto no sistema, sendo definida a incorporação dessas mudanças na versão ou como novos requisitos numa futura versão. Este mecanismo de avaliação está baseado na

sistemática de avaliação de protótipos definido por Mayhew et alii (1990) e descrito em detalhes em Werneck (1992).

Nossa proposta é alterar no sistema apenas aspectos de refinamento do conhecimento e mudanças superficiais ou locais. Essas mudanças deverão ser definidas conforme a classificação de alterações proposta por Mayhew e Dearnley (1989), que identifica-as como: *estética*, *local* e *global*. A vantagem dessa classificação, considerando o tipo de alteração, é a possibilidade de se ter um efetivo controle das alterações e de seu impacto.

Alterações estéticas são consideradas triviais e não têm grandes repercussões. Para sistemas baseados em conhecimento incluem alterações do tipo refinamento do conteúdo do conhecimento ou mudanças na interface com usuário.

Alterações locais são aquelas que tem impactos localizados no sistema e incluem mudanças onde se altera pouco a estrutura do conhecimento, acrescentam-se ou retiram-se campos, dividem-se ou agregam-se variáveis, reagrupam-se informações entre uma série de telas, etc.

Alterações globais são aquelas que tem impactos significativos em partes do sistema e incluem alterações como mudança ou aperfeiçoamento na representação do conhecimento, inclusão de novas funções ou conhecimentos diferentes.

As *alterações estéticas e locais* podem ser incorporadas na versão corrente. A realização de *alterações globais* devem ser atendidas, apenas, numa nova versão do sistema, sendo consideradas como novos requisitos do sistema.

As reuniões de refinamento tem o objetivo de discutir os resultados das validações do sistema, estratégias de testes, análise de novos casos de testes e refinamento do conhecimento da versão. Nesta reunião podem ser estudadas em maior profundidade as alterações sugeridas e seus impactos no sistema. A lista de alterações é revista e a incorporação imediata ou não, de cada alteração, depende dessa análise e da aprovação dos especialistas. Devem participar destas reuniões os desenvolvedores e o coordenador técnico. Nelas serão analisadas as mudanças necessárias e seu impacto no sistema, sendo definida a incorporação dessas mudanças na versão ou como novos requisitos numa futura versão.

Este mecanismo de avaliação se justifica para sistemas baseados em conhecimento devido à característica de aquisição permanente do conhecimento e pela sua própria natureza de amadurecimento constante e interativo ao longo do tempo.

IV.7. Conclusão

Neste capítulo foi definido um processo de desenvolvimento para sistemas baseados em conhecimento, sendo esta a primeira etapa para a automação da engenharia de software de sistemas baseados em conhecimento, um dos objetivos desta tese.

A definição do processo foi feita a partir de duas experiências, uma delas bastante significativa, o desenvolvimento do sistema SEC, por tratar-se de um projeto real. Esta experiência nos permitiu entender melhor o processo de desenvolvimento de sistemas baseados em conhecimento, reformular nossa proposta inicial e definir uma extensão ao método KADS. O método KADS-estendido será descrito no próximo capítulo.

O controle da qualidade de sistemas baseados em conhecimento proposto para este processo de desenvolvimento está descrito em Oliveira (1995). Embora este trabalho de controle da qualidade seja uma proposta específica para sistemas especialistas, a consideramos bastante satisfatória, devido à eficácia obtida com o resultado do desenvolvimento da versão 1.0 do SEC e seus refinamentos.

Esta proposta de controle da qualidade necessita apenas ter uma abordagem mais genérica, tratando de outras técnicas de verificação e validação adequadas a diferentes tipos de projetos, considerando o conhecimento tratado e a equipe de trabalho. Já estão em andamento na COPPE trabalhos nesta direção.

Segundo o *Software Engineering Institute*, as organizações podem ser classificadas quanto à maturidade de seu processo de desenvolvimento de aplicações em cinco níveis: inicial (caos), repetitivo, definido, gerenciado e otimizado (Arthur, 1993). A partir dessa classificação podemos considerar que o *Núcleo de Pesquisa e Desenvolvimento de Software Médico* da UCCV/FBC em relação ao processo de desenvolvimento de sistemas baseados em conhecimento encontra-se num estado intermediário, caracterizado por uma estrutura de processo de desenvolvimento definida, treinada e seguida por engenheiros do conhecimento, com a utilização de métodos.

Capítulo V

Método de Desenvolvimento KADS-estendido

V.1. Introdução

Este capítulo apresenta uma extensão para o método KADS. O método KADS-estendido foi definido com base na experiência obtida no desenvolvimento de sistemas baseados em conhecimento utilizando o método KADS.

Para esta definição, foram utilizados, também, conceitos e experiência no desenvolvimento de sistemas computacionais, em geral, com outros métodos de desenvolvimento, tais como, métodos estruturados (Análise e Projeto Estruturado (DeMarco, 1989), Análise Essencial (McMenamim e Palmer, 1991), (Yourdon, 1990), SADT (*Structured Analysis and Design Technique)* (Ross e Schoman, 1977), (Ross, 1985), (Davis, 1990), método estruturado para sistemas baseados em conhecimento (Keller, 1987) e Engenharia da Informação (Martin, 1990), (Martin, 1991). Também foram considerados conceitos e diagramas de alguns métodos orientados a objetos: OOA/OOD (*Object Oriented Analysis/ Object Oriented Design*) de Coad e Yourdon (1992), OMT (*Object Modeling Technique*) de Rumbaugh et alli (1991), e OOIE (*Object Oriented Information Engineering*) de Martin e Odell (1992).

V.2 Visão Geral do KADS-estendido

O método KADS, conforme proposto em (Schreiber, 1992) (Wielinga, Schreiber, e Breuker, 1992), (Schreiber, Wielinga e Breuker, 1993) prevê a geração dos seguintes modelos: Modelo Organizacional, Modelo de Aplicação, Modelo de Tarefas, Modelo de Cooperação, Modelo de Especialidade, Modelo Conceitual e Modelo de Projeto.

Entretanto, a ênfase deste método está no Modelo de Especialidade, pois este modelo consiste na realização da atividade central do desenvolvimento de um sistema

baseado em conhecimento. Os outros modelos estão ainda em fase de definição, existindo poucos artigos sobre o Modelo de Cooperação (Greef e Breuker, 1992) e (Breuker e Greef, 1993). O Modelo de Projeto é definido em termos da operacionalização do Modelo de Especialidade, apresentando uma alternativa para realização do projeto e construção (Schreiber, 1993).

Por esses motivos nos concentramos no Modelo de Especialidade, pois no nosso entender, o Modelo de Cooperação só deve ser definido na fase de projeto ao se verificar a parte automatizada ou não do sistema.

Nossa experiência no uso do Modelo de Especialidade, principalmente no projeto SEC, mostrou-nos que este não é suficiente para a especificação e projeto de um sistema baseado em conhecimento. O entendimento do modelo pelos especialistas, especialmente no que se refere à estrutura de inferência foi difícil. A avaliação feita pela equipe de desenvolvimento do SEC ao final do 1º. estágio (Anexo I), mostrou que esta modelagem foi útil no entendimento do processo de raciocínio dos especialistas, mas ao se implementar o sistema não foi utilizado o Modelo de Especialidade.

Em projetos desenvolvidos com alunos na COPPE, os comentários foram os mesmos. Alguns utilizaram a proposta descrita em Schreiber (1993), concluindo que a transformação do Modelo de Especialidade em uma linguagem lógica é um pouco artificial, dificultando o entendimento do programa e do problema resolvido por este.

Para resolver estas questões, definimos uma extensão do método KADS que denominamos KADS-estendido. Este método apoia o desenvolvimento de sistemas baseados em conhecimento de acordo com o ciclo de vida definido no capítulo anterior. No KADS-estendido tem-se, considerando os mesmos princípios propostos no método KADS, os seguintes modelos: *Modelo do Domínio do Problema*, *Modelo de Especialidade*, *Modelo Lógico* e *Modelo Físico*.

O ***Modelo de Domínio do Problema***, engloba a finalidade dos modelos organizacional, de aplicação e de tarefas propostos no KADS. Este modelo tem como objetivo definir o domínio do problema a ser tratado pelo sistema em termos das tarefas necessárias para resolução do problema e da abrangência do conhecimento com uma visão geral deste problema na organização.

O ***Modelo de Especialidade***, como no KADS, tem a finalidade de especificar o conhecimento necessário para executar as tarefas associadas à solução do problema a ser

tratado pelo sistema. Este modelo é definido a nível conceitual sem considerar aspectos de implementação.

Finalmente, tem-se o ***Modelo Lógico*** e o ***Modelo Físico*** onde está a maior contribuição do KADS-estendido. Sua inclusão foi motivada pelas observações da equipe de desenvolvimento do SEC, ao final do 1º estágio de desenvolvimento do projeto, que sentiu falta no método KADS de um apoio efetivo para a fase de projeto e na passagem para a implementação do sistema. Esses modelos, sempre que possível, utilizam a mesma notação do *Modelo de Especialidade*.

Nas próximas seções definimos, com detalhes, cada um destes modelos e a Figura V.1 apresenta uma visão geral do KADS-estendido

V.3 Modelo do Domínio do Problema

Durante o desenvolvimento da 1ª. versão do sistema SEC, o *Modelo do Domínio do Problema* foi elaborado de forma descritiva e informal. Entretanto, esta não é a forma mais adequada para sua definição existindo, atualmente, métodos que podem ser utilizados para realizar a *análise do domínio do problema* (Arango, 1994), (Schreiber, Wielinga e Breuker, 1993).

O *Modelo do Domínio do Problema* do KADS-estendido é composto de um *Diagrama de Tarefas* e de uma *Taxonomia Geral do Domínio do Problema*, sendo ambos gerados durante a atividade *Aquisição do Conhecimento Geral*, do processo de desenvolvimento definido no capítulo IV.

O ***Diagrama de Tarefas*** é baseado no *Modelo de Cooperação* do método KADS (Schreiber et alii 1993) e em métodos de análise do domínio (Arango, 1994). Este diagrama é uma decomposição do domínio do problema em tarefas, podendo conter operadores lógicos evidenciando o fluxo das tarefas. A Figura V.2 apresenta um exemplo do *Diagrama de Tarefas*.

A ***Taxonomia Geral do Domínio do Problema*** deve ser elaborada com base no esquema de classificação por facetas proposto por Prieto-Diaz (1987). Este esquema pressupõe a classificação de um domínio do problema sob diferentes aspectos, podendo fornecer um esquema abrangente de taxonomia. Na Figura V.3 apresentamos, como exemplo, parte da taxonomia elaborada para o projeto SEC.

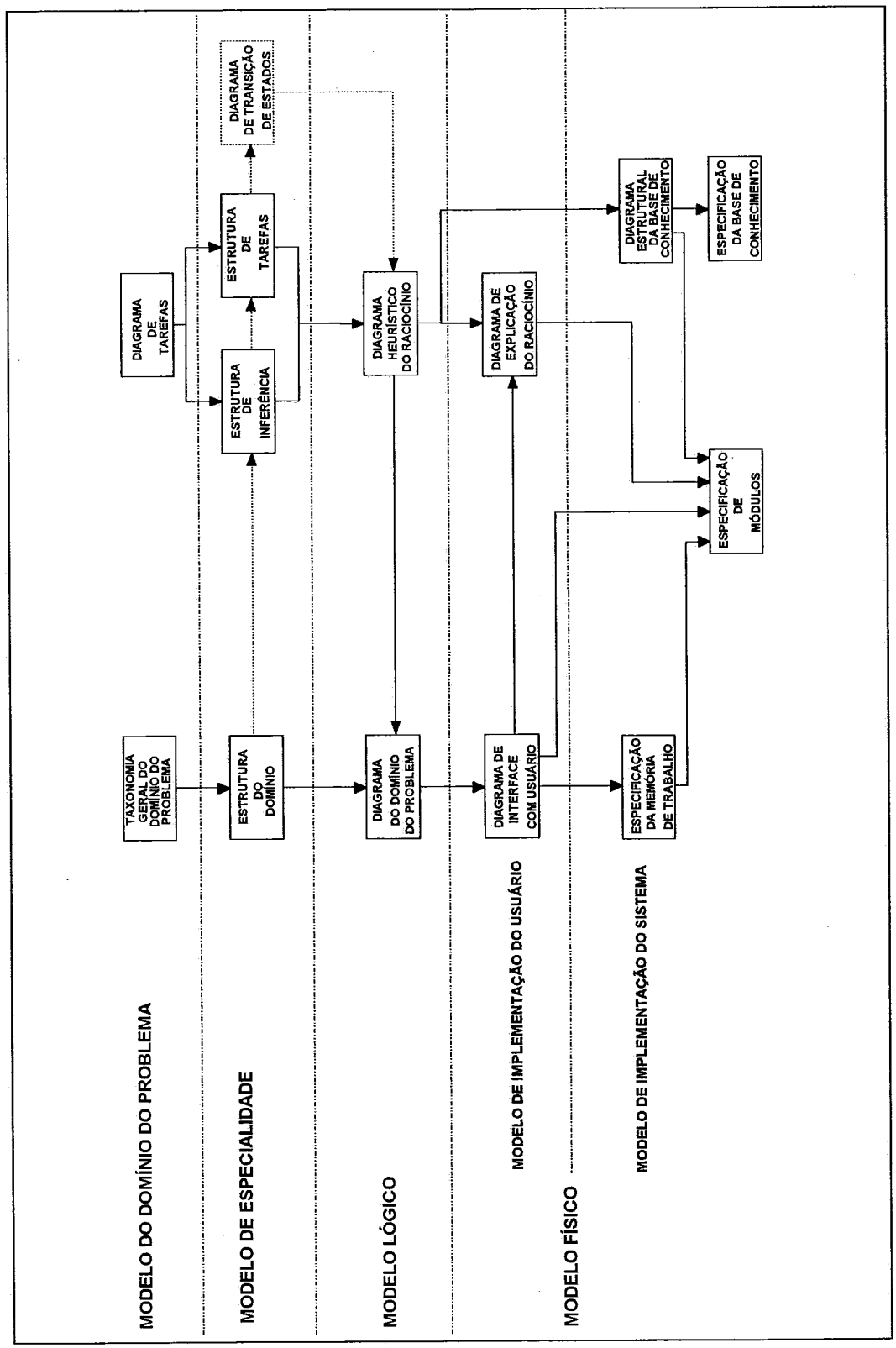


Figura V.1: Estrutura Geral do KADS-estendido

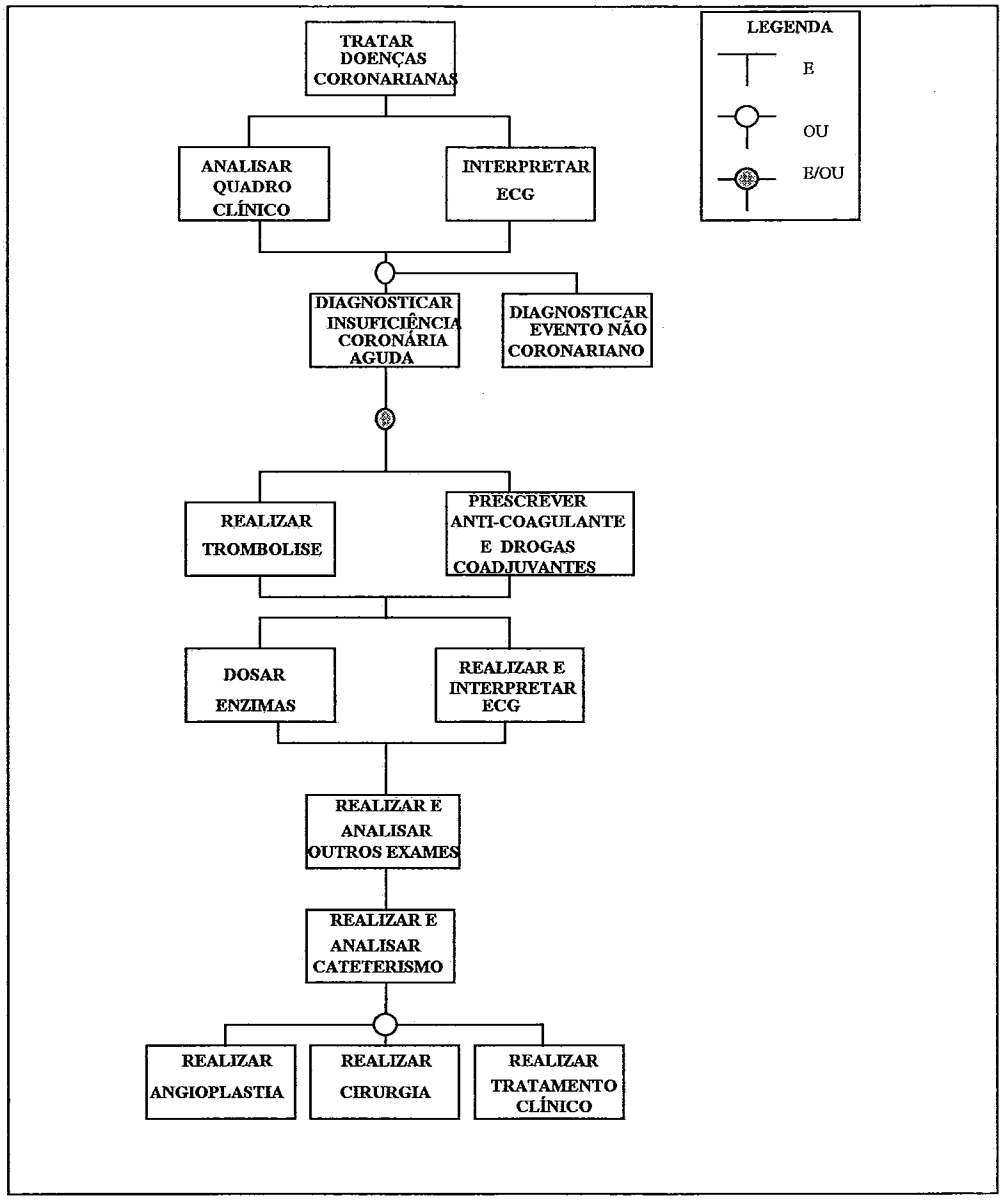
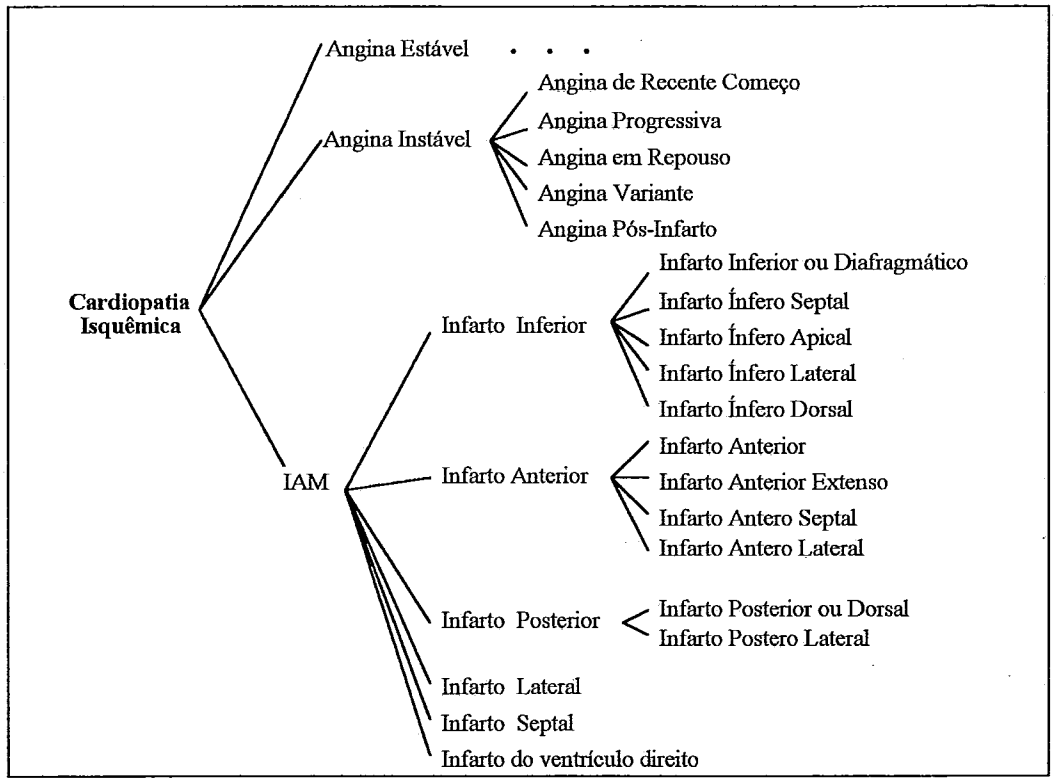


Figura V.2: Exemplo de Diagrama de Tarefas do SEC

V.4 Modelo de Especialidade

O Modelo de Especialidade do KADS-estendido incorpora sugestões feitas ao longo do desenvolvimento do SEC, pela equipe de especialistas e de desenvolvimento, com relação à forma de representação proposta no método KADS. Pretende-se com essas alterações obter uma melhor visualização do processo de raciocínio e consistência entre as estruturas das diversas camadas do conhecimento.

TAXONOMIA DE CARDIOPATIA ISQUÊMICA



Classificação Eletrocardiográfica do IAM segundo Área Lesada e Ocorrência:

IAM	Alterações Eletrocardiográficas	Artéria, geralmente, "culpada"	Ocorrência emergência
Antero Septal	V1 a V4	descendente anterior ou diagonal	**
Septal	V1,V2 e poucas alterações em V3	descendente anterior	*
Ínfero Lateral	DII, DIII,aVF, V5 e V6		
...	...	...	...

branco - pouco comum * - comum ** - muito comum

Classificação segundo as Ondas do ECG:

Situação	Achados Eletrocardiográficos
IAM Antero Septal	supradesnivelamento de ST, evoluindo para aparecimento de T negativo (isquêmico);evolutivamente, aparecimento de onda QS(2) (necrose).
IAM Septal	supradesnivelamento de ST, evoluindo para aparecimento de ondas T negativas; evolutivamente, aparecimento de ondas q(1) (necrose).
Angina Instável	presença de onda T negativa (isquêmica) ou infradesnivelamento de ST.
...	...

(1) ondas Q pequenas (2) aparecimento de ondas Q e S com ausência de onda R

Figura V.3: Exemplo de parte da Taxonomia do SEC utilizando Esquema de Classificação por Facetas

As modificações feitas ao método KADS referem-se a alterações na estrutura de inferência, na apresentação do modelo e na especificação das meta-classes, de forma a evidenciar, explicitamente, a consistência entre as camadas de domínio e de inferência. O *Modelo de Especialidade* do *KADS-estendido* é composto de três camadas[1] com as seguintes estruturas: *Estrutura de Domínio*, *Estrutura de Inferência* e *Estrutura de Tarefas*. O *Modelo de Especialidade* do *KADS-estendido* contém ainda um *Diagrama de Transição de Estados*.

A ***Estrutura de Domínio*** do KADS utiliza a Linguagem de Definição do Domínio (DDL-"Domain Description Language") (Schreiber, Wielinga e Breuker, 1993). No *KADS-estendido* utiliza-se, apenas, a representação gráfica da linguagem, não diferenciando as representações de instância e tuplas por se considerar que estes conceitos estão embutidos nos conceitos de objetos e relações. A Figura V.4 apresenta a *Estrutura de Domínio* do Sistema SEC Versão 2.0, utilizando-se o *KADS-estendido*.

Para facilitar o entendimento do modelo pelos especialistas e com base em suas sugestões, a ***Estrutura de Inferência*** do KADS foi alterada, nos seguintes aspectos:

- as fontes de conhecimento foram denominadas tarefas/inferências;
- as tarefas/inferências que podem ser repetidas ao longo do raciocínio, são representadas de forma explicita (várias elipses, sendo as de baixo com linha tracejada);
- a execução repetida de um conjunto de tarefas/inferências é representada com uma seta de linha tracejada;
- as meta-classes são especificadas considerando-se os componentes definidos na *Estrutura do Domínio*, para que haja uma consistência explícita entre as camadas de domínio e de inferência.

A *Estrutura de Inferência* e a *Estrutura de Tarefa* (Figura V.5) devem estar representadas numa mesma página, lado a lado, para uma melhor compreensão do processo de raciocínio. A *Estrutura de Tarefas* pode ser representada de forma

[1] Estas são as mesmas camadas propostas no KADS. A camada de estratégia, presente na versão inicial do KADS, normalmente, não é utilizada e já foi suprimida na nova versão do KADS (CommonKADS) (Velde, 1994).

procedural ou através de uma árvore de tarefas, sendo aconselhável, a partir de nossa experiência, que sejam feitas as duas representações.

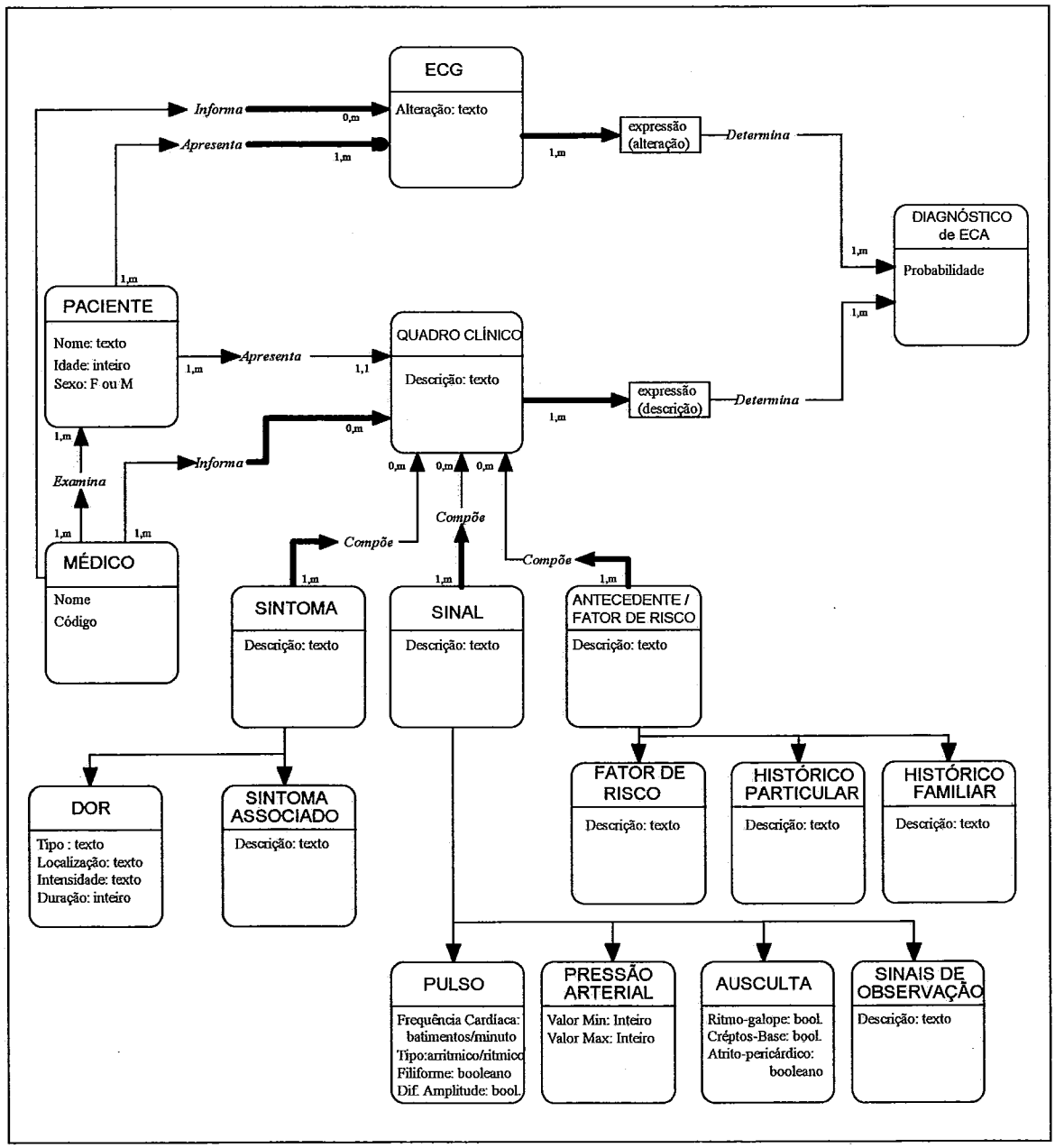


Figura V.4: Estrutura de Domínio do SEC versão 2.0 em DDL[1]

[1] A Linguagem de Definição de Domínio (DDL) está descrita no Anexo VI.

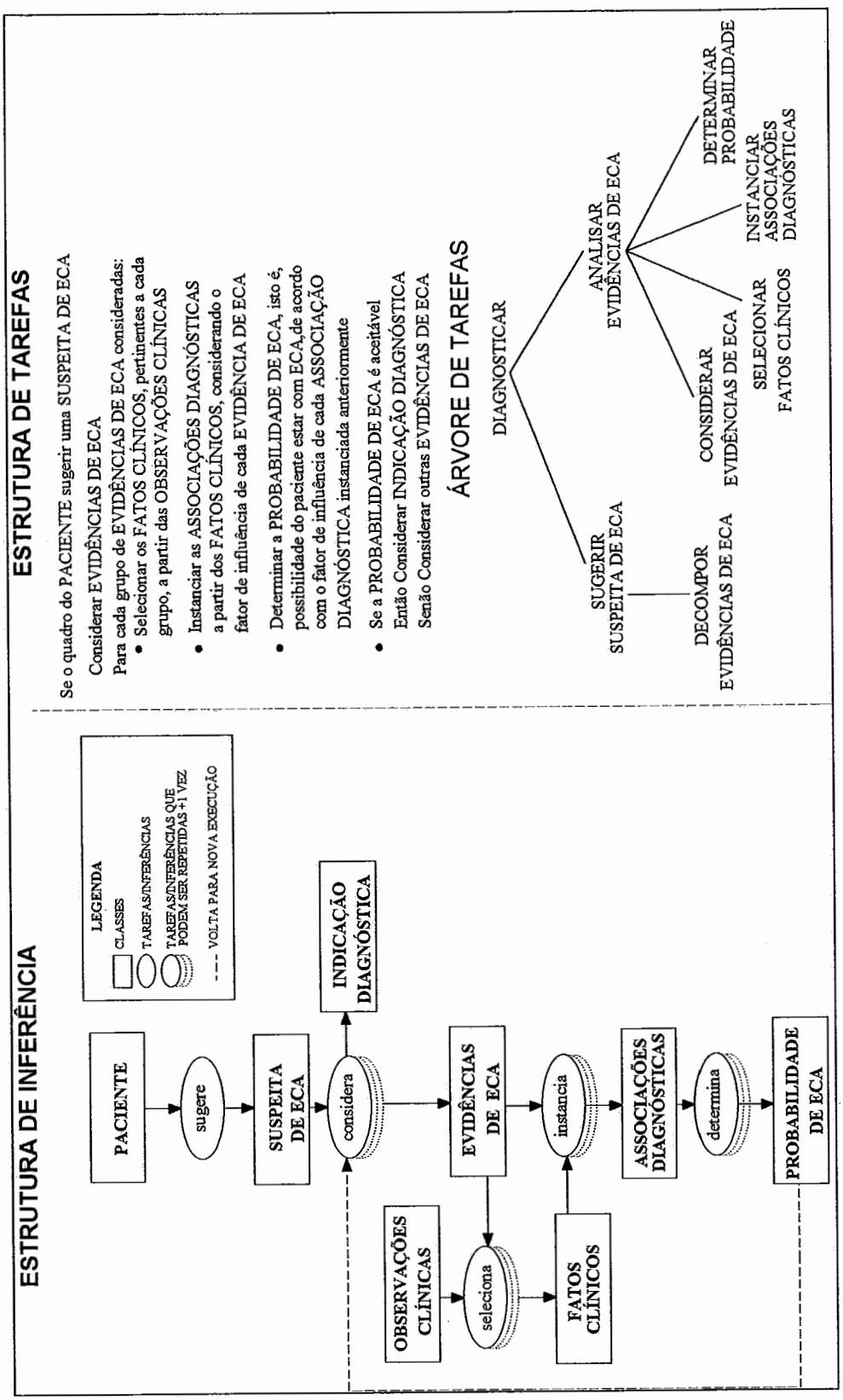


Figura V.5: Estrutura de Inferência e Estrutura de Tarefas do SEC Versão 2.0

O *Modelo de Especialidade* no *KADS-estendido* se completa com a construção do ***Diagrama de Transição de Estados*** da classe solução cujo objetivo é mostrar explicitamente como o sistema se comportará. Este diagrama é opcional[1] e tem como objetivo tornar mais claras as interações do sistema com o ambiente externo, mostrando os estados do processo de raciocínio, as condições de mudança de estado e as tarefas realizadas. Essas tarefas devem ser consistentes com as tarefas descritas na *Estrutura de Tarefas*. A Figura V.6 apresenta o *Diagrama de Transição de Estados* da classe Diagnóstico do sistema SEC Versão 2.0, utilizando em sua construção uma combinação das notações propostas no método OMT (Rumbaugh et alli, 1991) e na Análise Estruturada Moderna (Yourdon, 1990).

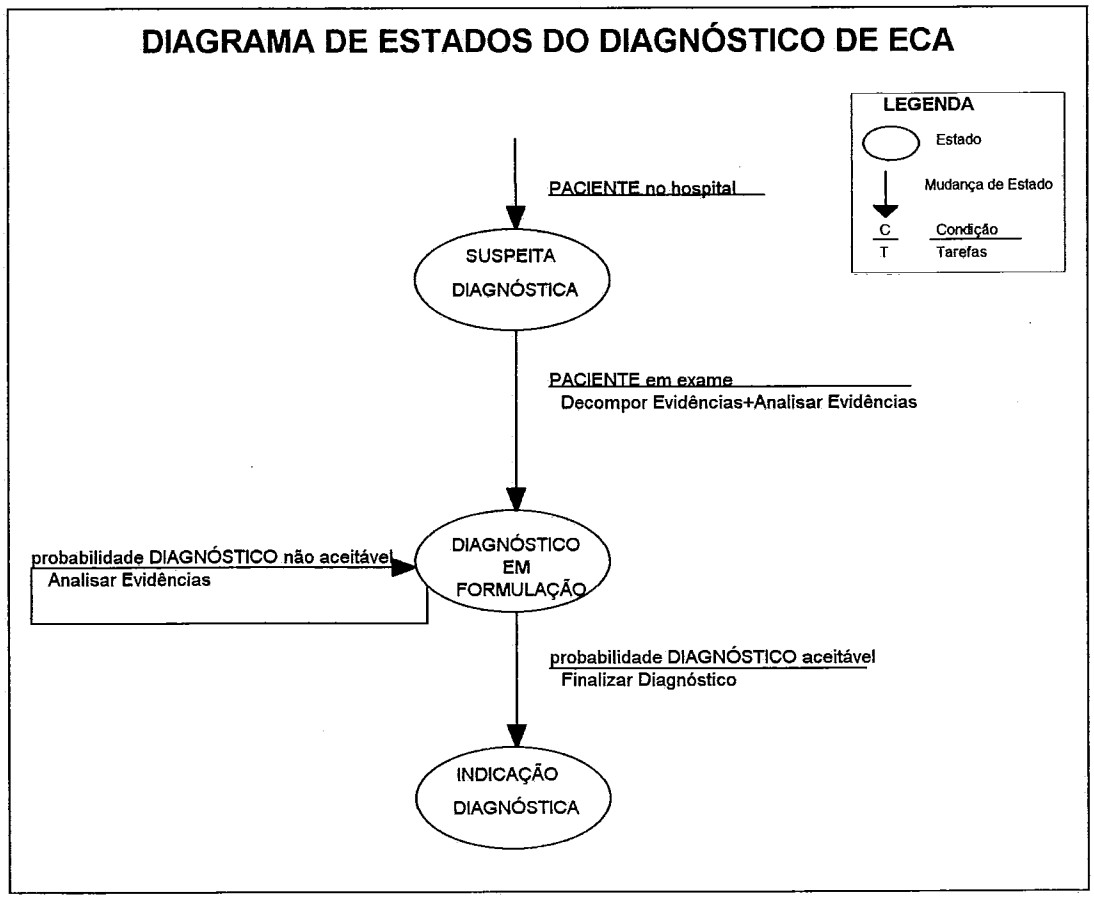


Figura V.6: Digrama de Transição do SEC Versão 2.0

[1] O Diagrama de Transição de Estados da classe solução é opcional pois é uma variação do processo de raciocínio, tornando claras as interações entre usuário e o sistema baseado em conhecimento. Este diagrama, entretanto, é útil para uma melhor visualização do SBC pelos especialistas.

V.5 Modelo Lógico

No *Modelo Lógico* tem-se a maior contribuição do *KADS-estendido*. Este modelo surgiu devido à dificuldade encontrada em nossas experiências de uso do KADS na COPPE e na FBC. Verificou-se, assim, que era necessário ter-se uma especificação do sistema com uma visão mais próxima do desenvolvedor. No método KADS existem várias propostas para essa passagem, fundamentadas em linguagens formais (Schreiber, Wielinga e Breuker, 1993), (Voβ e Karbach, 1993). A nossa, bem sucedida, experiência de emprego de métodos semi-formais/gráficos e o perfil das equipes de desenvolvimento, normalmente disponíveis em empresas, não habituadas ao uso de métodos formais, nos fizeram optar por continuar usando uma linguagem gráfica no *Modelo Lógico*.

O *Modelo Lógico* está composto por dois diagramas: o *Diagrama Heurístico do Raciocínio* e o *Diagrama do Domínio do Problema*, que são construídos a partir do *Modelo de Especialidade* do *KADS-estendido*.

O ***Diagrama Heurístico do Raciocínio*** tem como principal objetivo definir a estrutura de investigação do processo de solução do problema que originará a estrutura da base de conhecimento. Este diagrama é definido a partir do *Modelo de Especialidade* do *KADS-estendido*.

O ***Diagrama do Domínio do Problema*** é, na realidade, uma nova visão da *Estrutura de Domínio*, definida usando a Linguagem de Definição do Domínio. Este diagrama é gerado a partir do *Diagrama Heurístico do Raciocínio*. A representação gráfica dos dois diagramas evidencia sua compatibilidade. Assim, tem-se garantida a consistência entre o *Diagrama da Camada do Domínio do Problema* e o *Diagrama Heurístico do Raciocínio* o que é fundamental para a fase de *projeto*, pois a partir deles deverá ser definido o *Modelo Físico* do sistema.

V.5.1 Diagrama Heurístico do Raciocínio

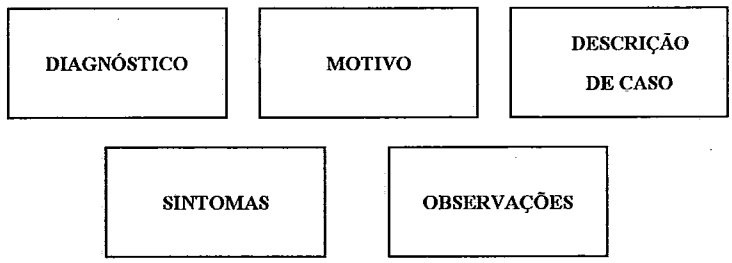
O *Diagrama Heurístico do Raciocínio* está baseado nos seguintes conceitos:

- **Classes heurísticas,** que são classes que fazem parte do processo de raciocínio. São representadas por um retângulo colorido (pode-se utilizar qualquer côr conforme a preferência do desenvolvedor) com o nome dentro.

Exemplo:

- **Classes de entrada ou saída,** que são classes introduzidas ou apresentadas ao ambiente externo do sistema. São representadas por um retângulo sem côr com o nome dentro.

  Exemplo:



- **Classes solução** que são classes resultantes do processo de raciocínio (objetivo da tarefa). Podem ser classes heurísticas ou de saída.

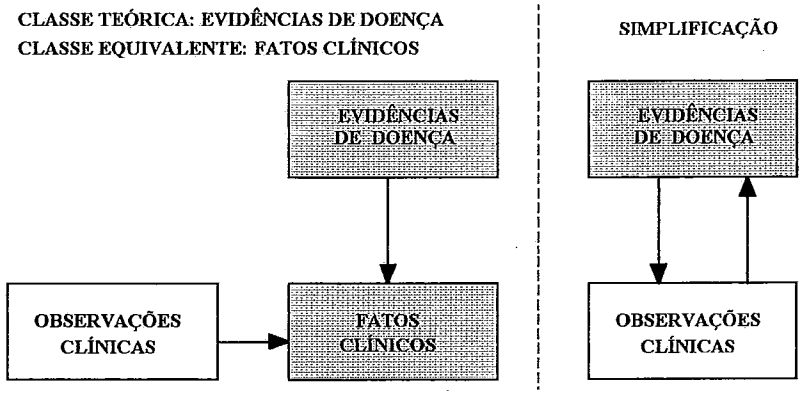
  Exemplo: classe DIAGNÓSTICO, classe PLANO, classe PROJETO.

- **Classes teóricas** que são classes heurísticas identificadas na *Estrutura de Inferência* do *Modelo de Especialidade*, e que só fazem parte da investigação de uma solução.

  Exemplo: classe EVIDÊNCIA, classe HIPÓTESES, classe NORMA.

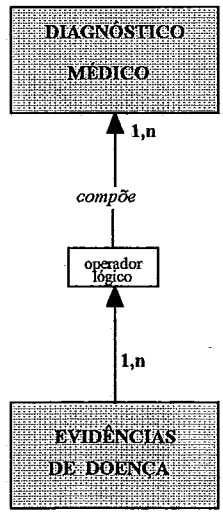
- **Classes equivalentes:** As classes teóricas podem ser equivalentes a classes heurísticas instanciadas com valores introduzidos pelo ambiente externo ao sistema. Essas classes são identificadas e eliminadas do *Diagrama Heurístico do Raciocínio* através da fusão das duas classes, permanecendo o nome de maior significado semântico para o domínio do problema. As setas da classe eliminada são associadas à classe equivalente que permaneceu no diagrama. Tem-se, assim, uma simplificação do diagrama.

Exemplo:



- **Associações lógicas** são combinações entre objetos e operadores lógicos (ou, e, "ou exclusivo", não, ...).

  Exemplo:



V.5.2 Passos para Construção do Diagrama Heurístico do Raciocínio

O *Diagrama Heurístico do Raciocínio* é obtido através da aplicação de um processo de transformação no *Modelo de Especialidade*, especificamente, a partir da *Estrutura de Inferência e da Estrutura de Tarefas*.

A seguir são definidos os procedimentos necessários para a construção do *Diagrama Heurístico do Raciocínio*.

i) O primeiro passo na construção do *Diagrama Heurístico do Raciocínio* consiste na transformação da *Estrutura de Inferência.* Para isto deve-se:

⇒ Identificar as tarefas/inferências manuais da *Estrutura de Inferência*, retirando-as do diagrama;

⇒ Retirar as tarefas/inferências da *Estrutura de Inferência*, mantendo as classes e as setas direcionadas às classes que as tarefas/inferências apontam;

⇒ Eliminar as setas relacionadas à execução repetitiva de inferências/tarefas, dado que esse tipo de seta representa, apenas, uma repetição do processo heurístico.

A Figura V.7 apresenta o diagrama obtido após a realização deste primeiro passo, no exemplo da Figura V.5.

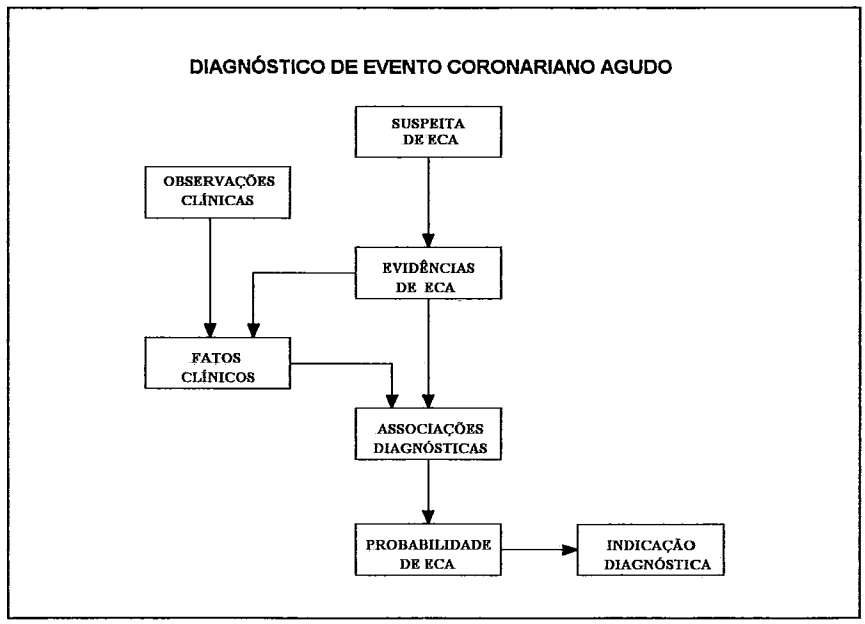


Figura V.7: Construção do Diagrama Heurístico do Raciocínio do SEC versão 2.0: diagrama obtido após a realização do passo *i)*

ii) Identificar as classes que fazem parte do processo de raciocínio da solução do problema, colocando-as em outra côr. As outras classes são entradas ou saídas que serão introduzidas ou apresentadas ao ambiente externo do sistema (Figura V.8).

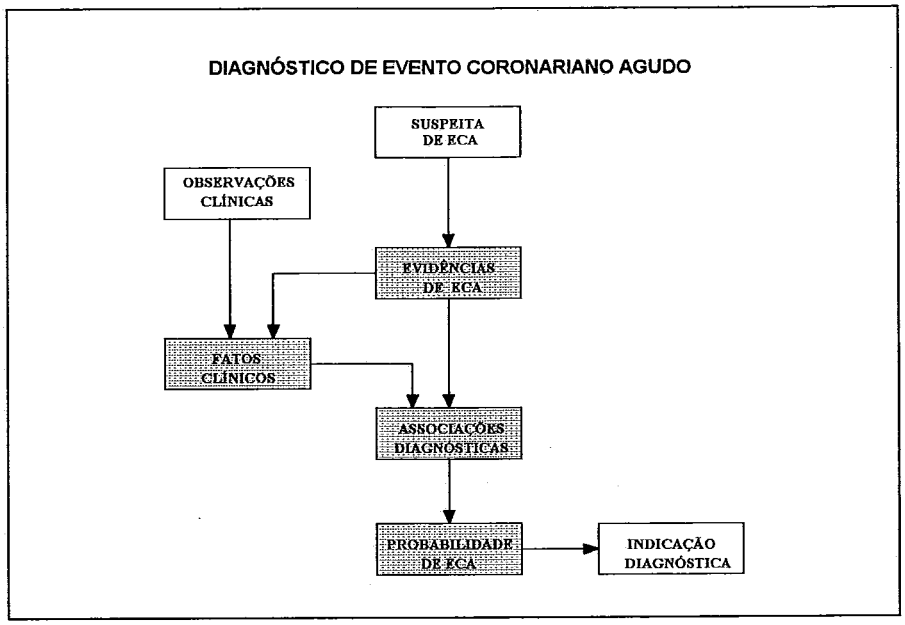


Figura V.8: Construção do Diagrama Heurístico do Raciocínio do SEC versão 2.0: diagrama obtido após a realização do passo *ii)*

iii) Identificar as classes onde ocorrem o início e o fim da execução da tarefa genérica modelada, colocando um círculo com a identificação interna de *i* (início) ou *f* (fim) (Figura V.9).

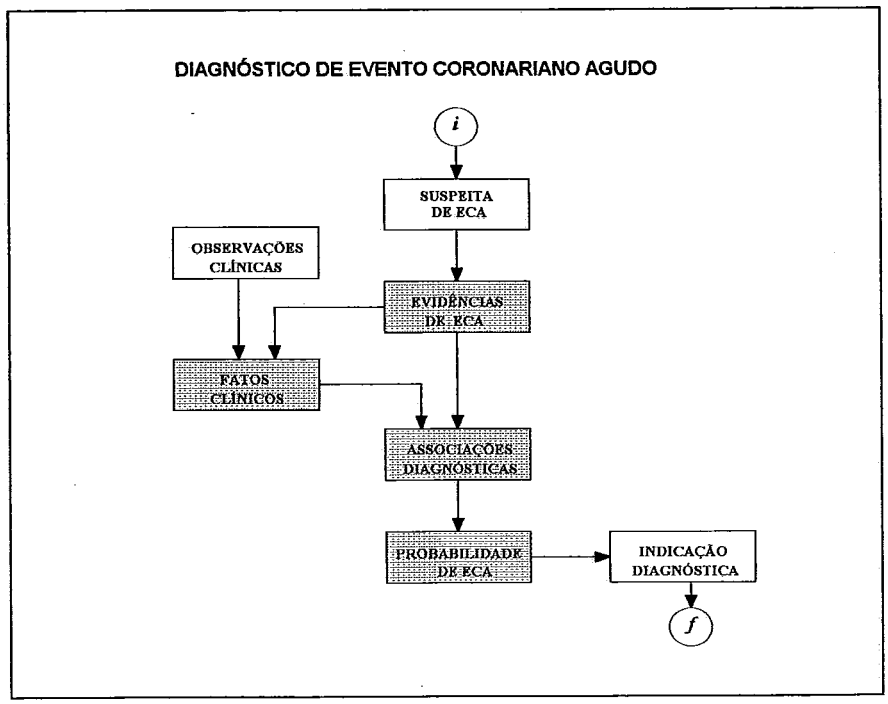


Figura V.9: Construção do Diagrama Heurístico do Raciocínio do SEC versão 2.0: diagrama obtido após a realização do passo *iii)*

iv) Alterar a disposição das classes no diagrama da seguinte forma:

⇒ A classe solução, isto é, a classe que aponta para o fim, deve ficar no topo do diagrama;

⇒ As classes que fazem parte do raciocínio devem ser colocadas uma em baixo da outra, na mesma coluna da classe solução;

⇒ A ordem das classes do raciocínio deve obedecer ao caminho invertido das setas, ou seja, elas serão percorridas de trás para frente, a partir da classe solução. As classes que tiverem duas ou mais classes apontando para ela deverão ter o relacionamento entre essas classes analisado. Caso haja entre essas classes uma seta, será dada preferência à classe que aponta, também, para esta outra classe;

⇒ As classes de entrada ficam à esquerda das classes do raciocínio.

A Figura V.10 apresenta o diagrama obtido após a realização deste passo.

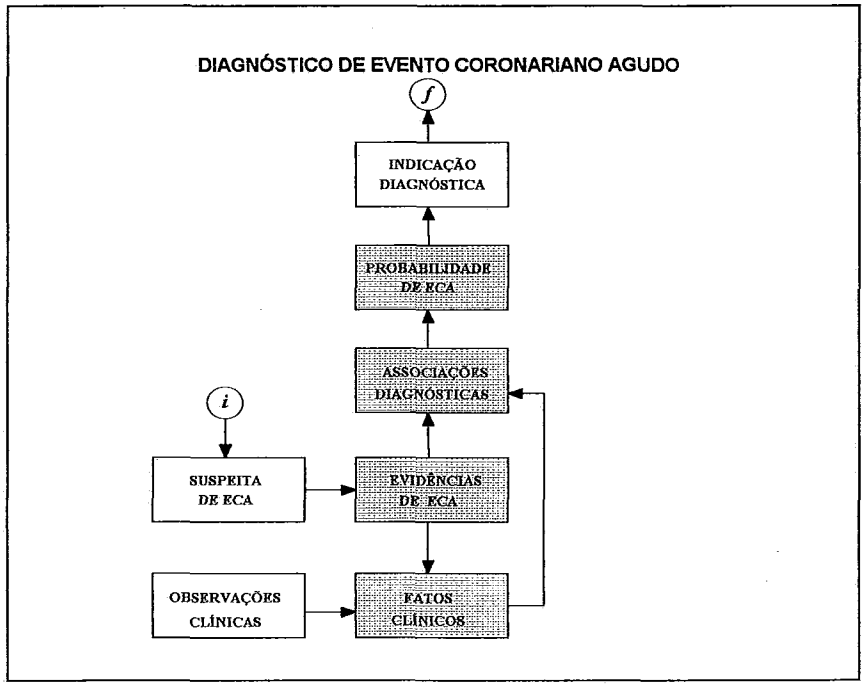


Figura V.10: Construção do Diagrama Heurístico do Raciocínio do SEC versão 2.0: diagrama obtido após a realização do passo *iv)*

v) Colocar o nome da tarefa correspondente a cada classe em cima do nome das classes:

⇒ Os nomes de tarefas devem ser verbos no infinitivo e encontram-se descritos na *Estrutura de Inferência* e na *Estrutura de Tarefas*;

⇒ O nome de uma classe solução que faz parte do processo de raciocínio pode ser o nome da tarefa genérica (diagnosticar, planejar, avaliar,...);

⇒ As classes que são entradas introduzidas pelo ambiente externo do sistema devem utilizar o mesmo nome da tarefa/inferência manual da *Estrutura de Inferência* ou, caso este nome não exista, um dos seguintes verbos: informar, observar, obter ou introduzir;

⇒ Nomes de classes que são saídas apresentadas ao ambiente externo do sistema devem utilizar um dos seguintes verbos: informar, mostrar ou apresentar.

Na Figura V.11 encontra-se o diagrama obtido após a realização deste passo.

Figura V.11: Construção do Diagrama Heurístico do Raciocínio do SEC versão 2.0: diagrama obtido após a realização do passo *v)*

vi) Analisar as classes de entrada que não fazem parte do início da execução da tarefa do sistema, com o objetivo de tornar explícito o diálogo entre o processo de raciocínio e o ambiente externo:

⇒ Considerar o momento em que deve ser introduzida a informação pelo ambiente externo, colocando explicitamente setas para essas classes de entrada. Neste momento, podem ser identificadas classes téoricas e suas classes equivalentes que devem ser simplificadas em uma só, conforme definição anterior.

A Figura V.12 apresenta o diagrama obtido após a realização deste passo.

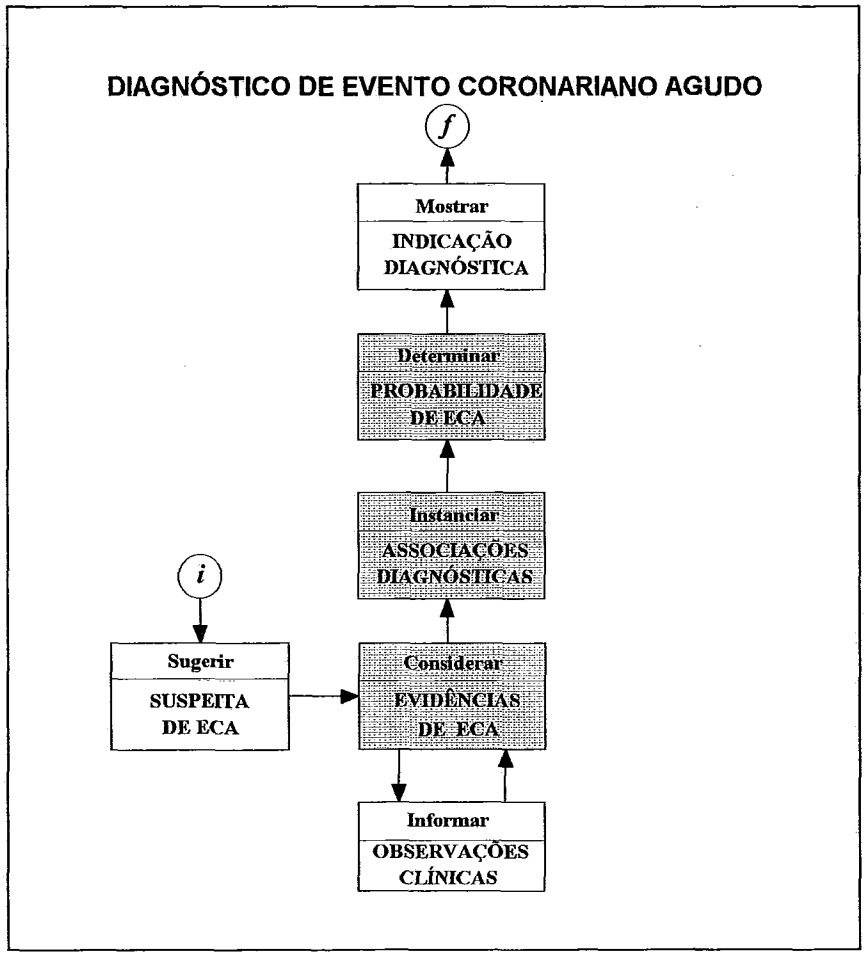


Figura V.12: Construção do Diagrama Heurístico do Raciocínio do SEC versão 2.0: diagrama obtido após a realização do passo *vi)*

vii) Identificar os relacionamentos entre as classes, a cardinalidade entre elas e suas associações lógicas, isto é, combinações entre objetos e operadores lógicos. Este passo deve ser realizado com o auxílio da *Estrutura de Domínio,* de forma que:

⇒ Os relacionamentos são identificados a partir das setas entre as classes. Para cada seta devem ser identificados os relacionamentos entre as classes que são representados na *Estrutura do Domínio.*
Em alguns casos, as setas são entre classes que não estão localizadas logo abaixo ou acima uma das outras. Neste caso, essas setas são transformadas em relacionamentos, passando por classes intermediárias (novas ou já existentes), que permitem estabelecer uma estrutura linear entre as classes;

⇒ Novos relacionamentos podem ser identificados a partir da *Estrutura do Domínio*;

⇒ Classes no plural normalmente mostram que essas classes compõem outra classe através do relacionamento de um conjunto de seus objetos. Este conjunto deve ser especificado com a mesma notação usada na *Estrutura do Domínio.* O nome dessas classes deve ser colocado no singular;

⇒ Novas classes podem ser identificadas a partir da *Estrutura do Domínio,* pois podem existir classes neste diagrama que compõem algumas classes do raciocínio. Assim sendo, essas novas classes devem ser definidas no diagrama logo abaixo da classe decomposta com o relacionamento de composição e decomposição entre as duas classes. Caso exista alguma classe abaixo da classe decomposta, o seu relacionamento será com a nova classe introduzida no diagrama;

⇒ Algumas classes heurísticas podem ser um subconjunto da outra e por isso podem ser unificadas;

⇒ Todos os relacionamentos entre classes heurísticas devem ter cardinalidade expressa nos dois lados.

⇒ As classes que são entradas ou saídas ao ambiente externo do sistema só tem cardinalidade na parte inicial da seta, ou seja, no lado de fora do sistema nas classes de entrada e no lado de dentro nas classes de saída.

A Figura V.13 apresenta o diagrama obtido após a realização deste passo.

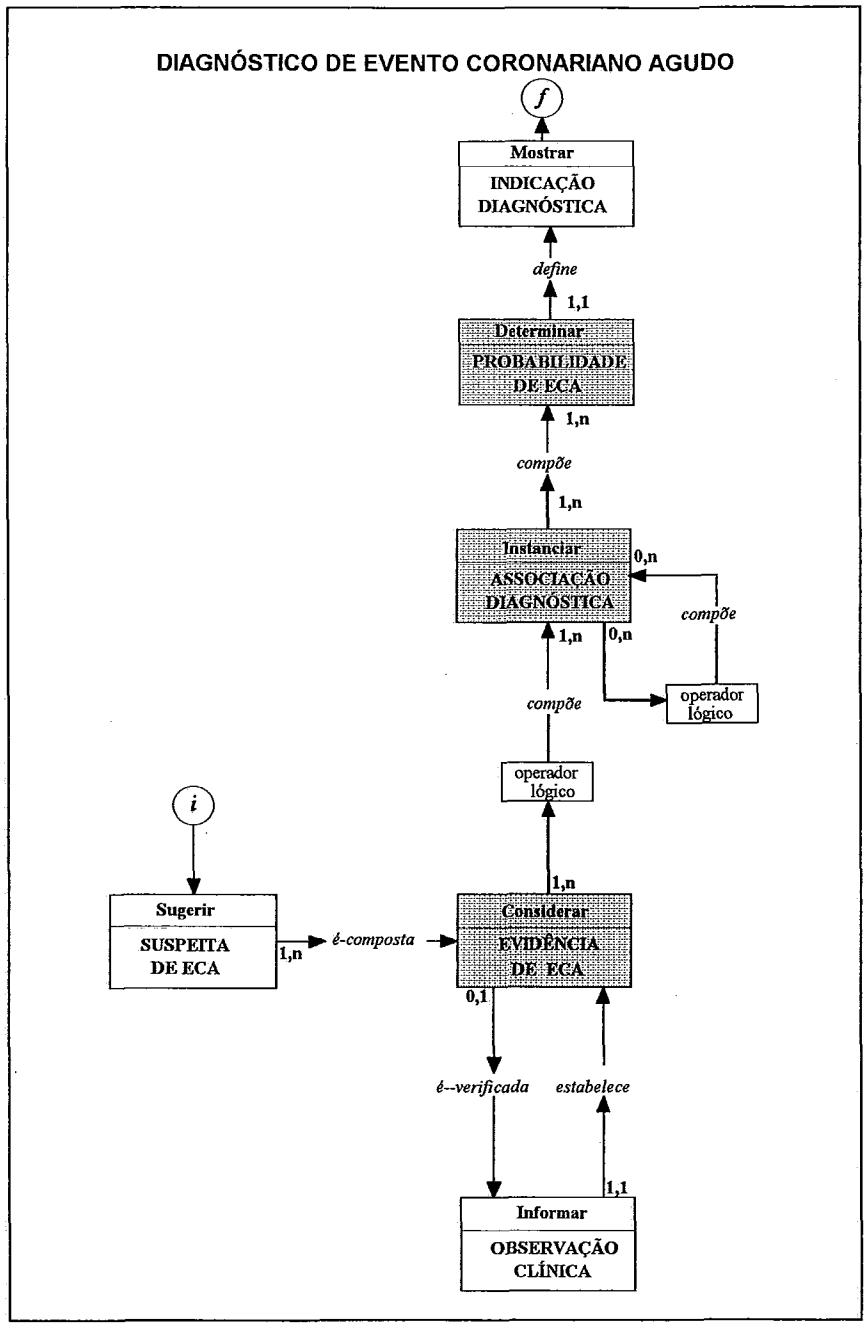


Figura V.13: Construção do Diagrama Heurístico do Raciocínio do SEC versão 2.0: diagrama obtido após a realização do passo *vii)*

viii) Identificar os atributos relevantes ao processo de raciocínio que possam existir nas classes do diagrama. Este item deve ser realizado com o auxílio da *Estrutura do Domínio* e do *Diagrama de Transição de Estados*, da seguinte forma:

⇒ Os nomes dos atributos devem ser substantivos localizados abaixo do nome da classe;

⇒ Neste momento, podem ser identificadas classes do raciocínio que são na realidade estados ou atributos de uma determinada classse. Neste caso o nome da classe deve ser o mesmo da *Estrutura do Domínio*, apresentando também os atributos.

Na Figura V.14 encontra-se o diagrama obtido após a realização deste passo.

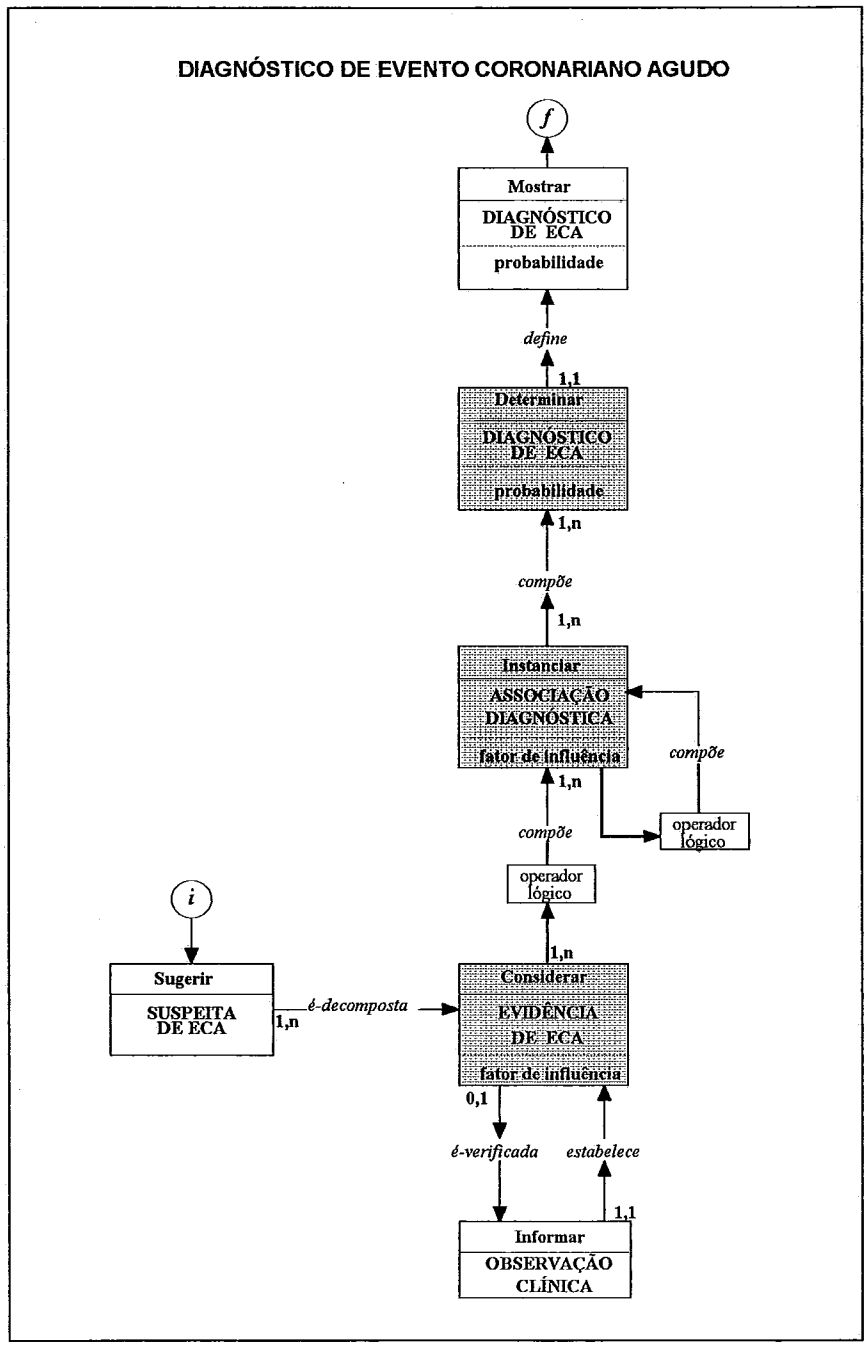


Figura V.14: Construção do Diagrama Heurístico do Raciocínio do SEC versão 2.0: diagrama obtido após a realização do passo *viii)*

ix) Simplificar o *Diagrama Heurístico do Raciocínio* retirando as classes de início e fim, permanecendo explicitamente as classes de entrada que serão introduzidas ao longo do processo de raciocínio. A identificação de início e fim deve permanecer e os nomes dos atributos de entrada ou saída devem ser explicitados quando necessário.

Obs.: Os passos vii e viii requerem uma análise do conhecimento, podendo necessitar de uma nova atividade de elicitação do conhecimento.

Na Figura V.15 temos o *Diagrama Heurístico do Raciocínio* correspondente ao *Modelo de Especialidade* do SEC Versão 2.0 (Figuras V.4, V.5 e V.6).

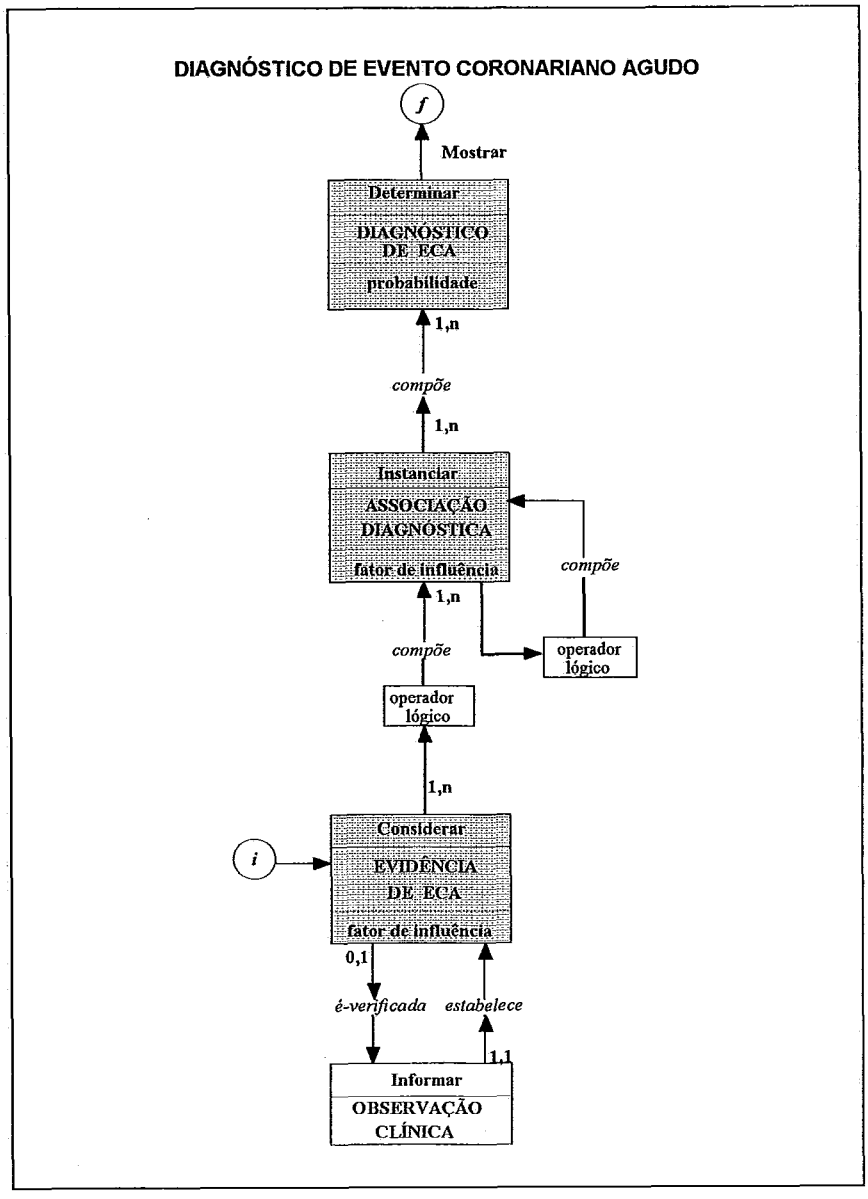


Figura V.15: Diagrama Heurístico do Raciocínio do SEC Versão 2.0

V.5.3 Diagrama do Domínio do Problema

O *Diagrama do Domínio do Problema* é uma nova visão da *Estrutura do Domínio* do *Modelo de Especialidade*, onde estão definidas as classes presentes no *Diagrama Heurístico do Raciocínio*.

A seguir são definidos os passos necessários para construção do *Diagrama do Domínio do Problema* a partir do *Diagrama Heurístico do Raciocínio*:

i) Identificar as classes heurísticas na *Estrutura do Domínio*. Essas classes podem estar representadas por objetos, relacionamentos e/ou expressões. No caso de serem objetos ou expressões devem ser colocados na mesma côr que no *Diagrama Heurístico do Raciocínio*. As classes heurísticas podem corresponder, a um conjunto de classes e sendo assim, além da mudança de côr devem ser colocadas dentro de uma linha pontilhada com seu nome em cima e à esquerda;

ii) Identificar as classes que fazem parte do início do processo de raciocínio da solução do problema, colocando-as dentro de uma linha pontilhada e colocando em cima à esquerda o círculo com identificação interna de *i* (início);

iii) Identificar as classes que fazem parte da solução do problema do processo de raciocínio, colocando-as dentro de uma linha pontilhada e colocando em cima e à esquerda o círculo com identificação interna de *f* (fim);

iv) Identificar as classes que são introduzidas ao longo do processo de raciocínio, colocando-as dentro de uma linha pontilhada e colocando em cima e à esquerda o mesmo nome que está no *Diagrama Heurístico do Raciocínio*.

A Figura IV.16 apresenta o *Diagrama do Domínio do Problema* do SEC versão 2.0 correspondente à *Estrutura do Domínio* do *Modelo de Especialidade* (Figura IV.4) e ao *Diagrama Heurístico do Raciocínio* (Figura IV.15).

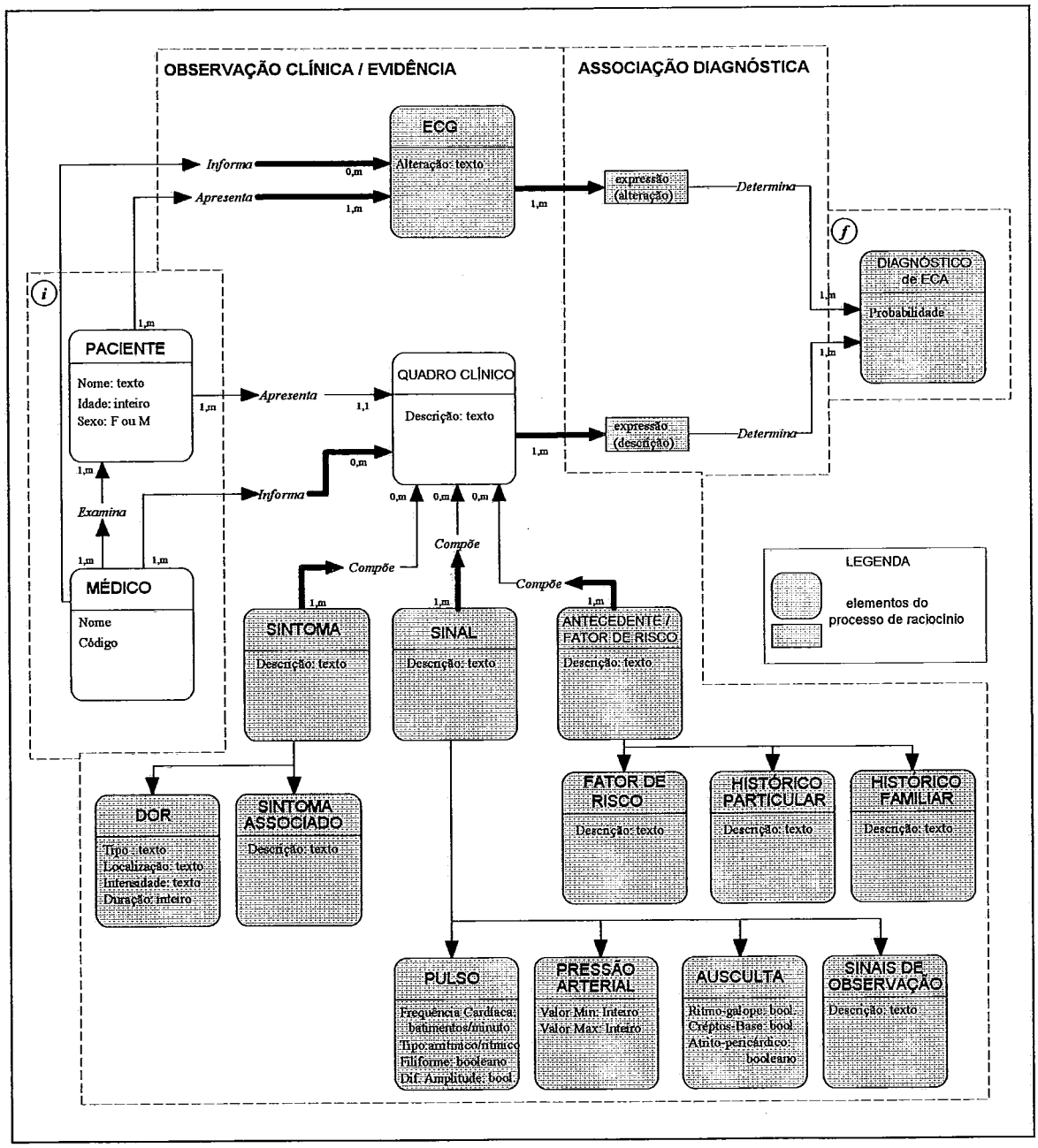


Figura V.16: Diagrama do Domínio do Problema do SEC versão 2.0

V.6 Modelo Físico

O *Modelo Físico* é construído a partir do *Modelo Lógico*, isto é, do *Diagrama do Domínio do Problema* e do *Diagrama Heurístico do Raciocínio*, sendo uma extensão desses diagramas. O objetivo deste modelo é definir o sistema numa representação possível de ser implementada na linguagem de programação escolhida.

No *KADS-estendido*, até este momento, o *Modelo Físico* foi definido e testado, considerando apenas linguagens de programação lógica e orientadas a objetos com regras, dada as experiências realizadas com os sistemas KBRF (Oliveira et alli, 1995) e SEC. Entretanto, outras definições podem ser feitas, sem dificuldades.

A *modelagem física* é composta de dois modelos: *Modelo de Implementação do Usuário* e *Modelo de Implementação do Sistema*. Esses modelos estão baseados na visão proposta na Análise Estruturada Moderna (Yourdon, 1990), procurando-se utilizar uma representação semelhante às usadas no modelo lógico do *KADS-estendido*.

O ***Modelo de Implementação do Usuário*** é composto do *Diagrama de Interface com o Usuário* e do *Diagrama de Explicação do Raciocínio*.

O ***Diagrama de Interface com o Usuário*** é construído a partir do *Diagrama do Domínio do Problema*, identificando-se explicitamente a hierarquia dos comandos e a interação do sistema com o usuário. Neste diagrama são definidas como deverão ser realizadas as consultas, os comandos disponíveis em cada tela, as informações fornecidas pelo usuário e as respostas do sistema.

O ***Diagrama de Explicação do Raciocínio*** é construído a partir do *Diagrama Heurístico do Raciocínio* e do *Diagrama de Interface com o Usuário*. Define as informações de explicação e o momento em que o sistema baseado em conhecimento fornece essas explicações a seus usuários. Deve ser apresentado junto com o *Diagrama de Interface com o Usuário* conforme exemplo da Figura V.17.

O ***Modelo de Implementação do Sistema*** possui o *Diagrama Estrutural da Base de Conhecimento*, a *Especificação da Base de Conhecimento*, a *Especificação da Memória de Trabalho* e a *Especificação dos Módulos*.

O ***Diagrama Estrutural da Base de Conhecimento*** é construído a partir das classes heurísticas do *Diagrama Heurístico do Raciocínio*, sendo definidas também as estruturas das regras do processo de raciocínio. Este diagrama contém a estrutura geral da base de conhecimento, isto é, a forma interna de representar o conhecimento no sistema e o formato das regras da base de conhecimento.

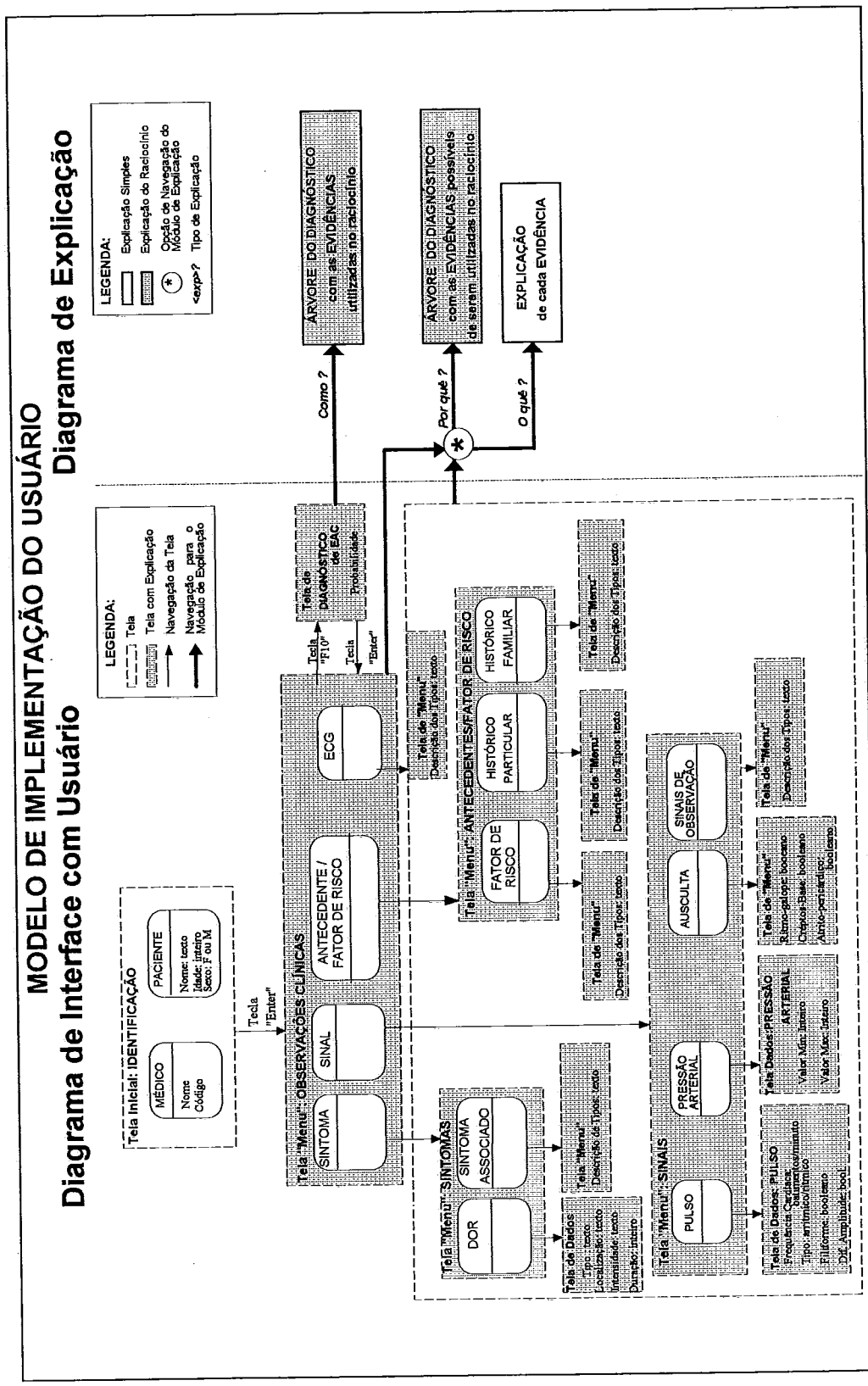
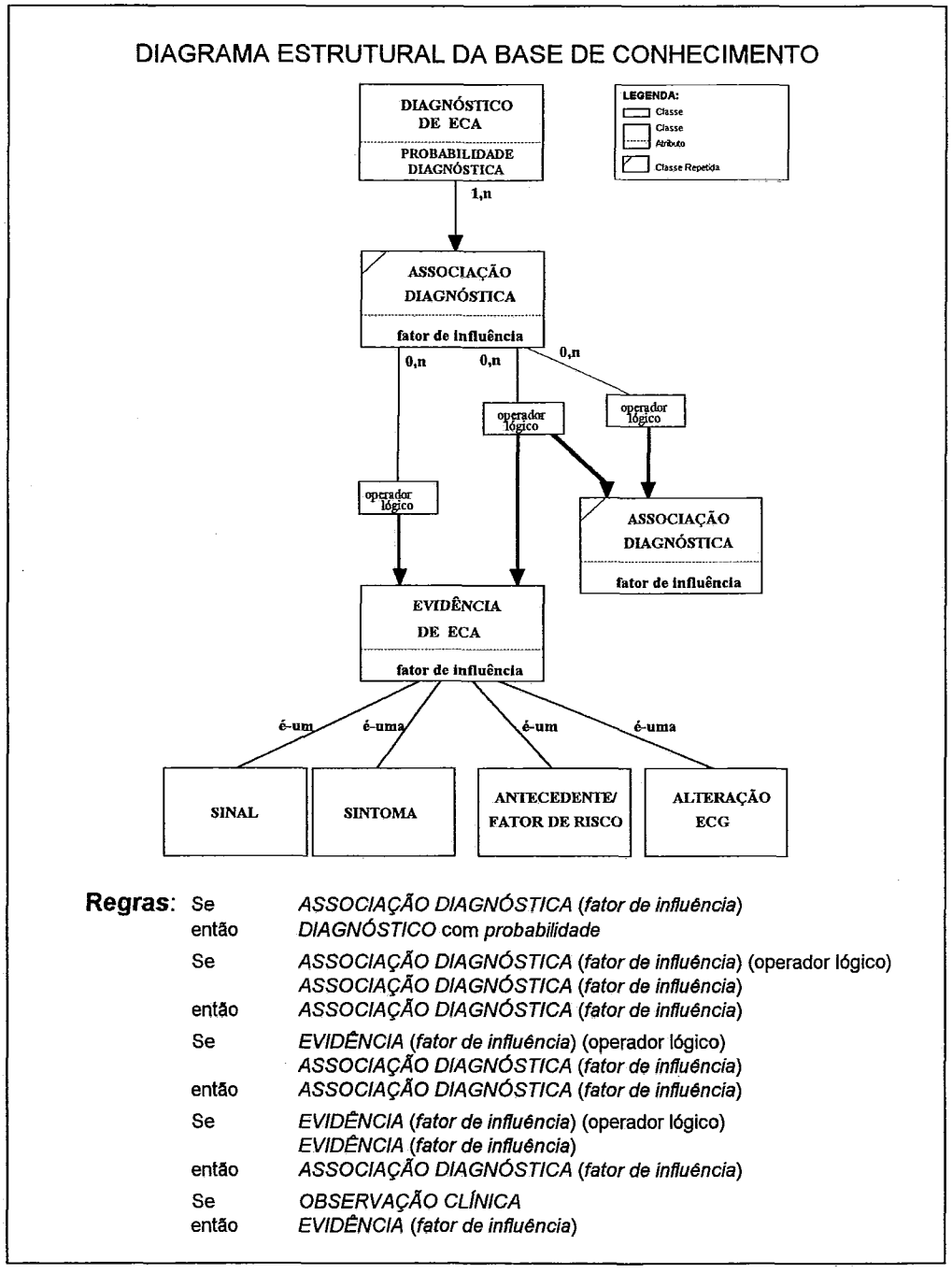


Figura V.17: Modelo de Implementação do Usuário do SEC versão 2.0

A Figura V.18 apresenta o *Diagrama Estrutural da Base de Conhecimento* do SEC Versão 2.0 que utiliza uma linguagem orientada a objetos e regras.



Regras:

Se	*ASSOCIAÇÃO DIAGNÓSTICA (fator de influência)*
então	*DIAGNÓSTICO* com *probabilidade*
Se	*ASSOCIAÇÃO DIAGNÓSTICA (fator de influência)* (operador lógico) *ASSOCIAÇÃO DIAGNÓSTICA (fator de influência)*
então	*ASSOCIAÇÃO DIAGNÓSTICA (fator de influência)*
Se	*EVIDÊNCIA (fator de influência)* (operador lógico) *ASSOCIAÇÃO DIAGNÓSTICA (fator de influência)*
então	*ASSOCIAÇÃO DIAGNÓSTICA (fator de influência)*
Se	*EVIDÊNCIA (fator de influência)* (operador lógico) *EVIDÊNCIA (fator de influência)*
então	*ASSOCIAÇÃO DIAGNÓSTICA (fator de influência)*
Se	*OBSERVAÇÃO CLÍNICA*
então	*EVIDÊNCIA (fator de influência)*

Figura V.18: Diagrama Estrutural da Base de Conhecimento do SEC versão 2.0

O conteúdo da base de conhecimento (***Especificação da Base de Conhecimento***) é definido a partir da estrutura do *Diagrama Estrutural da Base de Conhecimento*. A Figura V.19 apresenta, como exemplo, parte da *Especificação da Base de Conhecimento* do sistema SEC versão 2.0.

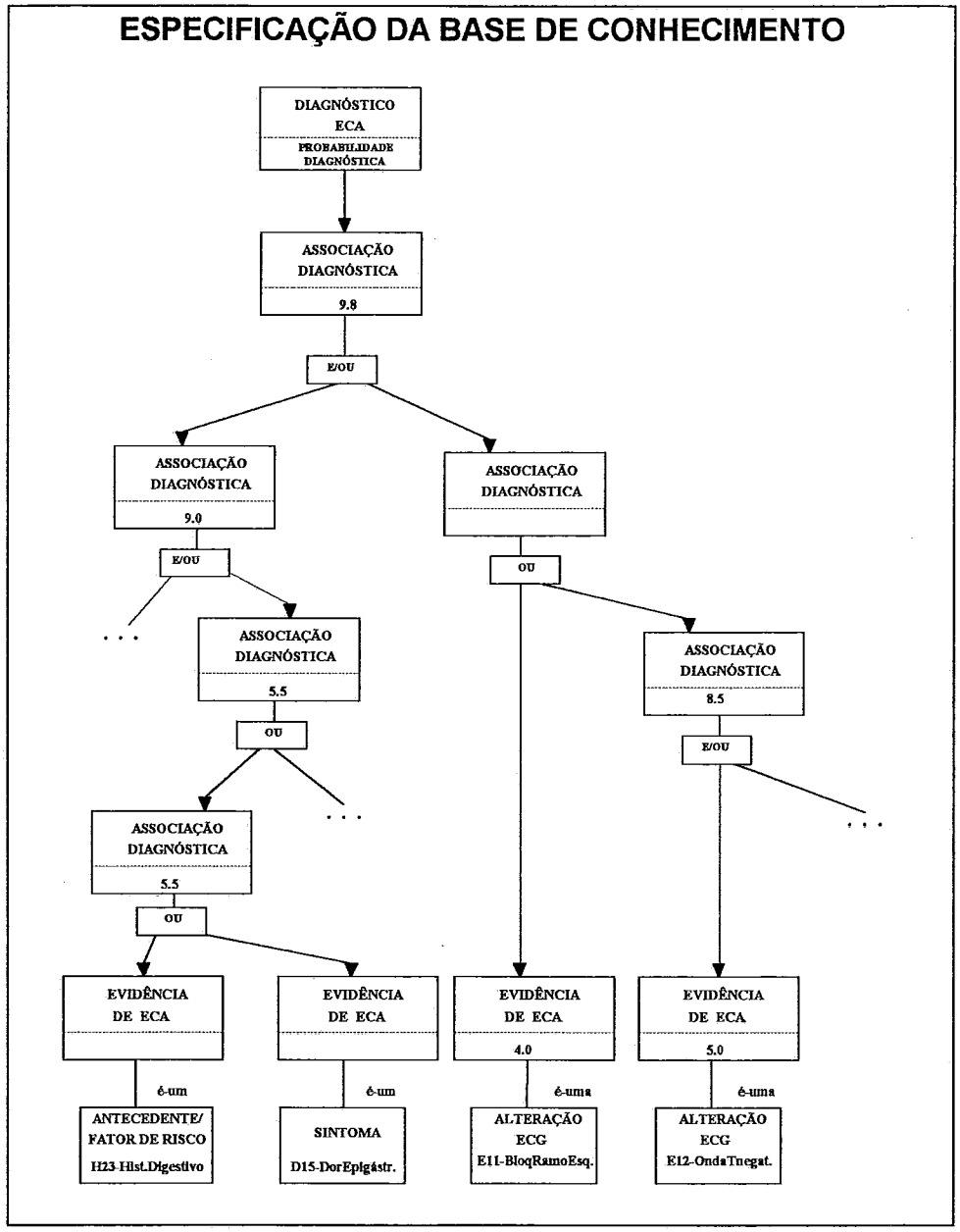


Figura V.19: Especificação da Base de Conhecimento do SEC versão 2.0

A ***Especificação da Memória de Trabalho*** implica na definição do conteúdo das possíveis entradas do sistema que serão armazenadas em arquivos ou banco de dados, dependendo das facilidades oferecidas pelo ambiente de programação utilizado para construção do sistema. O armazenamento desses dados de entrada é importante para posterior consulta ou nova execução, sendo também um mecanismo de avaliação do sistema, pois a partir dos resultados gerados podem ser avaliadas as respostas fornecidas pelo sistema. Em algumas situações, como por exemplo no caso do sistema SEC, este é, também, um mecanismo de segurança, pois permite ter-se acesso a informações sobre uma execução.

A *Especificação da Memória de Trabalho* do sistema é construída a partir do *Diagrama do Domínio do Problema*, com base em conceitos de definição lógica e física de banco de dados. O *Diagrama do Domínio do Problema* mostra os dados que são informados ao sistema sobre cada entidade ou objeto e seus relacionamentos. A Figura V.20 contém um exemplo de *Especificação da Memória de Trabalho* para o SEC, considerando a utilização de um banco de dados relacional.

ESPECIFICAÇÃO DA MEMÓRIA DE TRABALHO

PACIENTE (código, nome, idade, sexo)

MÉDICO (código, nome)

CONSULTA (código-médico,código-paciente,número-consulta, data, diagnóstico, probabilidade)

OBS-CLÍNICAS (número-consulta, tipo, tamanho, valor)

O layout de OBS-CLÍNICAS é variável e cada tipo terá os seguintes atributos:

- **ECG (descrição-alteração)**
- **DOR (tipo, localização, intensidade, duração)**
- **SINTOMA-ASSOCIADO (descrição)**
- **PULSO (frequência-cardíaca, tipo, filiforme, diferença-amplitude)**
- **PRESSÃO-ARTERIAL (máximo, mínimo)**
- **AUSCULTA (ritmo-galopante, créptos-base, atrito-pericárdico)**
- **SINAIS-OBSERVAÇÃO (descrição)**
- **FATOR-RISCO (descrição)**
- **HISTÓRICO-PARTICULAR (descrição)**
- **HISTÓRICO-FAMILIAR (descrição)**

Figura V.20: Especificação da Memória de Trabalho do SEC

A ***Especificação dos Módulos*** define cada módulo do sistema numa representação próxima da linguagem de programação a ser utilizada na etapa de construção.

Em linguagens orientada a objetos e regras, a *Especificação dos Módulos* é o *Modelo de Objetos*. Este modelo é composto pelas classes, atributos e relações do *Diagrama do Domínio do Problema* e das classes do *Diagrama Estrutural da Base de Conhecimento*. As estruturas de regras definidas no *Diagrama Estrutural da Base de Conhecimento* serão as regras genéricas da classe. A Figura V.21 mostra um exemplo para o SEC.

ESPECIFICAÇÃO DE MÓDULOS

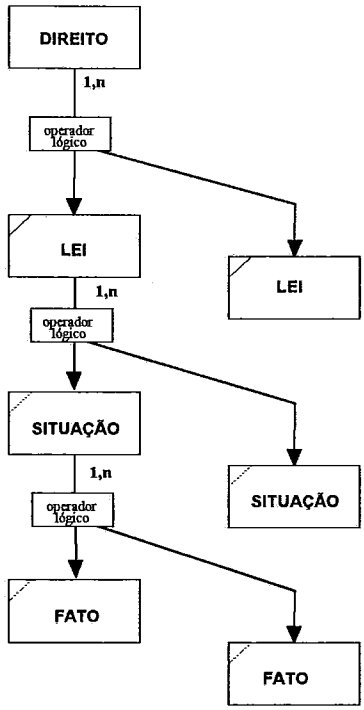
MODELO DE OBJETOS	DEFINIÇÃO DAS REGRAS
• Classes, atributos e relações do *Diagrama do Domínio do Problema*: e do *Diagrama Heurístico do Raciocínio* **Exemplo de Classes:** *PACIENTE, ECG, SINTOMA, SINAL, DIAGNÓSTICO DE ECA, ASSOCIAÇÃO DIAGNÓSTICA, EVIDÊNCIA DE ECA, OBSERVAÇÃO CLÍNICA* **Exemplo de Atributos da Classe *ASSOCIAÇÃO DIAGNÓSTICA*:** *fator influência e operador lógico*	Se *ASSOCIAÇÃO DIAGNÓSTICA (fator de influência)* então *DIAGNÓSTICO com probabilidade* Se *ASSOCIAÇÃO DIAGNÓSTICA (fator de influência)* (operador lógico) *ASSOCIAÇÃO DIAGNÓSTICA (fator de influência)* então *ASSOCIAÇÃO DIAGNÓSTICA (fator de influência)* Se *EVIDÊNCIA (fator de influência)* (operador lógico) *ASSOCIAÇÃO DIAGNÓSTICA (fator de influência)* então *ASSOCIAÇÃO DIAGNÓSTICA (fator de influência)* Se *EVIDÊNCIA (fator de influência)* (operador lógico) *EVIDÊNCIA (fator de influência)* então *ASSOCIAÇÃO DIAGNÓSTICA (fator de influência)* Se *OBSERVAÇÃO CLÍNICA* então *EVIDÊNCIA (fator de influência)*

Figura V.21: Especificação de Módulos do sistema SEC Versão 2.0

Em linguagens lógicas, como PROLOG, as classes *Diagrama Estrutural da Base de Conhecimento* são definidas como predicados e os seus relacionamentos com operadores lógicos entre as classes, como regras. Essas regras são definidas na estrutura de regras apresentada no *Diagrama Estrutural da Base de Conhecimento*. Na Figura V.22 encontra-se um exemplo do *Diagrama Estrutural da Base de Conhecimento* com a estrutura genérica das regras e a *Especificação dos Módulos*, definidos para o sistema CRIANDO que utiliza PROLOG como linguagem de programação.

A forma de *Especificação dos Módulos* é totalmente dependente do ambiente de programação utilizado na implementação do sistema. Assim sendo, pode ser necessária a definição de alguma função especial, tais como tratamento de incerteza, cálculos ou métodos utilizados por uma classe de objetos em linguagens orientadas a objetos. Em linguagens orientadas a objetos com regras, pode ser necessária a criação de métodos utilizados por objetos do sistema. Estes métodos deverão ser definidos na *Especificação dos Módulos*. Em linguagens lógicas, pode ser necessária a criação de predicados da máquina de inferência. Normalmente, ao se utilizar uma *shell*, esta necessidade é reduzida e a *Especificação dos Módulos* passa a ser uma correlação entre os predicados disponíveis na *shell* e os utilizados no sistema (Werneck, 1994b).

DIAGRAMA ESTRUTURAL DA BASE DE CONHECIMENTO



DEFINIÇÃO DAS REGRAS

Regras:

Se	LEI <operador lógico> LEI
então	DIREITO
Se	SITUAÇÃO<operador lógico> SITUAÇÃO
então	LEI
Se	FATO <operador lógico> FATO
então	SITUAÇÃO

ESPECIFICAÇÃO DE MÓDULOS

Predicados:

direito( ):- lei( ) <operador lógico> lei( )

lei( ) :- situaçao( ) <op.lógico> situacao( )

situaçao( ) :- fato( ) <op.lógico> fato( )

Figura V.22: Diagrama Estrutural da Base de Conhecimento e Especificação de Módulos do sistema CRIANDO

V.7. Conclusão

Neste capítulo foi definida uma extensão do método KADS, o *KADS-estendido*.

Esta proposta surgiu a partir da experiência adquirida no desenvolvimento do sistema SEC e de sistemas desenvolvidos em trabalhos acadêmicos nas áreas de Engenharia de Software e Informática na Educação da COPPE-Sistemas.

O *KADS-estendido* foi validado na Versão 2.0 do sistema SEC. A proposta do *KADS-estendido* também foi validada, para o sistema CRIANDO (Werneck, 1994b), DESPEJA (Passos, 1994) e KBRF (Oliveira et alli, 1995).

Capítulo VI

ORIXÁS[1]: Um Ambiente para Desenvolvimento de Sistemas Baseados em Conhecimento

VI.1. Introdução

O principal objetivo desta tese é definir um ambiente de desenvolvimento de software adequado aos sistemas baseados em conhecimento.

Ambiente de desenvolvimento de software (ADS) pode ser definido como um sistema de computação que provê o suporte para o desenvolvimento, reparo e melhorias em software e para o gerenciamento e controle dessas atividades. Esse sistema contém uma base central de dados que atua como um repositório para todas as informações relacionadas com o projeto ao longo do seu ciclo de vida, e um conjunto de ferramentas de software que oferecem suporte para as várias atividades técnicas e gerenciais desempenhadas no projeto (Moura, 1992).

Os ambientes de desenvolvimento de software variam quanto à natureza geral de suas bases de dados e o âmbito coberto por suas ferramentas. As propostas mais recentes enfatizam o apoio a todo o ciclo de vida, ao gerenciamento do projeto e ao controle da qualidade do produto.

---

[1] ORIXÁS são os deuses na cultura bahiana e abrem os caminhos do mar, do céu, da terra,... Este nome foi escolhido, em homenagem à Bahia, aos baianos e seus ORIXÁS, pela acolhida recebida por este trabalho e também, por analogia. Um Ambiente de Desenvolvimento é a infra-estrutura para o desenvolvimento de software, atuando como um guia que orienta os caminhos para construção do software.

A definição do processo de desenvolvimento deve preceder à automação da engenharia de software (Arthur, 1993). Assim, propor um ambiente de desenvolvimento de software consiste na definição do processo, dos métodos e das ferramentas para o desenvolvimento, controle da qualidade e gerência dos produtos de software de cada etapa do ciclo de vida do sistema.

Nos Capítulos IV e V foram definidos um processo de desenvolvimento para sistemas baseados em conhecimento e o método KADS-estendido. Em Oliveira (1995) foram definidos os procedimentos para verificação e validação de sistemas baseados em conhecimento. Tem-se, ainda, uma experiência concreta de uso do processo, do método KADS-estendido e dos procedimentos de avaliação da qualidade no projeto SEC da Fundação Bahiana de Cardiologia.

Neste capítulo, definimos e decrevemos o primeiro protótipo de um ADS capaz de apoiar o desenvolvimento de sistemas baseados em conhecimento. Este ambiente está inserido no contexto do Projeto TABA (Rocha, 1987), (Rocha, 1990), (Aguiar, 1992) e (Travassos, 1994), cujo objetivo é a construção de uma Estação de Trabalho, configurável, para desenvolvimento de software. A Estação TABA é configurável para atender às particularidades de diferentes domínios de aplicação, tecnologias de desenvolvimento ou mesmo projetos específicos.

VI.2. A Estação TABA

O projeto TABA visa construir uma Estação de Trabalho para o engenheiro de software, que permita a implementação de ambientes adequados ao desenvolvimento de software em diferentes domínios de aplicação, possibilitando também sua execução.

Este objetivo é atendido através das quatro funções da estação TABA (Rocha, 1987), (Rocha, 1990), (Aguiar, 1992) e (Travassos, 1994):

I) auxiliar o engenheiro de software na especificação e instanciação do ambiente mais adequado ao desenvolvimento de um produto específico;

II) auxiliar o engenheiro de software a implementar as ferramentas necessárias ao ambiente definido em (I);

III) permitir aos desenvolvedores do produto (software) o uso da estação através do ambiente definido em (I) e gerado em (II);

IV) permitir a execução do software na própria estação por ele configurada (o que é sempre verdade pelo menos na fase de testes).

Essas funções da Estação TABA determinam quatro ambientes que são:

- **Ambiente Especificador e Instanciador de ADSs**

É o meta-ambiente TABA, cuja função é especificar o ADS mais adequado para o desenvolvimento de um software específico e instanciar o ADS.

O instanciador de ADS seleciona os componentes já disponíveis na Estação que foram indicados para compor a instância de ADS e torna o ambiente operacional. Além disso cabe ao instanciador agregar ao ADS instanciado um assistente baseado em conhecimento, para auxiliar no seu uso.

A construção do especificador de ambientes XAMÃ (Aguiar, 1992), (Massolar, 1993) foi realizada e está operacional em ambiente Sun. Em Travassos (1994) foi concluída a definição e construção do meta-ambiente TABA através da proposta do mecanismo para integração de ferramentas. Este mecanismo possibilitou a instanciação dos ambientes especificados por XAMÃ.

- **Ambiente para Construção de Ferramentas**

O Ambiente para Construção de Ferramentas auxilia o engenheiro de software na construção de novas ferramentas para Estação TABA e sua incorporação ao meta-ambiente. Assim, através da construção de novas ferramentas pode-se ter a instanciação completa de ADS especificado por XAMÃ.

O mecanismo proposto por Travassos (1994) para integração de ferramentas possibilita a construção de novas ferramentas na Estação e, também, a integração de ferramentas externas, isto é, desenvolvidas fora da Estação.

- **Ambiente de Desenvolvimento**

O Ambiente de Desenvolvimento é o ADS que foi especificado e instanciado através do meta-ambiente.

- **Ambiente de Execução**

O Ambiente de Execução é o local onde o software poderá ser executado.

VI.2.1. Estrutura da Estação TABA2.1. Estrutura da Estação TABA

Na Estação TABA foram identificados os seguintes requisitos gerais (Travassos, 1994): ser configurável, possuir uma interface consistente, possuir um mecanismo de integração, apoiar na construção de novas ferramentas, possuir conhecimento sobre o processo e métodos de desenvolvimento, oferecer assistência inteligente ao usuário, possuir um modelo de armazenamento de dados comum e possuir suporte à reutilização.

Esses requisitos visam fornecer à Estação TABA uma estrutura que facilite a definição de novos ambientes de desenvolvimento e ferramentas, através dos ADSs instanciados na Estação TABA que possuem os seguintes requisitos específicos (Travassos, 1994):

- **possuir suporte para o controle e gerenciamento de versões**: É responsabilidade do ambiente controlar e gerenciar as modificações realizadas nos itens de software construídos ao longo do processo de desenvolvimento, mantendo os documentos gerados disponíveis para os usuários em suas diferentes versões;

- **possuir suporte para o gerenciamento de todas as atividades ao longo do processo de desenvolvimento**: A Estação TABA deve controlar e gerenciar o processo de desenvolvimento através da verificação do andamento do trabalho, controle da execução de atividades relacionadas e encadeadas. Este controle de gerenciamento não deve permitir que a ordem de execução das tarefas seja trocada, ou mesmo modificada, sem que haja uma intervenção do Engenheiro de

Software. Em Sardinha (1993) foi realizado um trabalho visando definir e construir um ambiente para gerência de projetos;

- **possuir suporte para a medição do produto**: O ADS armazena, ao longo do processo de desenvolvimento, informações relevantes que permitem fazer medidas sobre os produtos do processo de desenvolvimento. Estas métricas devem estar disponíveis para projetos futuros, possibilitando aperfeiçoamentos no processo de desenvolvimento e, consequentemente, melhorias na qualidade dos produtos. O trabalho de definição de métricas para o gerenciamento da qualidade do processo de desenvolvimento do software está sendo desenvolvido focalizando a qualidade do produto, com métricas que suportem a avaliação da qualidade e desempenho do processo (Martins e Rocha, 1994);

- **apoiar o trabalho cooperativo:** O desenvolvimento de um produto de software, principalmente quando se trata de desenvolvimento em larga escala, se dá através do trabalho de equipes. A coordenação e interação das pessoas envolvidas são necessárias para que as atividades possam ser realizadas corretamente. Ao longo do processo de desenvolvimento, a comunicação entre a equipe ocorre frequentemente, obrigando o ambiente a definir protocolos de comunicação que possibilitem este relacionamento;

- **possuir interfaces customizáveis**: O usuário do ambiente é responsável por determinar como será a forma de apresentação da interface, sendo preciso que se forneça a ele meios para que o ajuste da interface atenda às suas preferências pessoais;

- **possuir suporte para a avaliação do produto**: A Estação TABA deve oferecer suporte para a medição e avaliação da qualidade dos produtos nela desenvolvidos. A definição de atributos de qualidade para diferentes domínios de aplicação bem como a construção de ferramentas para avaliação da qualidade estão sendo desenvolvidas num conjunto de trabalhos (Passos e Rocha, 1994), (Belchior e Rocha, 1994), (Clunie e alii, 1994), (Campos, 1994), (Oliveira et alii,

1994a), (Commerlato, 1994), (Commerlato at alii, 1994), (Rabelo at alii, 1995), (Oliveira, 1995).

A estrutura adotada para a Estação TABA é mostrada na Figura VI.1 e baseia-se na idéia de que um ADS está inserido no contexto de um produto de software, devendo representar o mundo real para o mundo computacional, ou seja, facilitar a construção de produtos de software para o usuário do ADS a partir dos requisitos estabelecidos.

Para que sejam atingidos os requisitos da Estação TABA, devem estar presentes os seguintes componentes: sistema computacional, interface com o usuário, cooperação, controle de processos, suporte inteligente, reutilização, conhecimento, repositório comum do ADS e controle de versões. A descrição detalhada desses componentes pode ser encontrada em Travassos (1994).

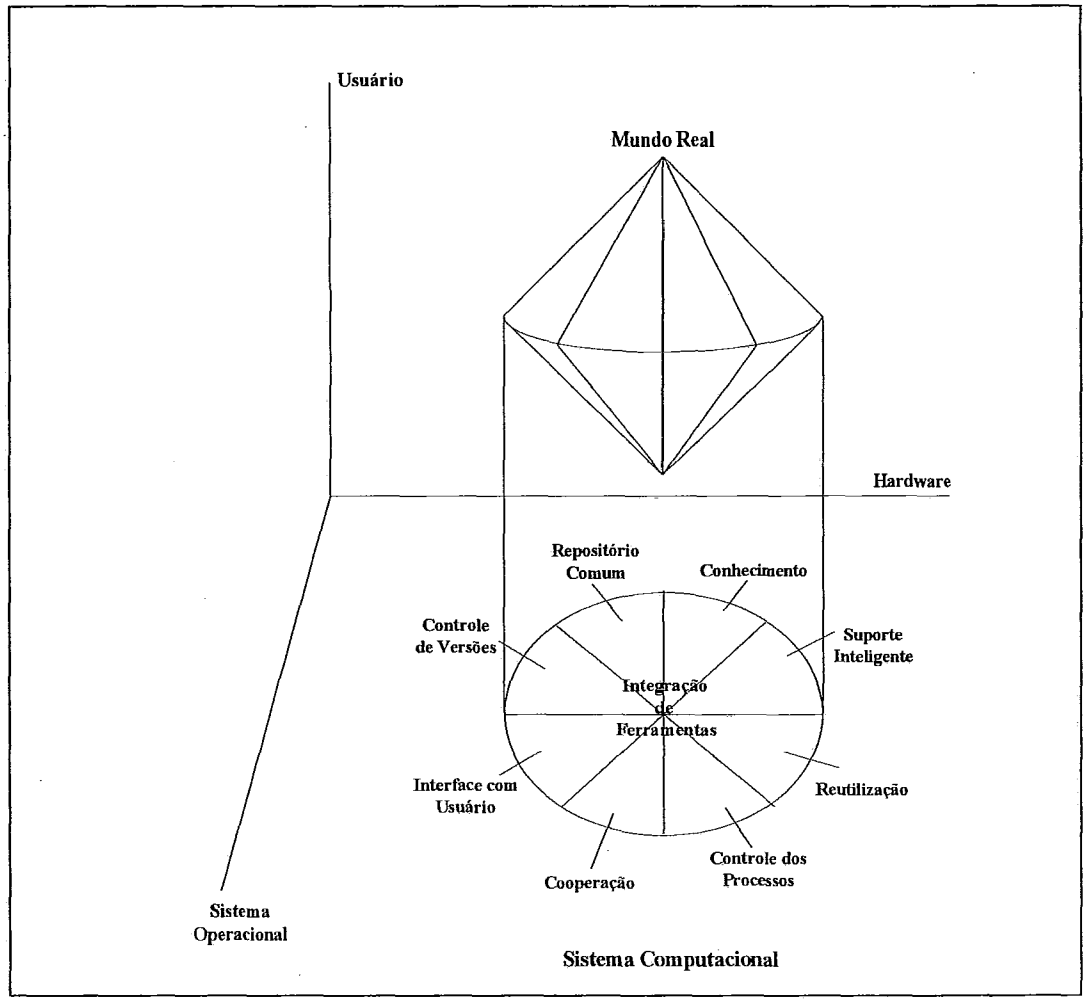


Figura VI.1 - Estrutura de um ADS na estação TABA (Travassos 1994)

VI.2.2. O Modelo de Integração TABA

O modelo de integração TABA considerou os requisitos definidos, identificando a modelagem relativa ao Meta Modelo e aos Ambientes Instanciados TABA. A modelagem do Meta Modelo é específica ao meta ambiente e suas funcionalidades. O modelo dos Ambientes Instanciados TABA se refere à utilização da Estação TABA, onde ambientes são instanciados e tornam-se passíveis de utilização.

O modelo de integração da Estação TABA foi definido em Travassos (1994) e aperfeiçoado por Travassos e Werner (1995). No Anexo III, encontra-se a representação desta modelagem, utilizando-se o método de Análise Orientada a Objetos proposto por Coad e Yourdon (1992). Aqui será fornecida apenas uma visão geral deste modelo para permitir o entendimento da modelagem do protótipo do Ambiente ORIXÁS (seção VI.4).

VI.2.2.1 Meta Modelo TABA

O diagrama de assuntos do Meta-Ambiente (Figura VI.2) descreve os seguintes assuntos representados pelas classes que os compõem e a funcionalidade da Estação (Travassos, 1994):

- ***Meta Ambiente:*** representa a essência do modelo com apenas a classe *TABA*, cujo objetivo é a coordenação das tarefas do meta ambiente, disponibilizando, quando necessário, conhecimento, padrões de interface com o usuário e estruturas para construção de ADSs e ferramentas. A instância da classe TABA possui um conjunto de serviços associados: definir ambiente, instanciar ambiente, testar ambiente, descrever ambiente, sugerir ferramenta, construir ferramentas internas e acrescentar ferramentas externas;

- ***Apresentação e Interação***: reúne os elementos responsáveis pela comunicação do usuário com a Estação na classe *Interface com o Usuário,* que representa todos os objetos de interação;

- ***Conhecimento***: reúne os objetos que fornecem conhecimento sobre métodos, processo de desenvolvimento e domínios de aplicação para a Estação, ADS e

ferramentas. Este assunto é composto de duas classes: *Conhecimento* e *Assistente Especialista*. A classe *Conhecimento* permite que as ferramentas obtenham informações referentes aos elementos utilizados por cada método, de forma a realizar um mapeamento do mundo de objetos para o mundo do usuário. A instanciação dessa classe é realizada pela representação e modelagem do conhecimento de cada método na Linguagem para Descrição e Manipulação de Métodos (LDMM)[1]. A classe *Assistente Especialista* é responsável por assistir o usuário na utilização do ADS e de suas ferramentas;

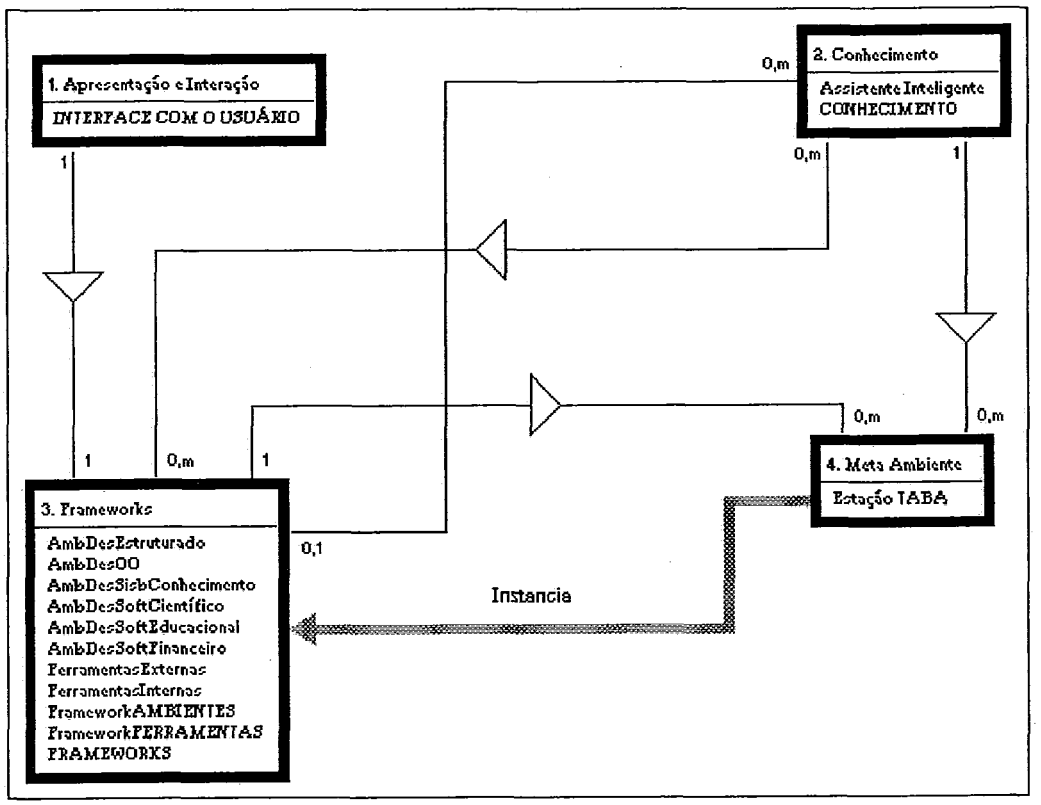


Figura VI.2- Diagrama de Assuntos do Meta Ambiente TABA (Travassos, 1994)

- ***Frameworks:*** representam as estruturas básicas para construção de ambientes e ferramentas utilizados na Estação. Este assunto é composto das classes *FrameworkAmbientes* e *FrameworkFerramentas*. Um framework é projetado

[1] No anexo IV, encontra-se uma descrição da linguagem LDMM.

para ser refinado ou especializado e é constituido de um conjunto de classes abstratas[2] e classes concretas[3]. Existem duas classes abstratas, a primeira está relacionada aos ambientes de desenvolvimento representando o conjunto de informações e funcionalidade de um ADS (*FrameworkAmbientes*). A outra classe, *FrameworkFerramentas*, diz respeito às ferramentas existentes e que possivelmente estão integradas num ADS.

VI.2.2.2 Modelo dos Ambientes Instanciados TABA

O meta ambiente TABA é o responsável pela definição e instanciação de ambientes, os quais são compostos por um processo, atividades, métodos e ferramentas internas e/ou externas. Esses componentes necessitam trabalhar de forma integrada. Esta integração é obtida conforme mostra o diagrama de assuntos apresentado na figura VI.3 e o modelo completo dos ambientes instanciados na Estação TABA, que pode ser encontrado no Anexo III. Este diagrama descreve a seguinte situação, onde as classes encontradas no modelo estão escritas em *itálico* (Travassos, 1994), (Travassos e Rocha, 1994) (Travassos e Werner, 1995):

> Um *ADS* existe para o desenvolvimento de algum *projeto*. Este *projeto*, por sua vez, pode ser desenvolvido por *equipes*, que são compostas por *pessoas*, que desempenham *papéis* nas diversas *atividades* ao longo do processo de desenvolvimento. Cada *atividade*, ao longo do processo, gera um conjunto de *documentos* e *utiliza recursos*. Estes documentos, que representam informações específicas do processo, são compostos por *grafos* manuseados pelas *ferramentas*, construídas para terem *conhecimento* sobre um determinado método de desenvolvimento. A forma de comunicação destas ferramentas com o usuário é determinada pela *interface com o usuário*, que é comum a todas as *ferramentas*, e ao próprio *ADS*.
>
> O usuário pode solicitar, ao longo do processo de desenvolvimento, alguma

[2] Classes abstratas são classes de alto nível que, a princípio, não permitem a instanciação de objetos.

[3] Classes concretas são classes passíveis de instanciação.

assistência ao *ADS*, que possui um *assistente inteligente* específico com conhecimento sobre o *domínio da aplicação*, *os métodos*, as *atividades* e o *processo* considerados pelo ADS.

A ordem de execução das *ferramentas* está relacionada à ordem de acontecimento das *atividades*, ao longo do processo de desenvolvimento. Cabe ao ADS certificar a ocorrência e término das *atividades*, e efetivar a manutenção das informações necessárias à realização de ensaios e medidas, a respeito da execução da atividade, que permitam um planejamento futuro mais adequado.

A seguir, serão descritas as principais classes existentes no modelo de *Ambientes Instanciados TABA*:

- A classe ***Grafos***, representa a estrutura de representação da informação do modelo de integração de ferramentas da Estação TABA. Este tipo de organização da informação é baseado num *modelo de grafos*, onde o conhecimento necessário para a descrição e representação de cada elemento pertencente a um determinado método é descrito a partir da Linguagem de Descrição e Manipulação de Métodos (Anexo IV). A estrutura de grafos é representada através da utilização do paradigma de orientação a objetos, permitindo que o comportamento desta estrutura reflita as modificações sofridas ao longo do processo de desenvolvimento de um produto de software. Um grafo é composto por *elementos* que podem ser de dois tipos: *nós* e *ligações*. *Nós* representam os vértices do grafo e podem assumir os seguintes papéis, ou funções, representados por subclasses de nós: *descritivo, transformador, fonte, destino, estado, armazenador, encapsulador. organizador* e *código fonte*. *Ligações* representam as arestas de um grafo e podem assumir os seguintes papéis, ou funções, representados por subclasses de ligações: *hierarquia, composição, estímulo, dependência* e *transportador*.
- A classe ***Equipe*** descreve as características das equipes envolvidas no desenvolvimento de algum projeto. Estas equipes são compostas por várias

pessoas, com o objetivo de cumprir o melhor possível as estimativas de prazo, custo e qualidade atribuídas para um determinado projeto.

- A classe ***Pessoas*** representa os integrantes das equipes de desenvolvimento que participarão de algum projeto. De maneira geral, pessoas desempenham atividades ao longo do processo de desenvolvimento, podendo assumir diferentes *papeis*, como por exemplo, funções de coordenação e gerência.

- A classe ***Projeto*** representa o produto que está sendo desenvolvido. Um projeto é desenvolvido por equipes de desenvolvimento, apoiadas por um Ambiente de Desenvolvimento de Software integrado. O processo de desenvolvimento ocorre com a execução de um conjunto de atividades, planejadas de acordo com o domínio da aplicação, o paradigma de desenvolvimento e o ciclo de vida adotados.

- A classe ***Atividades*** representa as atividades a serem realizadas ao longo do processo de desenvolvimento. Estas atividades são, normalmente, realizadas por pessoas, integrantes de alguma equipe de desenvolvimento, que utilizam recursos, podendo ter, como resultado, um conjunto de documentos associados. O acompanhamento da execução destas atividades, por parte do ADS, permite obter dados concretos sobre o tempo, esforço e custo, o que possibilita sua comparação com os valores estimados e, consequentemente, um melhor planejamento, no futuro, de processos de desenvolvimento de software.

- A classe ***Documentos*** descreve as características dos documentos que devem ser gerados nas *atividades* do processo de desenvolvimento. Estes documentos são compostos por *grafos*, que por sua vez são construídos e manuseados pelas *ferramentas*, num determinado formato. A partir deste formato, o engenheiro de software pode definir a forma de composição dos documentos, garantindo a utilização de padrões pré-estabelecidos e a qualidade desejada.

- A classe ***ADS*** representa um ambiente que tenha sido definido, instanciado e testado pelo meta-ambiente TABA. A descrição desta classe é semelhante à classe *FrameworkAmbientes*.

- A classe ***Ferramentas*** representa as ferramentas disponíveis na Estação TABA e que se encontram integradas ao ADS. A descrição desta classe é semelhante à classe *FrameworkFerramentas*.

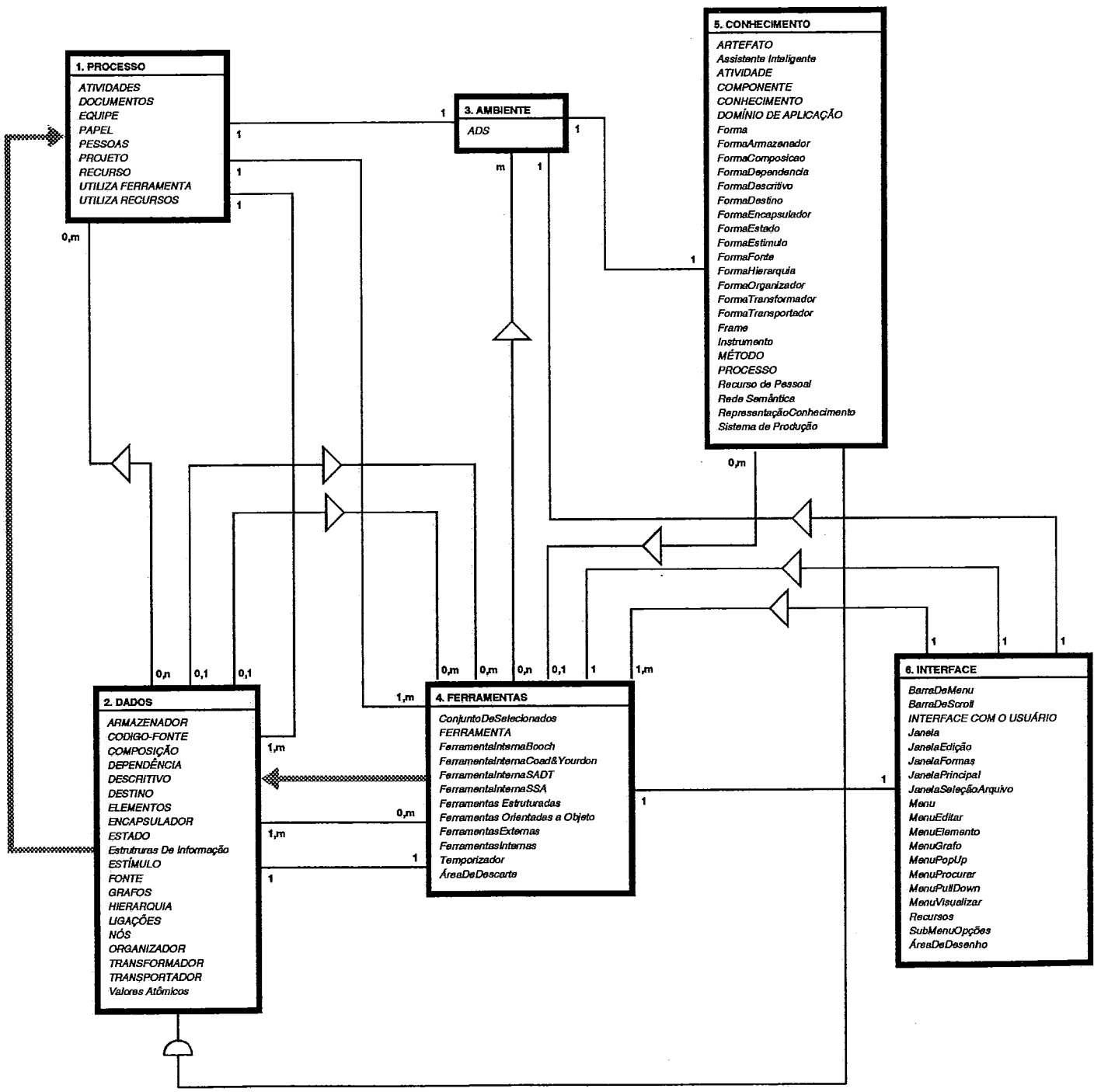


Figura VI.3 - Diagrama de Assuntos dos Ambientes Instanciados TABA (Travassos e Werner, 1995)

- A classe ***Conhecimento***[1] é composta de classes com a capacidade de representar o conhecimento referente a *métodos, processo de desenvolvimento* e *domínios de aplicação.*

- A classe ***Assistente Inteligente*** está associada a objetos responsáveis por dar suporte ao usuário na utilização do ADS e de suas ferramentas, utilizando o conhecimento sobre *métodos, processo de desenvolvimento* e *domínios de aplicação.*

- A classe ***Interface com o Usuário*** descreve as características essenciais dos objetos responsáveis pela interação do usuário com o ambiente e do usuário com as ferramentas integradas ao ADS.

- A classe ***Estruturas de Informação*** contém informações conduzidas pelos *Transportadores.* Estas estruturas descrevem agregados de informações do produto de software que está sendo desenvolvido a partir da utilização do ADS. Na composição de estruturas de informação são utilizados *valores atômicos*, que representam as informações elementares no contexto do produto.

VI.3. ORIXÁS: um Ambiente para Desenvolvimento de Sistemas Baseados em Conhecimento

O ambiente ORIXÁS está fundamentado no processo de desenvolvimento para sistemas baseados em conhecimento (Capítulo IV), no método KADS-estendido (Capítulo V) e nos procedimentos para avaliação da qualidade estabelecidos em Oliveira (1995).

Este ambiente é definido no âmbito do projeto TABA, sendo o primeiro ambiente a utilizar o modelo de integração da Estação TABA (Travassos, 1994) (Travassos e Rocha, 1994), (Travassos e Werner, 1995). O Ambiente ORIXÁS é, portanto, um

[1]As classes relacionadas ao assunto *Conhecimento* estão sendo modeladas em trabalho que se encontra em desenvolvimento (Falbo e Travassos, 1995).

ambiente instanciado TABA e, como tal, deve possuir os requisitos definidos para esses ambientes.

A seguir são descritos o objetivo do ambiente e uma visão geral do mesmo, indicando-se as características de seus usuários e suas ferramentas. É definido, também, o protótipo deste ambiente na estrutura da Estação TABA.

VI.3.1 Objetivo

O objetivo do ambiente ORIXÁS é apoiar o desenvolvimento de sistemas baseados em conhecimento, de acordo com um processo de desenvolvimento bem definido e através do uso sistemático de ferramentas que possibilitam a aplicação do método KADS-estendido e de procedimentos gerenciais e de controle da qualidade adequados.

VI.3.2 Descrição do Ambiente

ORIXÁS, o ambiente para desenvolvimento de sistemas baseados em conhecimento da Estação TABA, é um sistema computacional que apoia o desenvolvimento de sistemas baseados em conhecimento, através do gerenciamento de todas as atividades definidas no processo de desenvolvimento descrito no Capítulo IV. Este ambiente fornece apoio à utilização do método KADS-estendido, através de ferramentas de construção e apoio à avaliação da qualidade.

A Tabela VI.1 fornece uma visão geral do processo de desenvolvimento com suas fases, seus produtos finais e os métodos que foram definidos para serem utilizados nas fases deste processo.

O processo de desenvolvimento proposto no Capítulo IV define, também, as atividades de cada fase, a organização da equipe de desenvolvimento, o controle da qualidade e a gerência. As Tabelas VI.2 a VI.8 apresentam, de forma esquematizada, as atividades dessas fases, seus produtos e a equipe de trabalho. O ambiente deverá controlar e gerenciar este processo de desenvolvimento através da verificação do andamento do trabalho e do controle da execução dessas atividades.

FASES	PRODUTO	MÉTODOS	
		KADS-estendido	Outros
ANÁLISE DO DOMÍNIO DO PROBLEMA	Especificação do Domínio do Problema	Modelo do Domínio do Problema	Método de Elicitação do Conhecimento
PLANEJAMENTO DO PROJETO	Plano do Projeto	-	-
ANÁLISE DO CONHECIMENTO	Especificação de Requisitos	Modelo Especialidade	Método de Elicitação do Conhecimento
PROJETO DA VERSÃO	Especificação de Projeto	Modelo Lógico Modelo Físico	-
CONSTRUÇÃO DA VERSÃO	Versão Construída	-	-
AVALIAÇÃO DO PRODUTO	Versão Aceita	-	Método Rocha
OPERAÇÃO DA VERSÃO	Versão em Uso	-	-
AVALIAÇÃO DO PROCESSO	Processo Avaliado	-	-

Tabela VI.1 - Processo de Desenvolvimento para SBC segundo suas Fases, seus Produtos e Métodos

ANÁLISE DO DOMÍNIO DO PROBLEMA

Produto Final: **Especificação do Domínio do Problema (EDP)**
Sub-Produtos: Gravações (Vídeo/"Tape"), **Relatório Histórico do Projeto**

ATIVIDADES	MÉTODOS	EQUIPE
• **Pesquisa de SBC no Domínio do Problema**	-	EC
• **Aquisição do Conhecimento Geral**	-	EC, ESP, CT
♦ **Elicitação do Conhecimento Geral**	**Método Elicitação do Conhecimento**	EC, ESP
♦ **Documentação do Conhecimento Geral**	**KADS-estendido (Modelo do Domínio Problema)**	EC
♦ **Avaliação do Conhecimento Geral**	**Método Rocha**	EC,ESP,CT
• **Identificação das Áreas de Conhecimento**	-	EC, ESP
• **Descrição do Sistema Proposto**	-	EC, GER
• **Avaliação da EDP**	**Método Rocha**	QUAL, CP

Legenda: EC - engenheiros do conhecimento; ESP - especialistas / consultor especialista; CT - coordenador técnico; GER - equipe de gerência; QUAL - equipe de qualidade; CP - coordenador do projeto

Tabela VI.2 - Atividades, Métodos e Equipe da Análise de Domínio do Problema

PLANEJAMENTO DO PROJETO

Produto Final: **Plano do Projeto (PP)**
Sub-Produto: **Relatório Histórico do Projeto**

ATIVIDADES	EQUIPE
• **Planejamento do Desenvolvimento**	CT, CP
• **Planejamento do Estágio**	EC, CT
• **Avaliação do Plano do Projeto**	CP

Legenda: CT - coordenador técnico
EC - engenheiros do conhecimento
CP - coordenador do projeto

Tabela VI.3 - Atividades e Equipe do Planejamento do Projeto

ANÁLISE DO CONHECIMENTO

Produto Final: **Especificação dos Requisitos (ERS)**
Sub-Produtos: Gravações (Vídeo/"Tape"), **Relatório Histórico do Projeto**

ATIVIDADES	MÉTODOS	EQUIPE
• **Análise do Conhecimento Específico**	**Método de**	EC,ESP,CT
♦ **Elicitação do Conhecimento Específico**	**Elicitação do**	EC, ESP
♦ **Documentação do Conhecimento**	**Conhecimento**	EC
♦ **Validação dos Requisitos**	**Método Rocha**	EC, ESP, CT
• **Especificação do Conhecimento Específico**	**KADS-estendido**	EC, ESP, CT
♦ **Modelagem do Conhecimento**	**(Modelo de**	EC, ESP
♦ **Documentação da Modelagem**	**Especialidade)**	EC
♦ **Validação da Modelagem**	**Método Rocha**	EC, ESP, CT
• **Avaliação da Especificação de Requisitos**	**Método Rocha**	QUAL, CP

Legenda: EC - engenheiros do conhecimento
CT - coordenador técnico
CP - cootdenador do projeto
ESP - especialistas / consultor especialista
QUAL - equipe de qualidade

Tabela VI.4 - Atividades, Métodos e Equipe da Análise do Conhecimento

PROJETO DA VERSÃO

Produto Final: **Especificação de Projeto (EP)**
Sub-Produto: **Relatório Histórico do Projeto**

ATIVIDADES	MÉTODOS	EQUIPE
• **Definição da Representação e Inferência do Conhecimento**	-	EC, CT
• **Modelagem Lógica**	**KADS-estendido**	EC,ESP, CT, EBC
♦ **Construção do Diagrama Heurístico do Raciocínio (DHR)**	**(Modelo**	EC, ESP
♦ **Construção do Diagrama do Domínio do Problema (DDP)**	**Lógico)**	EC
♦ **Validação do Modelo Lógico**	**Método Rocha**	EC,CT,EBC
• **Modelagem Física**	**KADS-estendido**	EC,ESP, US, CT, EIA
♦ **Construção do Modelo de Implementação do Usuário**		EC,ESP, US
⇒ **Elaboração do Diagrama de Interface com Usuário**	**(Modelo**	EC,ESP, US
⇒ **Elaboração do Diagrama de Explicação do Raciocínio**		EC, ESP, US
♦ **Construção do Modelo de Implementação do Sistema**		EC,ESP
⇒ **Construção do Diagrama Estrutural da Base de Conhecimento**	**Físico)**	EC
⇒ **Especificação da Base de Conhecimento**		EC, ESP
⇒ **Especificação da Memória de Trabalho**		EC
⇒ **Especificação dos Módulos**		EC
♦ **Validação do Modelo Físico**	**Método Rocha**	EC,CT,EIA
• **Avaliação da Especificação de Projeto**	**Método Rocha**	QUAL, CP

Legenda: EC - engenheiros do conhecimento; CT - coordenador técnico; ESP - especialistas / consultor especialista; EBC - especialista em SBC; US - usuário; EIA - especialista em programação de IA; QUAL - equipe de qualidade; CP - cootdenador do projeto

Tabela VI.5 - Atividades, Métodos e Equipe do Projeto da Versão

CONSTRUÇÃO DA VERSÃO

Produto Final: **Versão Construída**
Sub-Produto: **Relatório Histórico do Projeto**

ATIVIDADES	PRODUTOS INTERMEDIÁRIOS	EQUIPE
• **Codificação de Módulos**	Módulo Construído	PGM
• **Avaliação de Módulos**	Módulo Avaliado	EC, PGM
• **Integração de Módulos**	Módulos Integrados	EC, PGM
• **Teste de Integração**	Versão Integrada	EC

Legenda: EC - engenheiros do conhecimento PGM - programador

Tabela VI.6 - Atividades, Produtos Intermediários e Equipe da fase de Construção da Versão

AVALIAÇÃO DO PRODUTO

Produto Final: **Versão Aceita**
Sub-Produto: **Relatório Histórico do Projeto**

ATIVIDADES	PRODUTOS INTERMEDIÁRIOS	EQUIPE
• **Testar a Nova Versão**	Versão ok	EC, ESP
• **Planejar a Operação**	Manual do Usuário e de Operação, Plano Implantação	EC, CT
• **Aceitar a Nova Versão**	Versão Aceita	CP

Legenda: EC - engenheiros do conhecimento ESP - especialistas / consultor especialista
CT - coordenador técnico CP - coordenador do projeto

Tabela VI.7 - Atividades, Produtos Intermediários e Equipe da fase de Avaliação do Produto

OPERAÇÃO DA VERSÃO

Produto Final: **Versão em Uso**
Sub-Produto: **Relatório Histórico do Projeto**

ATIVIDADES	PRODUTOS	EQUIPE
• **Treinar o Usuário**	Versão em Treino	EC, US
• **Implantar a Versão**	Versão Implantada	EC, PGM
• **Usar a Versão**	Novos Requisitos	US, ESP, EC

Legenda: EC - engenheiros do conhecimento US - usuário
PGM - programador ESP - especialistas / consultor especialista

Tabela VI.8 - Atividades e Equipe da Fase de Operação da Versão

A Tabela VI.9 apresenta os componentes dos métodos de construção e avaliação da qualidade utilizados no processo de desenvolvimento do ORIXÁS. Para gerência do projeto estão, ainda, sendo definidos os métodos específicos a serem utilizados. Existe na literatura poucos trabalhos relativos a aspectos gerenciais no desenvolvimento de sistemas baseados em conhecimento. Assim sendo, devem ser realizados trabalhos de pesquisa antes de se poder definir os métodos de gerência que serão utilizados em ORIXÁS. Neste momento, está prevista a elaboração de dois documentos: o Plano do Projeto e o Relatório Histórico do Projeto, que têm como objetivo realizar, respectivamente, o planejamento e o acompanhamento do projeto.

ORIXÁS é um ambiente que atende aos requisitos dos ambiente instanciados TABA através da modelagem definida para estes. Assim sendo, além do apoio ao controle e gerenciamento e do conhecimento sobre o método KADS-estendido, o ambiente integra um conjunto de ferramentas que facilitam a geração dos produtos previstos no processo de desenvolvimento, bem como a avaliação da qualidade destes produtos e a gerência do processo de desenvolvimento. A seguir são descritas as ferramentas que compõem o ambiente ORIXÁS.

VI.3.3 Descrição das Ferramentas do Ambiente ORIXÁS

As atividades fundamentais do processo de desenvolvimento de um software estão relacionadas às atividades de construção, de avaliação da qualidade do produto e de gerência do processo de desenvolvimento. No capítulo IV foi definido um processo de desenvolvimento para sistemas baseados em conhecimento, considerando esses três aspectos. Por isso, as ferramentas que compõe ORIXÁS serão descritas considerando-se três grupos: ferramentas para construção do produto, ferramentas para avaliação da qualidade e ferramentas para gerência.

MÉTODOS	COMPONENTES
MÉTODO DE ELICITAÇÃO DO CONHECIMENTO	**Técnicas de Elicitação do Conhecimento** **Técnica Delphi Modificada**
MÉTODO KADS-ESTENDIDO	
• **MODELO DO DOMÍNIO DO PROBLEMA**	**Diagrama de Tarefas** **Taxonomia Geral do Domínio Problema**
• **MODELO DE ESPECIALIDADE**	**Estrutura do Domínio** **Estrutura de Inferência** **Estrutura de Tarefas** **Diagrama de Transição de Estados**
• **MODELO LÓGICO**	**Diagrama Heurístico do Raciocínio** **Diagrama do Domínio Problema**
• **MODELO FÍSICO**	
♦ **MODELO DE IMPLEMENTAÇÃO DO USUÁRIO**	**Diagrama de Interface com o Usuário** **Diagrama de Explicação do Raciocínio**
♦ **MODELO DE IMPLEMENTAÇÃO DO SISTEMA**	**Diagrama Estrutural Base de Conhecimento** **Especificação da Base de Conhecimento** **Especificação da Memória de Trabalho** **Especificação dos Módulos**
MÉTODO ROCHA	**Objetivos, Atributos e Critérios de Qualidade** **Processos de Avaliação**

Tabela VI.9 - Componentes dos Métodos do Ambiente ORIXÁS

VI.3.3.1 Ferramentas para Construção do Produto

O objetivo principal das ferramentas de construção é auxiliar no processo de produção de um software, ao longo do seu ciclo de vida. Assim sendo, as ferramentas de construção propostas apoiarão desde a fase de elicitação do conhecimento passando pela análise do conhecimento, projeto e construção até o refinamento da versão do sistema. Isto pressupõe que estas ferramentas fornecem suporte à produção de documentos e de programas e à gerência das diferentes versões dos produtos que serão produzidos ao longo do processo de desenvolvimento.

➢ **Ferramenta de Apoio à Elicitação do Conhecimento**

A Elicitação do Conhecimento é realizada utilizando-se as técnicas específicas para apoio a esta atividade (Capítulo IV). Sempre que possível, as sessões de elicitação do conhecimento devem ser registradas através de uma câmara de vídeo. A experiência no desenvolvimento do sistema SEC nos mostrou que o uso de vídeo auxilia na aquisição do conhecimento no sentido de facilitar a assimilação e compreensão do assunto tratado, pois permite aos engenheiros do conhecimento tirar dúvidas ao longo da documentação e modelagem do conhecimento. No SEC serviu, também, para treinamento de novos engenheiros do conhecimento que foram incorporados ao projeto na 2ª versão.

No caso de haver impossibilidade do uso de vídeo, a elicitação do conhecimento deverá ser gravada. Registrar somente o som, entretanto, não é o mais recomendável, pois pode haver uma perda de conhecimento. Os especialistas normalmente gesticulam, fornecendo informações adicionais à compreensão do seu conhecimento.

ORIXÁS deverá possuir uma ferramenta para registro das imagens e sons das sessões de elicitação do conhecimento. Esta ferramenta permitirá, também, que sejam agregados comentários sobre partes da fita, bem como ligações entre as imagens e/ou sons às informações registradas nos documentos, principalmente na Especificação do Domínio do Problema e na Especificação de Requisitos.

Durante a elicitação do conhecimento deve-se utilizar, também, a técnica de Delphi modificada, que consiste na identificação de diferentes pontos de vista e integração

desses numa reunião, quando não houver consenso. Nesse caso, a mesma ferramenta que auxiliará na documentação da atividade deverá ser usada para apoiar a reunião.

Em trabalho anterior foi especificada uma ferramenta, HIPER-AVAL (Werneck, 1992), cujo objetivo é apoiar a documentação e o encaminhamento de uma reunião para discussão de refinamentos a serem realizados em protótipos. Esta ferramenta auxilia na elicitação dos requisitos, registrando os diferentes pontos de vista e as decisões tomadas nas reuniões. Essas informações servem para revisão do protótipo e como base de consulta para o implementador das alterações.

No ambiente ORIXÁS, HIPER-AVAL pode ser utilizada como uma ferramenta para complementar a documentação dos diferentes pontos de vista e também para conduzir a reunião de tomada de decisão, mostrando as diferentes visões e os seus argumentos.

No futuro está previsto o desenvolvimento de uma única ferramenta para auxiliar a elicitação do conhecimento, englobando o registro de imagens e sons, a funcionalidade de HIPER-AVAL e as ligações entre essas imagens e sons com outros documentos ou informações do ambiente. Pode-se, ainda, desenvolver um assistente inteligente para auxílio no encaminhamento da tomada de decisão sobre os diferentes pontos de vista. Esta ferramenta geral de elicitação do conhecimento apoiará, assim, a integração das informações.

➢ **Ferramenta de Apoio ao método KADS-estendido**

Esta ferramenta deverá apoiar o desenvolvimento dos modelos propostos na extensão do método KADS (Capítulo V). Assim sendo, deverá ser composta de vários módulos, cada um correspondendo a um modelo ou componente do método. A Figura VI.4 apresenta a estrutura geral desta ferramenta de apoio à construção de especificação e projeto utilizando-se o KADS-estendido. A seguir será descrito cada um dos módulos da ferramenta.

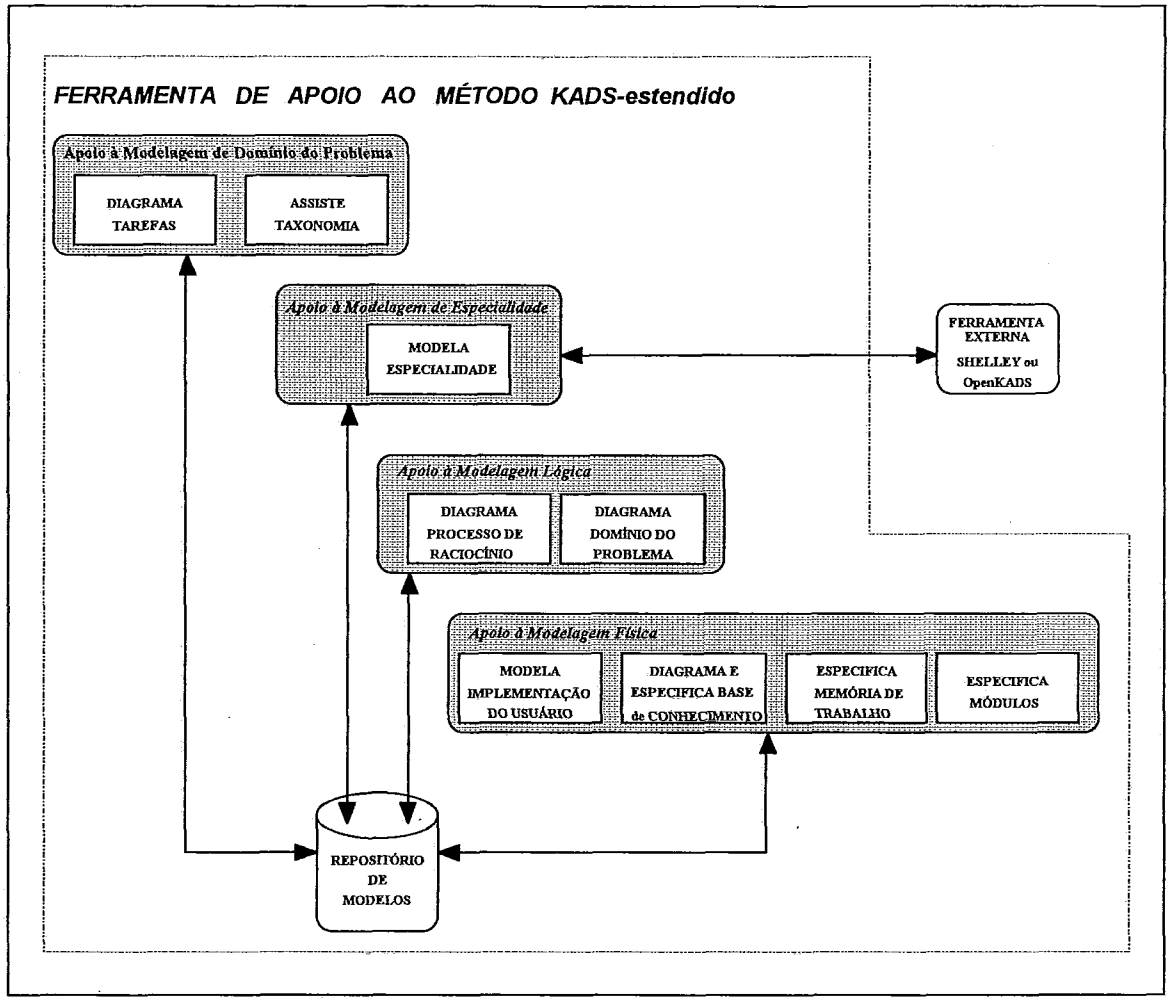


Figura VI.4 - Ferramenta de Apoio ao Método KADS-estendido

♦ **Módulo *Diagrama Tarefas***

Este módulo é responsável pela construção do Diagrama de Tarefas do Modelo de Domínio do Problema proposto na extensão do método KADS. Apoiará o engenheiro do conhecimento na decomposição do domínio do problema em tarefas, através de uma representação gráfica que define o fluxo das tarefas. Além do diagrama, esta ferramenta permitirá, também, uma descrição suscinta das tarefas. Esta especificação é opcional, embora seja recomendada, pois complementa a definição do conhecimento sobre as tarefas.

O Diagrama de Tarefas poderá ser integrado à ferramenta de apoio à elicitação do conhecimento para evidenciar uma conexão lógica entre as sessões de elicitação do

conhecimento e cada tarefa definida no diagrama. Assim, se poderá ter uma ligação efetiva entre a elicitação e a documentação do conhecimento adquirido.

No caso da elicitação do conhecimento ser realizada sem o apoio de especialistas, a fonte deste conhecimento deverá ser registrada na especificação de tarefas servindo, futuramente, para validar o sistema.

♦ **Módulo *Assiste Taxonomia***

Este módulo tem como objetivo apoiar na elaboração da taxonomia do domínio do problema que é outro componente do Modelo de Domínio do Problema proposto no método KADS-estendido. A taxonomia é gerada através da classificação do conhecimento por diferentes aspectos, aplicando a técnica de ordenação conceitual proposta para relacionar os diferentes aspectos de acordo com a classificação por facetas (Prieto-Diaz, 1987).

No início da elaboração da taxonomia, o engenheiro do conhecimento e/ou um coordenador dos especialistas deve definir uma lista dos conceitos envolvidos no domínio do problema e também uma lista dos assuntos gerais. Essas listas são o ponto de partida para a ordenação conceitual e criação da taxonomia. Os especialistas poderão modificar as listas, gerando suas próprias facetas de classificação do problema. que depois serão consolidadas através da técnica Delphi modificada definida para o SEC (seção IV.4.9), para resolução dos diferentes pontos de vista.

Este módulo deverá apoiar o engenheiro do conhecimento e os especialistas na produção de esquemas, grafos e tabelas que expressem a taxonomia. Deverá, também, permitir que a elaboração das diferentes taxonomias sejam realizadas de forma anônima para não inibir o fator criativo e crítico dos especialistas.

♦ **Módulo *Modela Especialidade***

Este módulo tem como objetivo construir o Modelo de Especialidade do KADS-estendido.

As modificações realizadas neste modelo foram pequenas em relação ao proposto originalmente no KADS e considerando que a Estação TABA permite a incorporação de ferramentas externas, é viável o uso de ferramentas já disponíveis no mercado. Sendo assim poderá ser utilizada uma ferramenta externa já desenvolvida anteriormente, como por exemplo, SHELLEY (Anjewierden 1993) ou OpenKADS[1] .

Desta forma, somente, é necessário incorporar ao modelo as alterações propostas e construir o Diagrama de Transição de Estados. Essas funções podem estar em um único módulo que, através do modelo gerado pela ferramenta externa, disponibilizará o mesmo para a fase de projeto lógico, permitindo a criação do Diagrama de Transição de Estados.

Caso não seja possível a utilização dessas ferramentas externas, um único módulo deverá ser desenvolvido, englobando funções que apoiem na construção dos quatro componentes do modelo.

- **Módulo *Diagrama Processo de Raciocínio***

Este módulo tem o propósito de modelar o processo de raciocínio do sistema através da construção do Diagrama Heurístico do Raciocínio. A elaboração desse diagrama deverá ser semi-automatizada, utilizando os passos de construção definidos no Capítulo V. Alguns desses passos devem ter a interação dos engenheiros do conhecimento, podendo gerar uma nova atividade de elicitação do conhecimento.

- **Módulo *Diagrama Domínio do Problema***

Este módulo tem o propósito de elaborar o Diagrama do Domínio do Problema, que é uma nova visão da Estrutura de Domínio do Modelo de Especialidade. A elaboração desse diagrama deverá ser semi-automatizada, utilizando os passos de construção definidos no Capítulo V, onde as classes presentes no Diagrama Heurístico do Raciocínio serão explicitamente relacionadas a esta estrutura.

---

[1] Esta ferramenta está sendo comercializada pela ABC Bull, devendo ser cedida às Universidades em geral.

♦ **Módulo *Modela Implementação do Usuário***

Este módulo apoia na geração do Modelo de Implementação do Usuário que contém o Diagrama de Interface com o Usuário e o Diagrama de Explicação do Raciocínio.

A partir do Diagrama do Domínio do Problema, este módulo gerará automaticamente uma primeira versão do Diagrama de Interface com o Usuário, com a definição das telas e seus encadeamentos. Isto é possível, pois no Diagrama do Domínio do Problema são definidas as classes que fazem parte do início e fim do sistema, bem como as classes de entrada introduzidas ao longo do sistema. O Diagrama Heurístico do Raciocínio contém, explicitamente, a ordem de execução dessas entradas, podendo ser esta ordem consultada para geração automática do Diagrama de Interface com o Usuário.

Após a geração da primeira versão devem ser realizados refinamentos no diagrama com informações do tipo de tela, comandos, encadeamentos, etc... O refinamento do Diagrama de Interface com o Usuário pode ser elaborado com auxílio de prototipagem, pois as ferramentas de programação permitem a simulação do encadeamento de telas e visualização dos diálogos. Assim sendo, os refinamentos podem ser feitos utilizando-se a própria ferramenta de programação, para possibilitar a simulação rápida do diálogo entre o sistema e o usuário.

O Diagrama de Explicação do Raciocínio deve ser definido pelo desenvolvedor ao ter concluído o Diagrama de Interface com o Usuário, com definição dos diferentes tipos de explicação e o conteúdo de cada tela. Este módulo apoiará o engenheiro do conhecimento na elaboração deste diagrama.

♦ **Módulo *Diagrama e Especifica Base de Conhecimento***

Este módulo apoia-se no Diagrama Heurístico do Raciocínio para gerar de forma semi-automática o Diagrama Estrutural da Base de Conhecimento. Este é obtido através das classes heurísticas e seus relacionamentos que estão contidos no Diagrama Heurístico do Raciocínio. A estrutura das regras é gerada através dos relacionamentos com expressão entre essas classes e seus atributos.

Outro produto deste módulo é a Especificação da Base de Conhecimento. Esta é gerada pelos especialistas ou pelo próprio engenheiro do conhecimento através da edição do conhecimento no Diagrama Estrutural da Base de Conhecimento.

♦ **Módulo *Especifica Memória de Trabalho***

Este módulo tem como objetivo gerar Especificação da Memória de Trabalho do sistema, o que é realizado através da consulta e visualização do Diagrama do Domínio do Problema. Os dados armazenados na memória de trabalho serão definidos na estrutura de armazenamento a ser utilizada pelo sistema. Este módulo deverá permitir a transformação semi-automática desses dados nas principais estruturas de armazenamento (banco de dados relacional, banco de dados orientados a objetos, arquivos sequenciais, arquivos VSAM, etc...).

♦ **Módulo *Especifica Módulos***

Este módulo é totalmente dependente do paradigma da linguagem de programação, devendo ser definido um módulo para cada paradigma. No ambiente ORIXÁS deverão ser desenvolvidos, inicialmente, os seguintes módulos: Especifica Módulos em Prolog e Especifica Módulos em Orientação a Objetos e Regras.

➢ **Ferramentas de Apoio à Programação**

No ambiente ORIXÁS deverá se dar preferência às ferramentas de construção do tipo *shell* ou a ambientes de programação que facilitem a implementação, pois acarretam numa maior produtividade no desenvolvimento dos programas. Assim, ferramentas já comercializadas deverão ser incorporadas ao ambiente ORIXÁS através do mecanismo de integração de ferramentas externas da Estação TABA. Essas ferramentas, normalmente, já tem embutidos mecanismos de depuração, auxílio à geração de interfaces com o usuário, máquina de inferência, mecanismo de controle, tratamento de incertezas, etc...

➢ **Ferramenta de Apoio à Documentação**

A documentação do produto, segundo o processo de desenvolvimento definido neste trabalho, é composta pelos diferentes modelos previstos no KADS-estendido e por

textos em linguagem natural, considerando o conjunto de roteiros para elaboração de documentos que devem fazer parte do Plano de Documentação. Uma sugestão de roteiros para estes documentos pode ser encontrada em Werneck et alii (1994d).

Na Estação TABA, a instanciação de ambientes inclui a definição dos roteiros de documentos a serem gerados. A ferramenta de edição prevista, deverá ser uma nova versão de PRODOC (Vale, 1988) que permite o planejamento de documentos, ajuda no preenchimento dos diversos itens e na confecção dos documentos de acordo com os padrões previamente estabelecidos.

➢ **Ferramenta de Apoio ao Refinamento de Versões**

Esta ferramenta deverá apoiar a atividade de refinamento do sistema através do controle das versões, explicitando as modificações realizadas no código, na base de conhecimento (através da geração de uma nova versão da Especificação da Base de Conhecimento) e as decisões tomadas que geraram as alterações.

Esta ferramenta é uma grande integradora de informações, podendo acessar a ferramenta de apoio à elicitação do conhecimento, a ferramenta de apoio ao método KADS-estendido (módulos de apoio a modelagem física) e a ferramenta de apoio à programação. Deverão ser adicionados a esses produtos, informações sobre os refinamentos e suas decisões.

VI.3.3.2 Ferramentas para Avaliação da Qualidade do Produto

As ferramentas de avaliação da qualidade apoiam desde o planejamento da qualidade até a verificação e validação do sistema baseado em conhecimento. Muitas ferramentas de programação têm embutidas ferramentas que auxiliam na depuração desses sistemas. Por isso, será pressuposto que o apoio à depuração está incorporado ao ambiente de programação utilizado na construção do sistema. Um conjunto de ferramentas para avaliação da qualidade de software está sendo desenvolvido na COPPE, através de projeto integrado apoiado pelo CNPq. Estas ferramentas estão fundamentadas em vários trabalhos anteriores realizados pela área de Engenharia de Software. ORIXÁS fará uso destas ferramentas.

➢ **Assistente Baseado no Conhecimento para Apoio ao Planejamento da Qualidade**

Esta ferramenta tem como objetivo auxiliar o engenheiro de software na definição dos requisitos de qualidade de projetos específicos e na elaboração do plano de qualidade, que irá guiar os procedimentos para a avaliação dos produtos gerados ao longo do desenvolvimento.

Este assistente está sendo desenvolvido por Passos (Passos e Rocha, 1994), utilizando o conhecimento adquirido em diversas pesquisas que buscaram identificar a importância dos diferentes atributos de qualidade de software para domínios específicos, tais como, software educacional (Campos, 1994), científico (Palermo e Rocha, 1989), (Rocha, 1992), (Passos, 1992), financeiro (Belchior, 1992). As diferentes tecnologias que podem ser utilizadas na construção do produto também influenciam a definição da qualidade. Para responder a esta realidade já foram realizados vários trabalhos onde se pesquisou atributos de qualidade relacionados a sistemas especialistas (Oliveira, 1995), a aplicações multimídia (Campos, 1994) e software orientado a objetos (Clunie et alii, 1994). Este conhecimento será incorporado à ferramenta.

O engenheiro de software, utilizando o ambiente ORIXÁS, fará uso deste assistente para definir os requisitos de qualidade de um projeto e estabelecer o Plano de Qualidade.

➢ **Meta-Avaliador de Especificações**

Outro trabalho, também em andamento na COPPE, tem como objetivo construir um meta-avaliador de especificações. Esta ferramenta, um assistente baseado em conhecimento, apoiará o engenheiro de software na avaliação de especificações (especificação de requisitos e de projeto) produzidas utilizando-se diferentes métodos de desenvolvimento (Belchior e Rocha, 1994). Este meta-avaliador poderá, portanto, ser configurado para diversos métodos, entre eles o KADS-estendido.

O ambiente ORIXÁS, utilizará esta ferramenta para apoio na avaliação de especificações de requisitos e de projeto, produzidas com KADS-estendido.

➢ **Verificador da Base de Conhecimento**

Esta ferramenta tem o objetivo de verificar a base de conhecimento em termos de consistência, redundância e completude através da procura de regras redundantes, conflitantes, circulares e desnecessárias.

O verificador da base de conhecimento é dependente da estrutura utilizada para representação do conhecimento e terá suporte à geração de tabelas de decisão e grafos, considerando as ferramentas para verificação de sistemas especialistas descritas em Oliveira (1995).

➢ **Ferramenta para Geração de Casos de Teste**

Esta ferramenta para geração de casos de teste deverá fornecer apoio à análise de sensibilidade (Capítulo IV), permitindo a criação interativa de novos casos de teste através da variação de variáveis de entrada.

➢ **Ferramenta para Avaliação Quantitativa da Qualidade**

O objetivo desta ferramenta é auxiliar na análise quantitativa dos resultados de teste, permitindo uma avaliação do sistema quanto a sua acurácia, grau de confiabilidade, especificidade e sensibilidade. Esta análise foi feita para o SEC e mostrou-se de grande utilidade (Oliveira, 1995).

A ferramenta orientará o usuário nos testes estatísticos mais pertinentes à análise quantitativa que este deseja realizar. Pacotes estatísticos, já disponíveis no mercado (por exemplo SPSS), poderão ser integrados a esta ferramenta de forma a fornecer uma maior disponibilidade de funções. Também deverá fornecer facilidades para elaboração de gráficos ou estar integrada a outros pacotes gráficos.

➢ **Ferramenta para Apoio à Inspeção por Fases**

No ambiente ORIXÁS, o controle da qualidade do produto é realizado através da técnica de inspeção por fases (Knight, 1993) através da qual os produtos gerados ao longo do processo de desenvolvimento são submetidos a uma série de avaliações parciais. Estas avaliações podem ser avaliações individuais ou reuniões de inspeção.

Esta ferramenta controla a execução do Plano de Qualidade, elaborado com auxílio do Assistente para Apoio ao Planejamento da Qualidade. Interage, também, com o Meta-Avaliador de Especificações. São funções desta ferramenta:

- controlar a execução do Plano de Qualidade;
- interagir com o Meta-Avaliador de Especificações e guardar os resultados das avaliações realizadas;
- apoiar na avaliação de código e guardar os resultados das avaliações realizadas;
- apoiar na realização de reuniões de inspeção.

VI.3.3.3 Ferramentas de Gerência

As ferramentas de gerência apoiam o engenheiro de software no planejamento do projeto (estimativa de custos, elaboração de cronograma, identificação dos recursos necessários, análise dos riscos) e no acompanhamento gerencial do projeto. Para isto ORIXÁS terá as seguintes ferramentas:

➢ **Assistente Baseado no Conhecimento para Apoio ao Planejamento do Projeto**

Este assistente baseado em conhecimento tem como objetivo apoiar a gerência técnica na elaboração do Plano do Projeto. O Plano do Projeto é, na realidade, um conjunto de planos: Plano do Processo de Desenvolvimento, Plano de Organização da Equipe, Plano de Documentação, Plano de Controle da Qualidade, Plano de Recursos e Produtos, Plano dos Estágios. Um exemplo de roteiros para estes planos pode ser encontrado em Werneck (1994d) e FBC (1994).

Esta ferramenta interage com várias outras. O Plano do Processo de Desenvolvimento é realizado através de XAMÃ, que auxilia na definição do ciclo de vida, métodos e ferramentas a serem utilizados no desenvolvimento do projeto. No caso de sistemas baseados em conhecimento, este processo poderá ser o próprio ORIXÁS ou especializações deste.

O Plano de Controle da Qualidade é feito através do Assistente Baseado em Conhecimento para Apoio ao Planejamento da Qualidade, descrito anteriormente.

O Plano de Documentação é feito através da ferramenta de apoio à documentação, também descrita anteriormente.

A análise de riscos será feita através de um Assistente Baseado em Conhecimento que está sendo desenvolvido na COPPE para apoiar esta atividade, considerando diferentes tipos de projetos, entre eles os sistemas baseados em conhecimento.

A base para construção desta ferramenta serão os trabalhos realizados anteriormente por Ribeiro (1989), Menezes (1989) e Sardinha (1993). Os procedimentos para planejamento estabelecidos nestes trabalhos ainda necessitam, entretanto, ser especializados para atender às características do desenvolvimento de sistemas baseados em conhecimento.

➢ **Ferramenta para Apoio ao Acompanhamento do Projeto**

O acompanhamento do projeto é fundamental principalmente, num paradigma de desenvolvimento evolutivo e dinâmico como o ciclo de vida do ambiente ORIXÁS. Esta ferramenta tem como referência básica o Plano do Projeto e, a partir deste, apoia o gerente do projeto no acompanhamento do projeto em todas as suas fases e atividades. Subsídios para esta ferramenta estão sendo definidos por Martins (1994).

No modelo de ambientes instanciado TABA já foi previsto um acompanhamento do processo em termos de atividades, recursos e documentos, sendo definida uma infra-estrutura de classes contida no asssunto Processo, conforme pode ser visto na modelagem descrita no Anexo III. ou no Diagrama de assuntos da Figura VI.3.

VI.3.4 Usuários do Ambiente

Os usuários deste ambiente deverão ser os engenheiros do conhecimento, o

coordenador técnico[1] e o coordenador do projeto. Os especialistas também serão usuários, pois poderão utilizar as ferramentas relacionadas à elicitação, especificação e validação do conhecimento.

Os engenheiros do conhecimento serão os maiores usuários do ambiente pois são responsáveis pelas atividades apoiadas por diversas ferramentas. Eles apoiam os especialistas no uso das ferramentas de apoio à Elicitação do Conhecimento, à elaboração da Taxonomia do Conhecimento e na Especificação da Base de Conhecimento. Utilizam, ainda, as ferramentas de apoio ao método KADS-estendido, à programação, à documentação, ao refinamento das versões, além das ferramentas de avaliação da qualidade do produto.

O coordenador técnico e coordenador do projeto são usuários das ferramentas de gerência. O coordenador técnico deverá ter acesso, também, às ferramentas de construção e avaliação da qualidade, para gerenciar o processo de desenvolvimento.

VI.4. Especificação do ADS ORIXÁS através de XAMÃ

XAMÃ na versão 1.0 implementou a função *planejamento de ADS* da Estação TABA, cujo objetivo é especificar o ADS mais adequado para o desenvolvimento de um determinado software através da definição do processo de desenvolvimento a ser realizado e da indicação de métodos e ferramentas de software (Aguiar, 1992), (Massolar, 1993).

Esta primeira versão foi desenvolvida para estações de trabalho Sun, utilizando Xview e a linguagem C. Este sistema possui um sub-sistema baseado em conhecimento que foi desenvolvido em Prolog com uma base de conhecimento sobre métodos e ciclos de vida. Além desses componentes existe também uma base de características de aplicações e uma base de elementos de ADS (atividades, métodos e ferramentas).

---

[1]O coordenador técnico é o responsável a nível de informática, com conhecimento sobre sistemas baseados em conhecimento e projetos de desenvolvimento de sistemas de computação.

A alteração realizada em XAMÃ para incorporar elementos que permitam a especificação do ambiente ORIXÁS, em situações onde seu uso é o adequado, implicou na inclusão de características de sistemas baseados em conhecimento na base de características de domínios de aplicação. Na base de conhecimento de XAMÃ foram incluídas novas regras, que indicam as situações onde é adequado o uso do ciclo de vida do ambiente ORIXÁS e o método KADS-estendido. XAMÃ contém, ainda, um conjunto de arquivos com informações sobre as atividades pertinentes a cada modelo de ciclo de vida, descrição de métodos e sugestão de ferramentas a serem desenvolvidas. Assim sendo, informações relativas às atividades do ciclo de vida de ORIXÁS, aos métodos utilizados e às ferramentas sugeridas para implementação foram incluídas em XAMÃ.

As figuras VI.5, VI.6 e VI.7 mostram as janelas relativas ao ambiente ORIXÁS em XAMÃ.

VI.5 O Ambiente Instanciado ORIXÁS

A modelagem de ADSs específicos no TABA, introduz a abordagem de *framework*. O *framework* é projetado para ser refinado ou especializado. A classe FrmAmbientes representa as características comuns dos ADSs no TABA, que devem ser especializadas para adequação aos ADS específicos.

Cima et alii (1994) discutem duas formas de especialização de *framework*. A primeira forma inclui o comportamento específico da aplicação na arquitetura genérica, acrescentando-se métodos às subclasses de uma ou mais classes do *framework*. Esta estrutura é denominada *framework* do tipo *caixa branca* e embora seja bastante flexível requer a criação de muitas subclasses na sua especialização. A segunda forma de especialização, *framework caixa preta*, recebe um conjunto de parâmetros que representa o comportamento específico da aplicação. O meta-ambiente da Estação TABA utiliza um *framework* deste tipo no processo de instanciação de ADSs.

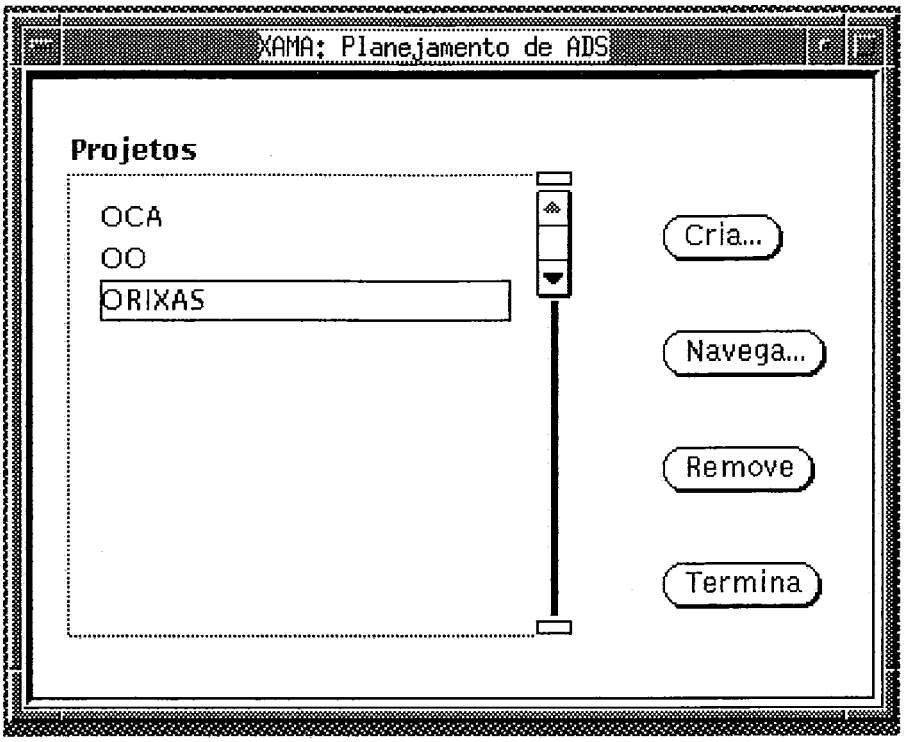


Figura VI.5 - Janela de Planejamento de ADS do sistema XAMÃ

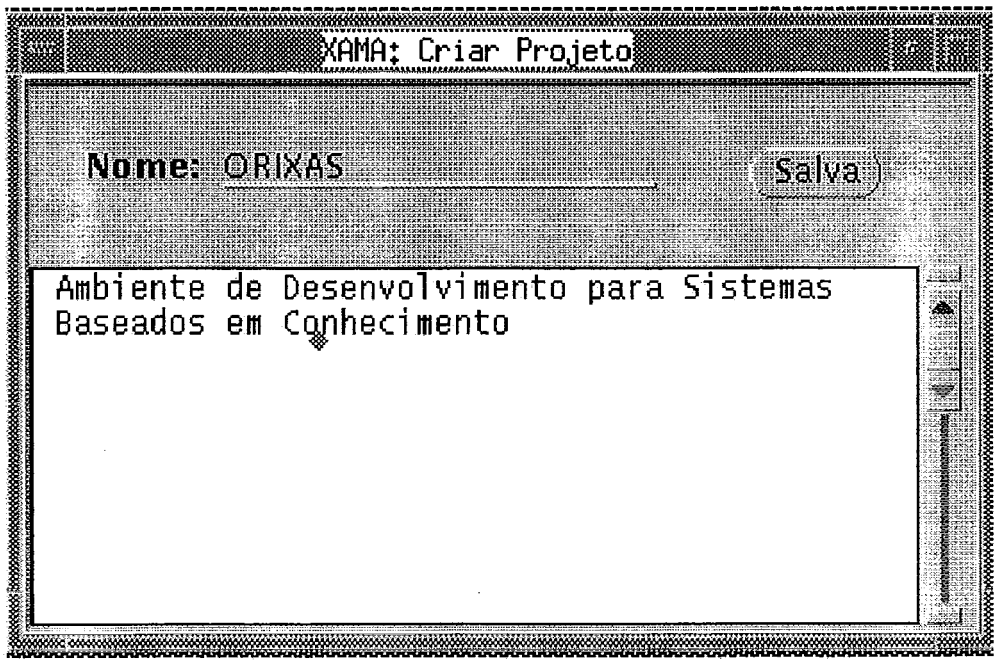


Figura VI.6 - Janela de Criar Projeto do sistema XAMÃ

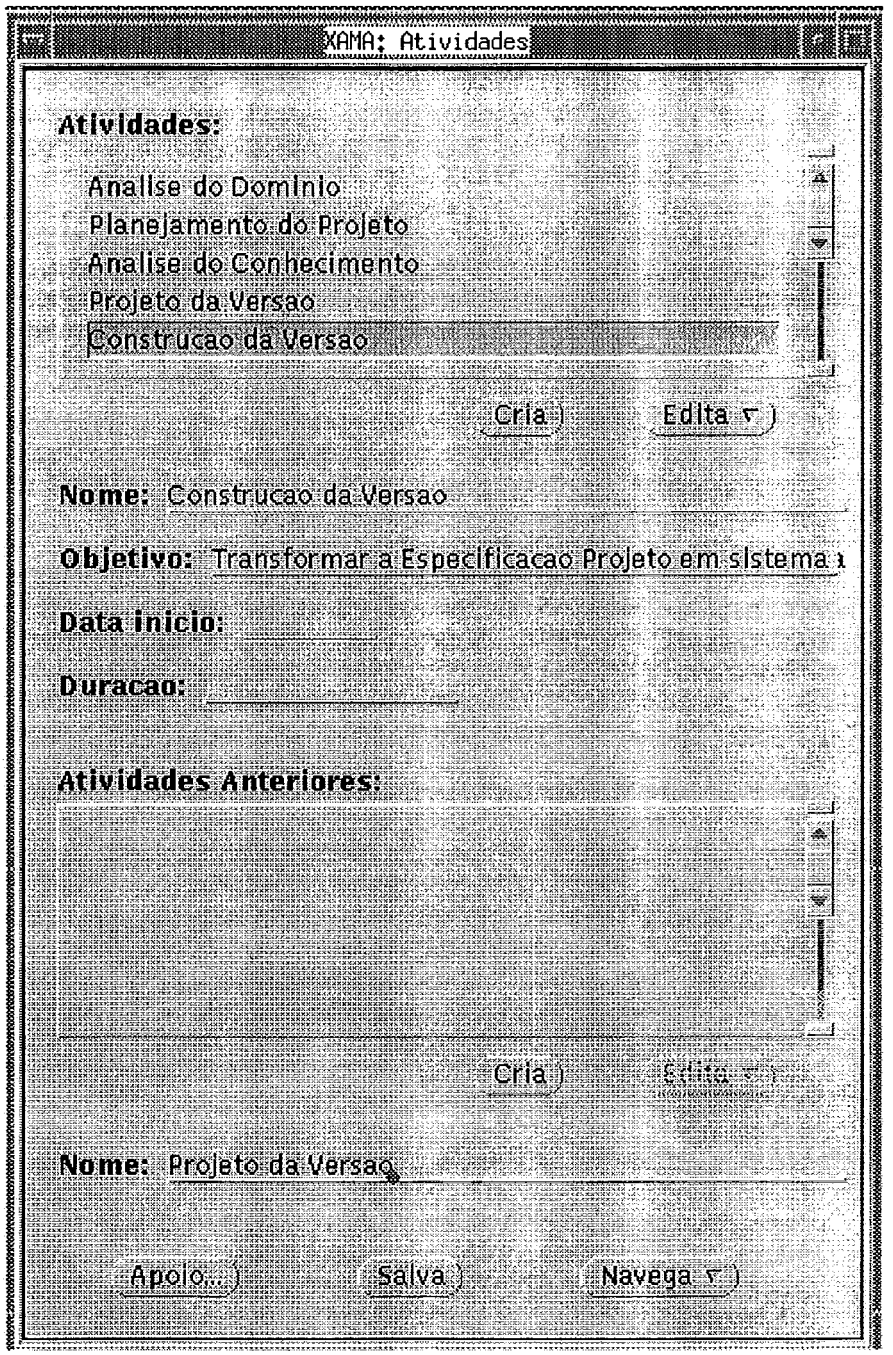


Figura VI.7 - Janela de Atividades do sistema XAMÃ

A maioria das linguagens orientadas a objetos não possuem suporte direto à criação de classes abstratas (Wirfs-Brock e Johnson, 1990) o que acarretava problemas a nível técnico à utilização de *framework*. Entretanto, algumas linguagens atuais, por exemplo Eiffel (ISE, 1994), suportam a definição de *framework*.

A modelagem do ambiente ORIXÁS é uma especialização do meta-ambiente da Estação TABA. A Figura VI.8 apresenta o diagrama de assuntos para esta especialização e o Anexo V contém o diagrama completo do meta ambiente.

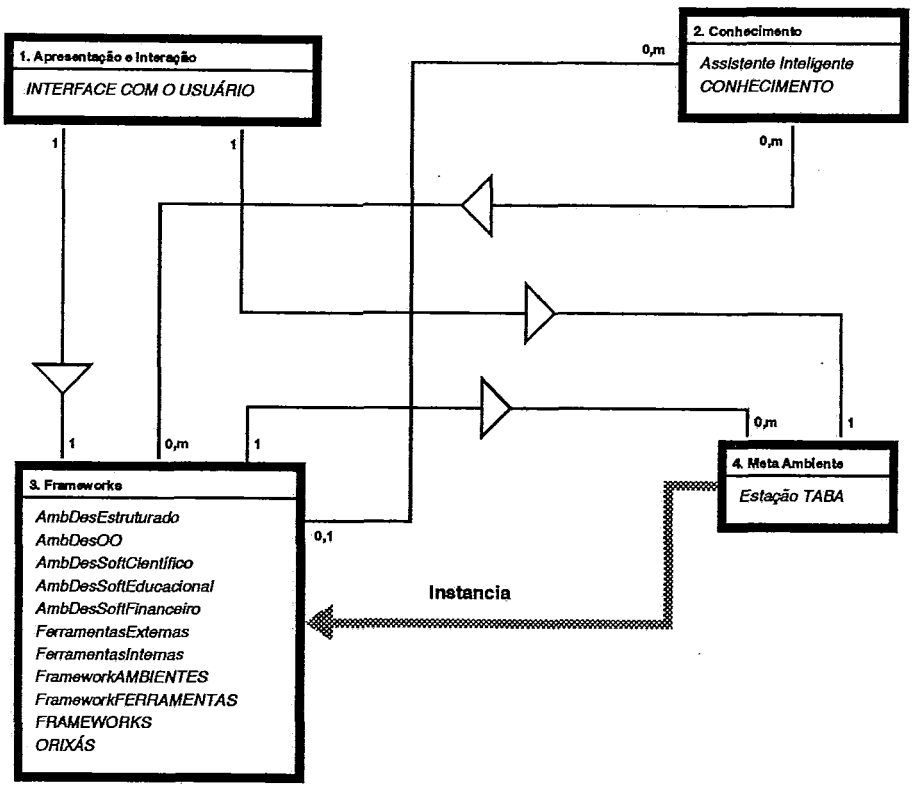


Figura VI.8- Diagrama de Assuntos do Meta Ambiente TABA

A nível de ADS instanciado foi necessária uma revisão da modelagem do TABA para que fosse possível a instanciação do ambiente ORIXÁS. A Linguagem para Descrição e Manipulação de Métodos (LDMM) foi extendida para permitir a definição do método KADS-estendido.

No Anexo IV pode ser encontrado um resumo da linguagem LDMM e a descrição da extensão realizada. A nível de exemplo de utilização de LDMM encontram-se no anexo IV a definição do *Modelo de Especialidade*, o *Modelo Lógico* e o *Diagrama Estrutural da Base de Conhecimento (Modelo Físico)*.

A Figura VI.9 apresenta o novo diagrama de assuntos do modelo de ADS instanciado TABA e no Anexo V encontra-se o diagrama completo.

As principais alterações ocorridas no modelo de ADS instanciado, para incorporar as características do ORIXÁS, com relação à proposta de Travassos e Werner (1995) foram as seguintes:

- definição dos atributos *Tipo e Número de Estágio* na classe *Atividades*, no assunto *Processo*;
- definição das classes *Decisor*, *Ordenador*, *DependênciaConjunto*, *DependênciaLógica* e *DependênciaLógicaConjunto* com seus atributos e relacionamentos. Essas classes permitem a utilização da classe *Grafos* e surgiram ao se definir os modelos em LDMM (Anexo V). A classe *Decisor* é um sub-tipo da classe *nós* e as demais são subclasses da classe *Ligações*;
- definição da classe *ferramentas KADS-estendido* que é uma subclasse de ferramentas Internas. As ferramentas definidas na seção VI.3.3 deverão ser desenvolvidas ou integradas para a efetiva disponibilização do ambiente ORIXÁS, estando no nível meta como sugestão de ferramentas.

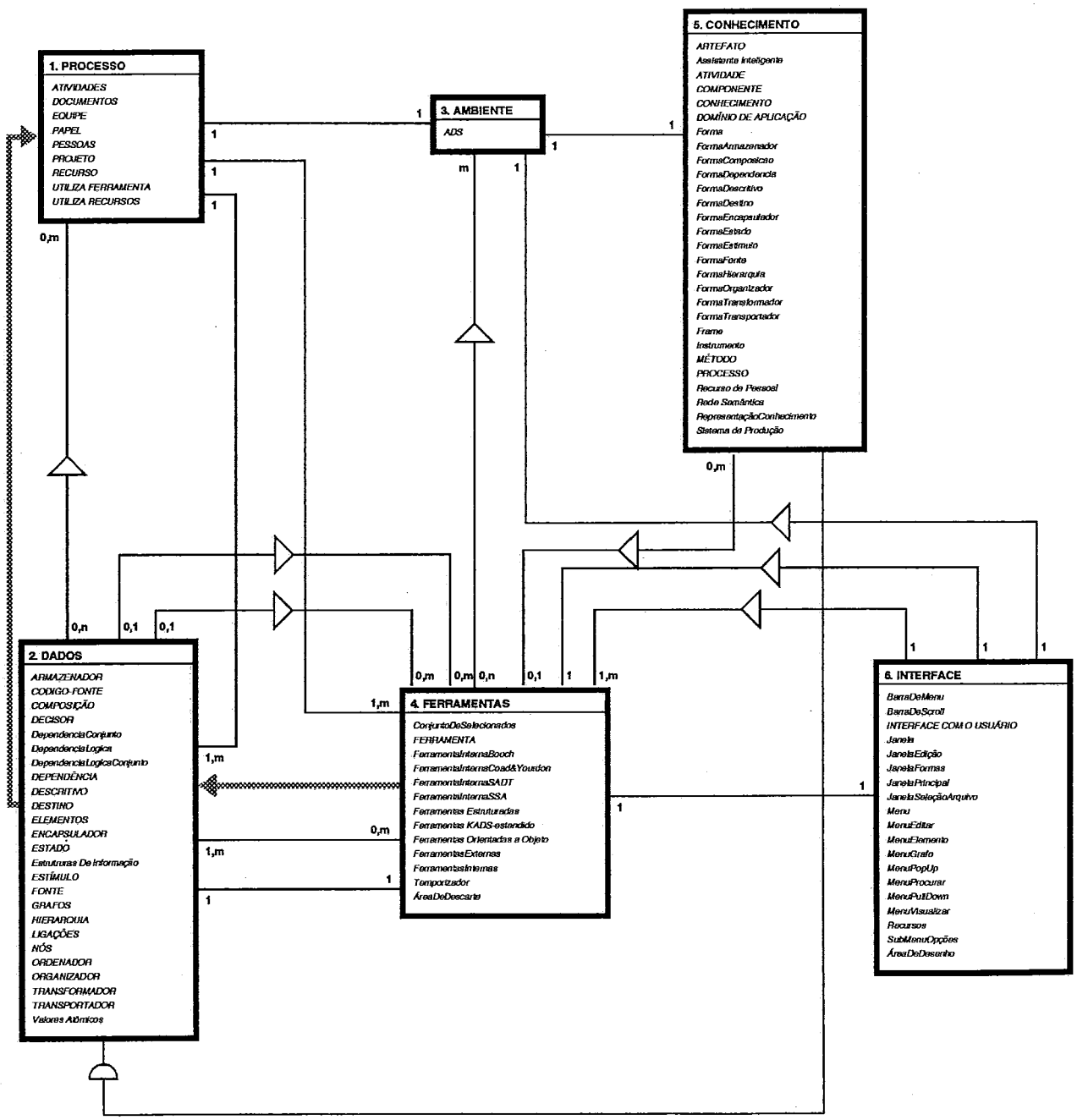


Figura VI.9- Diagrama de Assuntos dos Ambientes Instanciados TABA

VI.6. Protótipo do Ambiente ORIXÁS

A Estação TABA, em sua primeira versão operacional, foi implementada num sistema gerenciador de banco de dados orientado a objetos denominado O_2 (1993). Esta versão está descrita em Travassos (1994). Atualmente, se está implementando uma nova versão da Estação TABA utilizando a linguagem Eiffel em sua versão 3.2. As facilidades disponíveis em Eiffel permitem a utilização de *frameworks* através de herança simples e múltipla, construção de classes genéricas e abstratas e a filosofia de contratos com definição de pré-condições, pós-condições e condições invariantes para seus métodos (ISE, 1994). A especificação completa desta nova versão pode ser encontrada em Travassos e Werner (1995).

Neste contexto, ORIXÁS, é definido como uma especialização da classe FrmAmbientes, pois consiste na definição abrangente de um ambiente de desenvolvimento para sistemas baseados em conhecimento. Ao se iniciar um projeto onde será desenvolvido um sistema baseado em conhecimento, o ambiente ORIXÁS deverá ser particularizado considerando-se o domínio do problema a ser tratado e a especificidade do projeto em questão.

O protótipo de ORIXÁS está inserido no Ambiente Especificador e Instanciador de ADSs. A construção deste primeiro protótipo de ORIXÁS foi realizada através de sua modelagem como ambiente instanciado e do mecanismo de integração de ferramentas da Estação TABA (Anexo V). Foi utilizada a linguagem Eiffel versão 3.2 (ISE, 1994).

A primeira versão do ambiente ORIXÁS implementa, apenas, o assunto processo. Como a primeira atividade no desenvolvimento de um projeto é estabelecer seu processo de desenvolvimento, este foi o assunto escolhido. Além disso, tendo-se este assunto implementado, tem-se a infra-estrutura do ambiente ORIXÁS, dado que as demais ferramentas serão chamadas a partir das atividades definidas no processo.

Assim sendo, através do protótipo, pode-se criar um novo projeto que será desenvolvido e particularizar o processo de desenvolvimento do ambiente ORIXÁS de forma a atender às especificidades do projeto em questão, o que é feito através das

seguintes atividades: planejamento do processo de desenvolvimento e planejamento da documentação. Desta forma a cada atividade do processo são definidos os documentos que serão produzidos. A chamada às diversas ferramentas definidas para ORIXÁS é feita através da atividade do processo que irá necessitá-la. Tem-se, assim, a estrutura do ambiente. As Figuras VI.10, VI.11 e VI.12 apresentam as principais janelas deste protótipo. .

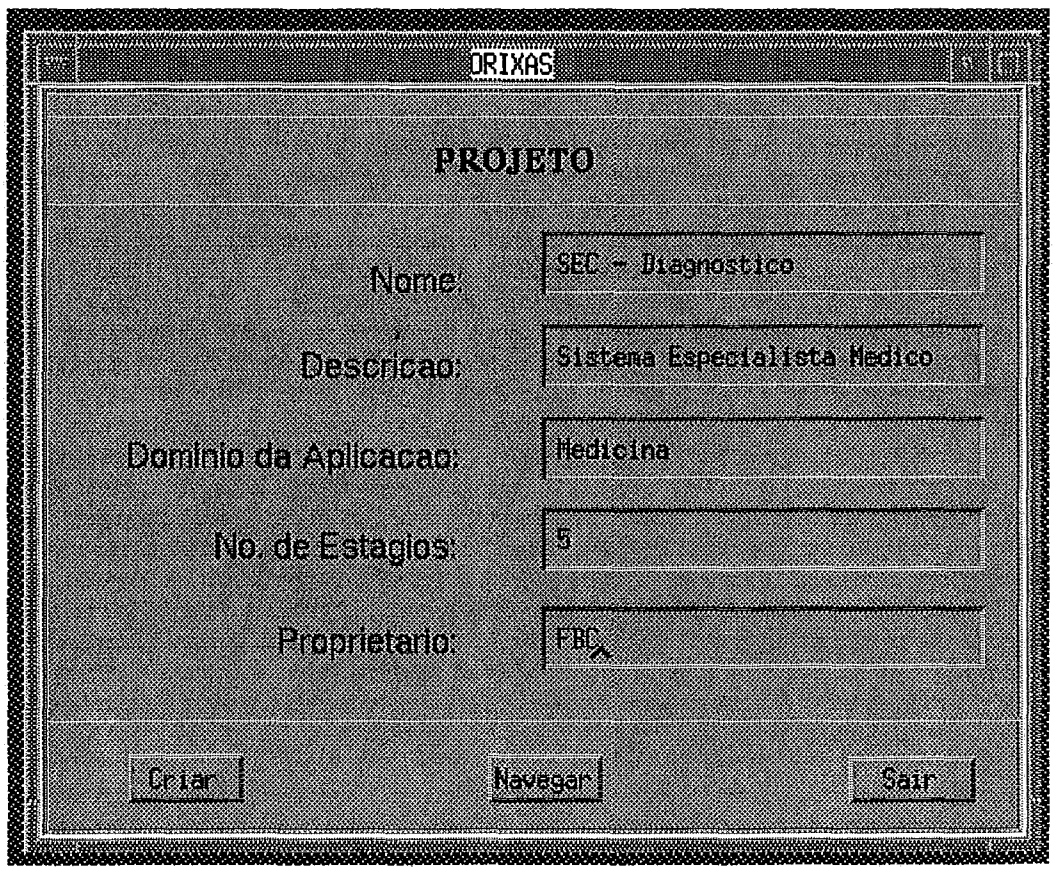


Figura VI.10 - Janela Inicial do ADS ORIXÁS

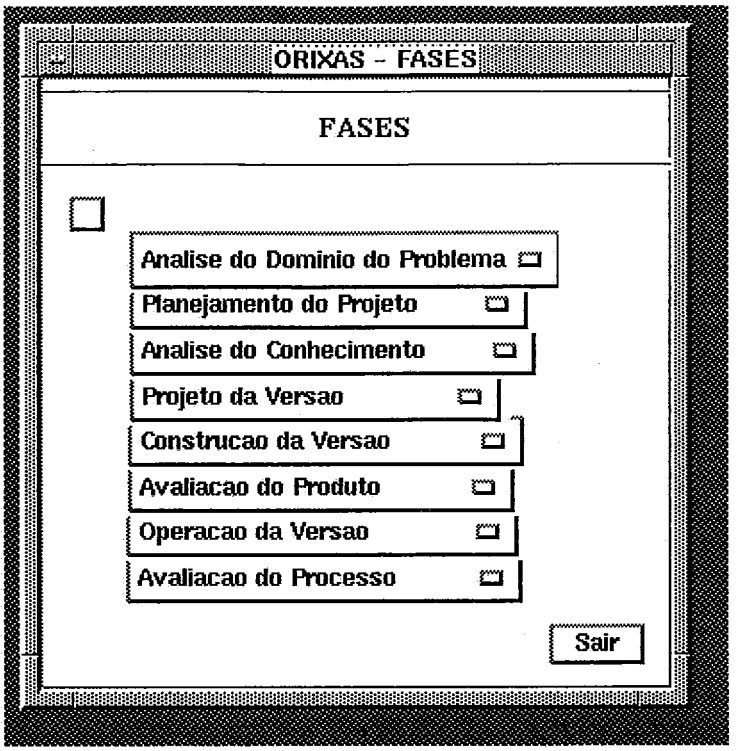


Figura VI.11 - Janela de Atividades do ADS ORIXÁS

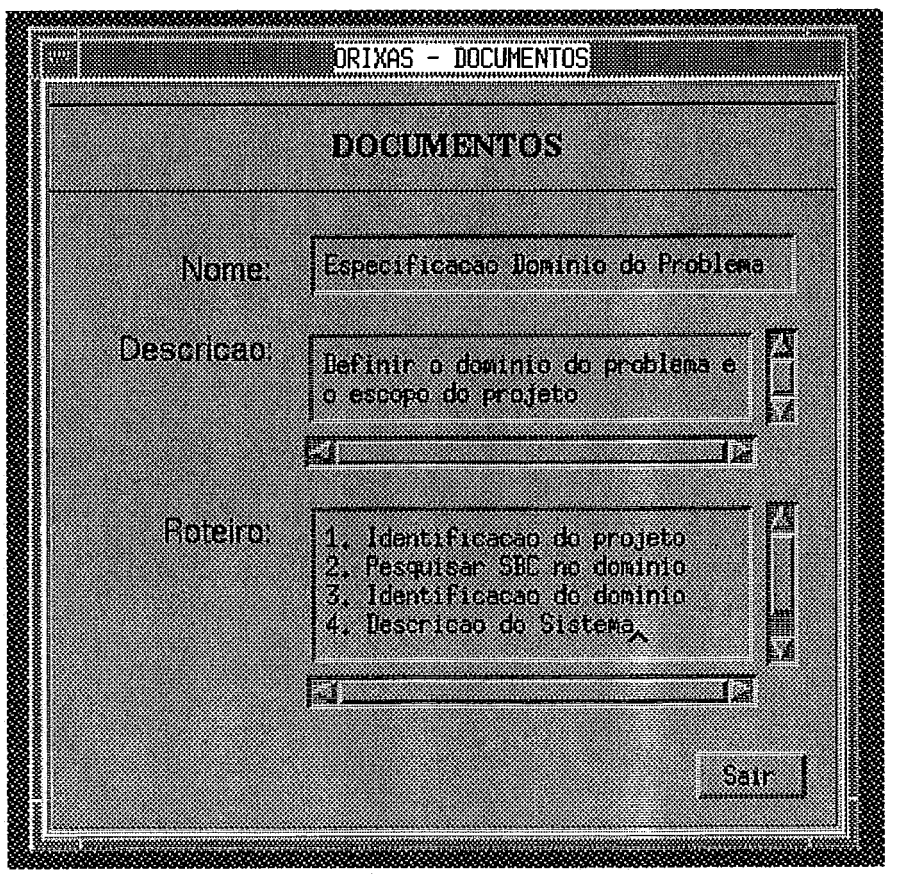


Figura VI.12 - Janela de Documentos do ADS ORIXÁS

VI.7. Conclusão

Neste capítulo foi definido ORIXÁS, o Ambiente para Desenvolvimento de Sistemas Baseados em Conhecimento da Estação TABA, que está baseado no processo de desenvolvimento definido no Capítulo IV, no método KADS-estendido definido no Capítulo V e nos procedimentos para avaliação de sistemas baseados em conhecimento definido em Oliveira (1995).

ORIXÁS é um ambiente instanciado TABA e foi definido neste contexto. O primeiro protótipo do ambiente foi implementado na linguagem Eiffel Versão 3.2 a partir de sua modelagem como ambiente instanciado. Este primeiro protótipo mostra a viabilidade de implementação do ORIXÁS como um ambiente instanciado TABA.

Capítulo VII
Conclusões e Perspectivas Futuras

Este trabalho teve como objetivo definir um processo adequado ao desenvolvimento de sistemas baseados em conhecimento e propor um ambiente de desenvolvimento de software baseado neste processo.

O processo de desenvolvimento para sistemas baseados em conhecimento, foi definido a partir das normas ISO 9000-3 e ISO 9126, contemplando as atividades relativas à construção e avaliação da qualidade do produto e à gerência do projeto. O processo foi definido e validado através da experiência no desenvolvimento de vários SBCs sendo, entre estas, a mais significativa a experiência no projeto Sistemas Especialistas em Cardiologia, da UCCV/FBC.

O ambiente de desenvolvimento de software foi definido, no contexto do Projeto TABA da COPPE/UFRJ, a partir do processo estabelecido.

O projeto TABA tem como objetivo construir uma Estação de Trabalho para o engenheiro de software, que permita a definição e implementação de ambientes adequados ao desenvolvimento de software em diferentes domínios de aplicação e utilizando diversas tecnologias de desenvolvimento. Para atingir este objetivo a Estação TABA contém um meta-ambiente com as funções de especificar o ambiente de desenvolvimento de software mais adequado a um projeto, instanciar e tornar operacional o ambiente especificado.

Este trabalho, teve como ponto de partida os trabalhos realizados anteriormente no âmbito do Projeto TABA: XAMÃ, um assistente baseado em conhecimento que apoia o engenheiro de software na especificação de ambientes e o trabalho realizado por Travassos (Travassos 94) que tornou operacional a instanciação dos ambientes especificados por XAMÃ, através de um modelo para integração de ferramentas e da construção de ferramentas a partir da utilização de *frameworks*. Neste contexto foi, então, definido o ambiente ORIXÁS, um ambiente instanciado TABA, adequado ao desenvolvimento de sistemas baseados em conhecimento.

São contribuições deste trabalho:

- a definição de um processo, completo e sistemático, adequado ao desenvolvimento de sistemas baseados em conhecimento;
- o relato da experiência no uso do processo no desenvolvimento de um projeto real, o SEC-Diagnóstico, e os procedimentos utilizados para avaliação e aperfeiçamento do processo;
- a experiência de uso e a avaliação realizada no método KADS;
- a definição do método KADS-estendido;
- a extensão da Linguagem para Definição e Manipulação de Métodos para descrição do método KADS-estendido;
- a especialização do modelo de ambientes instanciados TABA, para incorporar a definição do ambiente de desenvolvimento de sistemas baseados em conhecimento;
- a definição do ambiente ORIXÁS, um ambiente instanciado TABA para sistemas baseados em conhecimento, que integra o apoio à construção do produto com o apoio à avaliação da qualidade e gerência do projeto.

Neste contexto, surgem várias perspectivas para pesquisas que dêm continuidade a este trabalho:

- a implementação do ambiente ORIXÁS;
- a realização de experiências de uso do processo, do KADS-estendido e do ambiente ORIXÁS em projetos, de forma a permitir novos aperfeiçoamentos;
- a definição e implementação de ambientes que especializem ORIXÁS. Neste sentido, tem-se já previstos dois ambientes: um ambiente específico para o desenvolvimento de software médico e um ambiente para desenvolvimento de tutores;

- a continuidade e aprofundamento do trabalho realizado em Oliveira (1995), no que se refere à verificação e validação de sistemas baseados em conhecimento;

- a definição de procedimentos gerenciais, tais como técnicas para estimativa de custos, tempo de desenvolvimento e análise de riscos associados ao desenvolvimento de sistemas baseados em conhecimento.

Finalmente, cabe ressaltar um aspecto que foi essencial para a realização deste trabalho. Não se faz Engenharia de Software sem a experiência de projetos reais, que superem a escala de projetos acadêmicos. Apenas neste âmbito é possível perceber os problemas que acontecem no desenvolvimento de produtos de software.

A definição do processo de desenvolvimento e do método KADS-estendido, propostos nesta tese, não teriam sido possíveis sem a participação no projeto SEC. A definição do processo de desenvolvimento para o SEC e sua posterior utilização, avaliação e reformulação foi a base fundamental para a realização deste trabalho. Por isso o ambiente ORIXÁS é uma homenagem à Bahia e aos baianos. Sem a acolhida da UCCV/FBC ao nosso trabalho ele não teria se realizado ou não seria o mesmo.

Referências Bibliográficas

Aguiar, Teresa C.; **"Um Sistema Especialista de Suporte à Decisão para Planejamento de Ambientes de Desenvolvimento de Software"** Tese de Doutorado, Rio de Janeiro, COPPE/UFRJ, 1992.

Allen, Bradley P., "Case Based Reasoning: Business Applications"; **Communications of the ACM**, Volume 37, No 3, Março de 1994, 40-42.

Angele, J. et alii, "Model-Based and Incremental Knowledge Engineering: The Mike Approach"; in **Knowledge Oriented Software Design,** José Cuena (ed.), North Holland, 1994.

Anjewierden, A.; "Shelley - Computer-Aided Knowledge Engineering" em **"KADS: A Principled Approach to Knowledge-Based System Development"**; Schreiber,G., Wielinga, B. e Breuker, J.(ed.); Academic Press; 1993.

Arango, Guilherme e Prieto-Diaz, Ruben; "Overview: A Domain Analysis Concepts and Research Directions**";** em **"A Domain Analysis and Software Systems Modeling";** 1991.

Arango, Guilherme; "Domain Analysis Methods: Software Reusability**";** Schäfen, Wilhelm, Prieto-Diaz, Rubem, Matsumoto, Masao, (ed); Ellis Horwood Inc, 1994.

Arora, V. e Cooke, J. E.; "Towards Effective Management of Expert Systems Projects", **Proceedings of the IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 339-345.

Arthur, Lowell; **Improving Software Quality**; New York; John Wiley & Sons; 1993.

Arthur, Lowell; **Measuring Programmer Productivity and Software Quality**; New York; John Wiley & Sons; 1985.

Asawaka, K. e Takagi, H.; "Neural Network in Japan", **Communications of the ACM**, Volume 37, Número 3, Março de 1994, 105-112.

Assis, Selma Glitz de; **"Aquisição de Conhecimento para Sistemas Baseados em Conhecimento: Uma Experiência de Engenharia de Software"**; Tese de Mestrado, Rio de Janeiro, COPPE/UFRJ, 1992.

Atabakhsh, Homa e Chan, Albert W.; "Incorpoating Rules with Object: A Hybrid Methodology for Decision Support"; **Object Magazine**, Março de 19484, 26-37,46.

Avanzato, Robert L.; "Uncertainty Management Issues in The Development of An Acoustic Signal Interpretation Expert System, **IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 68-75.

Bader, Jon, Edwards, John, Harris-Jones, Chris, Hannaford, David; "Practical Engineering Methodology of Knowledge-Based Systems"; **Information and Technology**; Volume 30, Nº 5; Junho 1988, 266-277.

Bailor, Paul D.; "Educating Knowledge-Based Software Engineers"; **Proceedings KBSE 92 - 7th Knowledge-Based Software Engineering Conference**; Setembro 1992, 226-237.

Balzer, T.;"Transformational implementation: an example"; **IEEE Transactions on Software Engineering**, Volume 7, Nº 1; Janeiro, 1981, 3-14.

Basu, A. e Hevner, A. R.; "Box Structured Development of Embedded Knowledge Based Systems", **Proceedings of the IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**; Setembro/Outubro 1991, 304-313.

Basu, Amit e Ahad, Rafiul; "Using Relation Database to Support Explanation in a Knowledge-Based System"; **IEEE Transactions on Knowledge and Data Engineering**; Volume 4, Número 6, Dezembro 1992, 572-581.

Belchior, Arnaldo Dias; **"Qualidade de Software Financeiro",** Tese M.Sc.,COPPE/UFRJ, junho 1992.

Belchior,A.D.;Rocha,A.R.C.da "Meta-Avaliador de Especificações"; **Workshop Qualidade de Software,** Curitiba, PR, outubro 1994.

Bench-Cupon, T., Coenen, F., Nwana, H., Paton, R. e Shave, M.; "Two Aspects of The Validation and Verification of Knowledge-Based Systems"; **IEEE Expert**; Junho 1993, 76-81.

Bersoff, E.H., Davis, A.M.; "Impacts of Life Cycle Models on Software"; **Communications of the ACM,** Volume 34, Nº 8; Agosto 1991.

Blackman, Michael J.; "Case for Expert Systems"; **AI Expert**; Fevereiro 1990, 27-31.

Bochsler, D. C. e Goodwin, M. A.; "Software Engineering Techniques Used to Develop an Expert System for Automating Space Vehicle Rendezvous"; **ROBEX 86: Second Annual Workshop on Robotics and Expert Systems**; Junho 1986.

Boehm, Barry; **"Software Engineering Economics"**; Prentice Hall; 1981.

Boehm, Barry; "A Spiral Model of Software Development and Ehancement"; **Computer**; Maio 1988, 61-72.

Boehm, Barry; Software Engineering - AS IT IS; Academic Press; 1980.

Botten, N., Kusiak, A e Raz, T.; "Knowledge Bases: Verification and Partitioning"; **European Journal of Operational Research**, Número 42; 1989; 111-128.

Breuker, J. A. e Greef, P.; "Modelling System-User Cooperation in KADS" em "**KADS: A Principled Approach to Knowledge-Based System Development**"; Schreiber, G.., Wielinga, B. e Breuker, J.(ed.); Academic Press; 1993.

Breuker, J. A. e Wielinga, B. J.; **"Models of Expertise in Knowledge Acquisition";**Guida G. e Tasso C. (Eds.), Amsterdam, North-Holland, 1988.

Buchanan, Bruce G. e Shortliffe, Edward H. (ed.); **"Rule-Based Expert Systems: The MYCIN Experiments of Stanford Heuristic Programming Project"**; Massachusetts, Addison Wesley, 1985.

Campos, Gilda Helena B.; de **"Metodologia para Avaliação da Qualidade de Software Educacional: Diretrizes para Desenvolvedores e Usuários"**, Tese D.Sc.; COPPE/UFRJ, novembro 1994.

Campos,G.H.B.de e Rocha,A.R.C. "Some Trends and Needs for Latin American Countries in Computers and Education"; **The Tenth International Conference on Technology and Education; MIT**, Cambridge,MA, março 1993.

Chandersekaran, C. S., Linder, R. C.; "Software Specifications using SPECIAL Language"; **The Journal of Systems and Software**; Fevereiro 1981.

Charette, Robert (1986); **"Software Engineering Environments: Concepts and Technology"**; New York; McGraw-Hill Inc.; 1986.

Chen, Peter; **"The Entity-Relationship Approach to Logical Data Base Design";** Information Sciences, 1977.

Cherubini, M. A., Fanti, L., Torrigiani, P e Zallocco, M.; "An Integrated Expert-Systems Builder"; **IEEE Software**; Novembro 1989, 44-52.

Cho, TaiHoon; Conners, Richard W. e Araman, Philip A.; "A Comparison of Rule-Based, K-Nearest Neighbor e Neural net Classifiers for Automated Industrial Inspection"; **IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 203-209.

Cima A. M., Werner, C. e Travassos, G. H; "O Uso de Frameworks como Arquitetura Reutilizáveis na Construção de Ambientes de Desenvolvimento de Software"; **VIII Simpósio Brasileiro de Engenharia de Software**; Curitiba, Paraná; outubro 1994.

Clunie,C.; Werner,C.; Rocha, A.R.C.da "Qualidade de Especificações Orientadas a Objetos"; **Congresso Latinoamericano de Informática**; Cidade do México, México; setembro 1994.

Coad, P. e Yourdon, E.; **Análise Baseada em Objetos**; Editora Campus, 1992.

Coats, Pamela; "Why Expert Systems Fail"; **Financial Management**, Autumm 1988, 77-86.

Comerlato, Cesar; **"Uma Ferramenta para Seleção de Componentes Reutilizáveis"**, Tese M.Sc., COPPE/UFRJ, abril 1994.

Commerlato,C.; Xexeo,G.B.; Rocha,A.R.C.da "Avaliação da Reutilizabilidade de Componentes de Software"; **VIII Simpósio Brasileiro de Engenharia de Software**; Curitiba, Paraná; outubro 1994.

Connors, Danny T.;"Software Development Methodologies and Traditional and Modern Information Systems"; **Software Engineering Notes**, Volume 17, Nº 2; Abril 1992, 43-49.

Crispim, E.M.e Rocha, A.R.C.da "Avaliação de Métodos para Desenvolvimento de Software"; **XII Congresso de Metodologias en Ingenieria de Sistemas**; Santiago, Chile, julho 1992.

Cuena, José, "Contributions to a Knowledge Oriented View of Software Design" in **Knowledge Oriented Software Design,** José Cuena (ed.), North Holland, 1994.

Culbert, C.J., Riley, G. e Savely, R. T.; "Approaches to the Verification of Rule-Based Expert Systems"; SOAR 87: **Proceedings of the First Annual Workshop on Space Operation Automation and Robotics**; Agosto 1987.

Czejdo, B., Eick, C. F. e Taylor, M.; "Integrating Sets, Rules and Data in an Object-Oriented Environment"; **IEEE Expert**; Fevereiro 1993, 59-66.

Dai, H. e Hughes, J. G.; "A Distributed Real-Time Knowledge-Based System and Its Implementation using Object-Oriented Techniques"; **Proceedings of International Conference on Intelligent and Cooperative Information Systems**, IEEE Computer Society Press, Maio 1993, 23-30.

Davis, Alan; **"Software Requirements: Analysis and Specification"**; Englewood Cliffs, New Jersey; Prentice Hall Inc, 1990.

Davis,Alan M.; "Operational Prototyping: a new development approach"; **IEEE Software**; Setembro, 1992.

Davis, Randall; "Meta-Rules: Reasoning about Control"; **Artificial Intelligence**, No 15, 1980, 179-222.

Davis, Randall e Lenat, Douglas B.; **"Knowledge Systems in Artificial Intelligence"**; New York, Mc Graw Hill, 1982.

De Marco, Tom; **"Análise Estruturada e Especificação de Sistema"**; Editora Campus, 1989.

Denning,P.; "What is software Quality", **Communications of ACM**; Volume 35, 1992; 12-15.

Duarte, Marco Antonio G.; **"Um Sistema para Representação do Conhecimento de Métodos de Desenvolvimento de Software"**; Tese de Mestrado, Rio de Janeiro, COPPE/UFRJ, 1991.

Ducassé, Mireille e Emde, Anna Maria; "OPIUM: A Debugging Environment for Prolog Development and Debugging Research", **Software Engineering Notes**, Volume 16, Nº 1; Janeiro 1991, 54-59.

Duff, B. L. e Carlson, C. K.; "Applications of Object-Oriented Approaches to Expert Systems in the Earth Sciences", **Proceedings of the IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 160-165.

Fairley, Richard; **Software Engineering Concepts**; McGraw-Hill, Inc.; 1986.

Falbo, R. A. e Travassos, G. H.;"Um Estudo sobre Ambientes de Desenvolvimento de Software com Suporte Baseado em Conhecimento"; XV Congresso da Sociedade Brasileira de Computação/XXI Conferência Latino Americana de Informática; Canela,RS; Julho 1995.

Falcão, João Fernando D.; **"Um Sistema Baseado em Conhecimento de Apoio à Métodos de Desenvolvimento de Software"**; Tese de Mestrado, Rio de Janeiro, COPPE/UFRJ, 1992.

FBC - Fundação Bahiana de Cardiologia; **"Especificação de Requisitos do Projeto Sistemas Especialistas em Cardiologia(SEC)"**; documento interno, 1994a.

FBC - Fundação Bahiana de Cardiologia; **"Plano do Projeto Sistemas Especialistas em Cardiologia (SEC)"**; documento interno, 1994b.

Feigenbaum, Edward A. ("chair"), Friedland, Peter E., Johnson, Bruce B., Nii, H. Penny, Schorr, hebert, Shrobe, Howard e Engelmore, Robert S, "Knowledge Based Systems in Japan"; JTEC Panel Report, **Communications of the ACM**, Volume 37, Número 1, Janeiro de 1994, 17-19.

Felix, Rudolf; "Reasoning with Uncertainty in the Knowledge Engineering Environment KEE"; **IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 231-235.

Fenn, J. A. e Veren, L. C.; "Expert Systems Development Methodologies in Teory and Practice", **Proceedings of the IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 262-266.

Ferreira, Ulisses; **"LIDIA, uma Linguagem para Desenvolvimento de Sistemas Especialistas com Tratamento de Incertezas"**, Tese de Mestrado, UFPB, Campina Grande,1990.

Fikes, Richard e Kehler, Tom; "The Role of Frame-Based Representation in Reasoning"; **Communications of ACM**, Volume 28, No 9, Setembro 1985, 904-920.

Forsyth, Richard (ed.); **"Expert Systems: Principles and Case Studies"**; London, Chapman and Hall, 1984.

Garcia, Oscar N. e Chien, Yi-Tzuu; **"Knowledge Systems: Fundamentals and Tools"**; IEEE/ Computer Society Press, 1991.

Garrett, R.; "Out of the Lab and into The Field: System Design of Large Expert Systems", **Proceedings of the IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 273-278.

Geissman, J. R. e Schultz, R. D.; "Verification and Validation of Expert Systems"; **AI Expert**; Fevereiro 1988, 26-33.

Gevarter, W. B.;"The Nature and Evaluation of Commercial Expert Systems Building Tools"; **Computer**, Volume 20, Número 5, Maio de 1987, 24-41.

Ghezzi, Carlo, Jazayeri, Mehdi e Mandrioli, Dino; **"Fundamentals of Software Engineering"**; Prentice-Hall Internacional Inc, New Jersey.

Gottgtroy, Márcia P. B.; **O Processo de Aquisição do Conhecimento na Construção de Sistemas Especialistas;** Tese de Mestrado, Rio de Janeiro, COPPE/UFRJ, 1990.

Greef, P. e Breuker, J. A.; "Analysing System-User Cooperation in KADS" em **"Knowledge Acquisition: The KADS Approach to Knowledge Engineering";** Schreiber,G.(ed.); Academic Press; 1992.

Green, C. J. R. e Keyes, M. M.; "Verification and Validation of Expert Systems"; **Western Conference on Expert Systems**, 38-43.

Guida, Giovani e Tasso, Carlo; **"Design and Developments of Knowledge-Based Systems: from Life cycle to Methodology";** John Wiley & Sons; 1994.

Gupta, Uma G.; **"Validation and Verification of Knowledge-Based Systems"**; **IEEE Computer Society Press**, California, USA.

Hall, Curtis; "Neural Net Technology: Ready for Prime Time?"; **IEEE Expert**, Volume 7, No 6, Decembro 1992, 1-3.

Harmon, Paul e King, David; **Sistemas Especialistas**; Rio de Janeiro, Editora Campus, 1988.

Hatley, D, e Pirbhai, I..; **"Estratégias para Especificação de Sistema em Tempo Real"**; Mc Graw Hill, 1991.

Haynes, P. J.; "The Logic of Frames" em **"Frame Conceptions and Text Understanding"**; Dieter Metzing (ed.), Berlin, Walter de Gruyter, 1980.

Haynes-Roth, Frederick; "Knowledge Bases Expert Systems"; **IEEE Computer**, Volume 17, Número 10, Outubro de 1984, 263-271.

Haynes-Roth, Frederick e Jacobstein, Neil, "The state of Knowledge Based Systems"; **Communications of the ACM**, Volume 37, Número 3, Março de 1994, 27-39.

Hembry, D.M.; "Knowledge-based Systems in the AD/Cycle Environment"; **IBM Systems Journal**, Volume 29, Número 2, 1990, 274-286.

Hickman, Frank et alii; **"Analysis for Knowledge-Based System: A Practical Guide to the KADS Methodology"**; Ellis Horwood; 1989.

Hillmer, D. et alii.; "Identifying Risks in Expert System Projects", **Proceedings of IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 45-52.

Hoppe, Thomas e Meseguer, Pedro; "VVT Terminology: A Proposal"; **IEEE Expert**, Junho 1993, 48-55.

Hu, David S.; **"Expert Systems for Software Engineers and Managers"**; New York, Chapman and Hall, 1987.

Hull, L. G. e Kay, P.; "Expert System Development Methodology and Management", **Proceedings of IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 38-44.

IEEE/ACM; **Proceedings of the IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**, IEEE Computer Society Press, Los Alamitos, Valifornia, Setembro/Outubro 1991.

ISE - Interative Software Engineering; **"The Eiffel: An environment Manual Version 3.2.2"**; Janeiro de 1994.

ISO 9000, **"Quality Management and Quality Assurance Standards - Guidelines for the Application of ISO 9000"**, ISO, setembro 1987.

ISO 9000-3, **"Quality Management and Quality Assurance Standards - Part 3: Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software"**, ISO, setembro 1990.

ISO/IEC 9126, **"Information Technology - Software Evaluation - Quality Characteristics and Guidelines for their use"**; ISO, dezembro, 1991.

Ito, M., Okemura, S., Okuzawa, O. e Takahashi, S.; "Software System Integration Support Expert System Based on the Object-Oriented Method", **IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 106-113.

Jackson, Michael.; **"System Development"**; Prentice-Hall, 1983.

Jacob, Robert J. K. e Froscher, Judith N.; "A Software Engineering Methodology for Rule Based Systems"; **IEEE Transactions on Knowledge and Data Engineering**; Volume 2; Nº 2; Junho 1990, 171-189.

Jafar, Musa e Bahill, A. Terry; "Interactive Verification of Knowledge Based Systems"; **IEEE Expert**, Fevereiro 1993, 25-32.

Jain, Hemant K. e Chaturvedi, Alok R.; "Expert System Problem Selection: A Domain Characteristics Approach"; **Information Management**, Volume 17, 1987, 245-253.

Jao, C.S. e Hier, D.B.; "Applying Hypermedia and Expert System Technology to the Neurological Consultation" in **Computer-Based Medical Systems**, IEEE, Estados Unidos, Junho 1993.

Jones, C. B.; **"Systematic Software Development Using VDM"**, Prentice-Hall, 1990.

Jong, Lieuwe S. de; "Engineering of Expert Systems"; **Information and Software Technology**, Volume 30, Nº 7, Setembro 1988, 418-425.

Juristos, Natalia e Pazos, Juan, "Towards A Joint Life Cycle for Software and Knowledge Engineering" in **Knowledge Oriented Software Design,** José Cuena (ed.), North Holland, 1994.

Keller, Robert; **"Expert Systems Technology: Development and Application"**; Prentice-Hall Inc; New Jersey, 1987.

Keyes, J.; "Why Expert Systems Fail"; **AI Expert**; Novembro, 1989, 50-53.

Kim P.T.H.; "FDA and the Regulation of Medical Software" in **Computer-Based Medical Systems**, IEEE, Estados Unidos, Junho 1993.

Kiper, James D.; "Structural Testing of Rule-Based Expert Systems"; **ACM Transactions on Software Engineering**, Volume 1, Nº 2, Abril 1992, 168-187.

Knight,J.; Myers,E.A. Improved Inspection Technique; **Communications of ACM**, novembro 1993.

Krisnamurthy, C., Padalkar, S., Sztipanovits, J. e Purves, R. S.; "Methodology for Testing and Validation Knowledge Bases"; **Proceedings of the Third Conference on Artificial Intelligence for Space Applications**; Novembro 1987, 21-32.

Leão, Beatriz de F.; **"Construção da Base de Conhecimento de um Sistema Especialista de Apoio ao Diagnóstico de Cardiopatias Congênitas"**; Tese de Doutorado, Escola Paulista de Medicina; 1988.

Leão, Beatriz de F. e Reategui, R.B.; "A Hibrid Conectionist Expert System to Solve Classification Problems "; **Computer in Cardiology**; IEEE Computer Society Press; Londres, setembro de 1993.

Leite, Julio C. S. P.; Validação de Requisitos: O uso de Pontos de Vista; **RBC - Revista Brasileira de Computação**; Outubro/Dezembro, 1990, 39-52.

Laufmann, S. C., DeVaney, D. M. e Whiting, M. A.; "A Methodology for Evaluating Potential KBS Applications"; **IEEE Expert**, Dezembro de 1990, 43-61.

Lucena, Carlos; **Inteligência Artificial e Engenharia de Software**; Rio de Janeiro, Jorge Zahar Editor Ltda, 1987.

Luger, George e Stubblefield, William A.; **"Artificial Intelligence and Design of Expert Systems"**; Benjamim Cummings Publishing Company, 1989.

Marcus, S.; "Salt: A Knowledge Acquisition Tools that Checks and Helps Test a Knowledge Base"; **AAAI Workshop on Validation and Verification of Expert Systems**, Agosto 1988.

Martin, James; **"Information Engineering"**, Prentice-Hall, 1990.

Martin, James; **"Engenharia da Informação: Introdução"**, Editora Campus, 1991.

Martin, James e Odell, James J., **"Object-Oriented Analysis & Design"**, Prentice-Hall, 1992.

Martins,A.V.;Rocha,A.R.da "Requisitos de Métricas para Gerenciamento da Qualidade do Processo de Desenvolvimento de Software"; **Workshop Qualidade de Software,** Curitiba, PR, outubro 1994.

Massolar, Jobson; **"Modelagem do Processo de Desenvolvimento",** Tese M.Sc., COPPE/UFRJ; dez 1993.

Matrey, William; "Expert Systems and Tools: Myths and Realities"; **IEEE Expert**, Fevereiro de 1992, 4-11.

Matsumoto, S.; "ES/SDEM - Software Development Engineering Methodology Expert Systems"; **Future Generation Systems**, Fevereiro de 1992, Número 5, 1989, 33-39.

Mayhew, P.J., Worsley, C.J. e Dearnley P.A.; "Control of Software Prototyping Process: Change Classification Approach"; **Information and Software Technology**; Volume 31, Número 2; Março de 1989.

Mayhew, P.J. e Dearnley P.A.; "Organization and Management of Systems Prootyping"; **Information and Software Technology**; Volume 32, Número.4; Maio de 1990; 245-252.

McMenamim, Stephen e Palmer, John; **"Análise Esssencial de Sistemas"**; Editora McGraw Hill, 1991.

Mendonça, E.; Leão, B. e .Guazelli, A.; "HYCONES II: Uma Ferramenta para Construção de Sistemas Especialistas Conexionistas Híbridos", **Congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994.

Menezes, I. M. G.; **"ESTIME: Uma Ferramenta para Estimativa de Custos de Desenvolvimento de Software",** Tese M.Sc., COPPE/UFRJ, setembro de 1988.

Merlevede, P. e Vanthienen, J.; "A Structured Approach to Formalization and Validation of Knowledge", **IEEE/ACM Internanational Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 149-159.

Meyer, Marc H. e Curley, Kathleen F., "Putting Expert Systems Technology to Work"; **Sloan Management Review**, Winter 1991, 21-31.

Mihaguti, Elisa F.; **Sistemas Baseados em Conhecimento: Um Estudo Exploratório em Empresas Brasileiras;** Tese de Mestrado em Andamento, São Paulo, FGV-SP (previsão 1º semestre de 1995).

Monat, André S.; **Métodos de Raciocínio Impreciso para Sistemas Especialistas Baseados em Regras;** Tese de Mestrado, Rio de Janeiro, COPPE/UFRJ, 1989.

Moulin, Bernard; "Strategic Planning for Expert Systems"; **IEEE Expert**, Abril de 1990, 69-79.

Moura, Lygia Maria; **"Taxonomia de Ambientes de Desenvolvimento de Software",** Tese M.Sc., COPPE/UFRJ, maio 1992.

Munakata, Toshinori; "Practical AI: Where it's been, and Where it is now"; **IEEE Expert**, Fevereiro de 1993, 3-5.

Munakata, Toshinori; "Commercial and Industrial AI"; **Communications of the ACM**, Volume 37, Número 3, Março de 1994a, 23-25.

Munakata, Toshinori; "Fuzzy Systems: An Overview"; **Communications of the ACM**, Volume 37, Número 3, Março de 1994b, 69-76.

Musen, M. e Tu, S., "Problem-Solving Models for Generation of Task-Specific Knowledge-Acquisition Tools" in **Knowledge Oriented Software Design,** José Cuena (ed.), North Holland, 1994.

Mylopoulos, Jonh, Wang, Huaiqing e Kramer Bryan; "Knowbel: A Hybrid Tool for Building Expert Systems"; **IEEE Expert**, Fevereiro 1993, 17-24.

Naser, J. A.; "Nuclear Power Plant Expert System Verification and Validation"; **Proceedings of the Workshop on Validation and Verification of Expert Systems**; Agosto 1988, 1-18.

Nemecek, S. e Bemley, J.; "A Model for Estimating Cost of AI Software Development: What to Do If There Are No Lines of Code?"; **IEEE International Conference on Developing and Managing Intelligent Systems Projects**; Março 1993, 2-9.

O'Leary, D. E.; "Validation of Expert Systems - with Applications to Auditing and Accounting Expert Systems"; **Decision Sciences**, Volume 18, Nº 3, Summer 1987, 468-486.

O_2, Technology; "**The O_2 User Manual Version 4.3 Released"**; França; Julho de 1993.

Oliveira, K.M.; Werneck,V.M.B.; Rocha, A.R.C.da; "Definição dos Requisitos de Qualidade de um Sistema Especialista em Cardiologia", **Congresso Brasileiro de Informática em Saúde**; Porto Alegre, RS; outubro 1994a.

Oliveira,K.M; Werneck,V.M.; Rocha,A.R.C.da; "Avaliação da Qualidade de Sistemas Especialistas"; **Workshop Qualidade de Software**, Curitiba, PR, outubro 1994b.

Oliveira,Káthia Marçal; **Avaliação da Qualidade de Sistemas Especialistas**; Tese de Mestrado, Rio de Janeiro, COPPE/UFRJ, março1995.

Oliveira,Káthia M., Belchior, A., Araujo, R., Vasconcelos Jr., F. e Werneck, V. M.; **"KBRF: Um Sistema Baseado no Conhecimento para Avaliar o Uso da Tecnologia Baseada em Conhecimento";** Relatório Técnico, COPPE/UFRJ, Junho 1995.

Page-Jones, Meilir; **"The Practical Guide to Structured Systems Design"**; Yourdon Press, 1980.

Palermo, S.e Rocha, A.R.C. da ; "Software Quality Assurance in HEP; **Computer Physics Communications**, Volume 57; North Holland; 1989.

Park, E. K., Chae, K. e Kang, C. S.; "The Structured Prototyping Life Cycle Model for Systems Development Management", **IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 267-272.

Passos, M.C. e Rocha, A.R.C.da; "Um Assistente Baseado em Conhecimento para Apoio ao Planejamento da Qualidade de Projetos de Desenvolvimento de Software"; **Workshop Qualidade de Software**, Curitiba, PR, outubro 1994.

Passos, M.C.; "**Despeja: Sistema de Locação de Imóveis**"; Exame de Qualificação, 1994 (não publicado).

Patterson, Dan W.; **"Introduction to Artificial Intelligence and Expert Systems"**; London, Prentice Hall, 1990.

Perot, F. C.; "A Crystal Model of Knowledge Based Application Design", **IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 279-285.

Philip, G. C. e Hilbert, H. F.; "What's Happening with Expert Systems"; **AI Expert**; Novembro 1990, 57-59.

Prieto-Diaz, Ruben; **"A Software Classification Scheme"**, Tese D.Sc, University of California, Irvine, 1985.

Pressman, Roger; **Software Engineering: A Practitioner's Approach**; Singapore; McGraw-Hill International Editions.

Rabelo Jr., A. et alli; "Experiência na Definição, Uso e Avaliação do Processo de Desenvolvimento de Sistemas Especialistas em Cardiologia"; **Workshop Qualidade de Software**, Curitiba, PR, outubro 1994a.

Rabelo Jr., A. et alli; "**Avaliação do 1º Estágio do Projeto Sistemas Especialistas em Cardiologia**"; Publicações Técnicas, FBC, 1994b.

Rabelo Jr., A. et alli; "An Expert System for Diagnosis of Acute Myocardial Infarction", **ACM Symposium on Applied Computing - SAC'95**, Nashville, USA; Fevereiro, 1995.

Rewari, A., Swartwout, M. W. e Register M. S.; "THE CANASTA Experience: Key Management and Technical Decisions in the Hybrid Expert System Project"; IEEE/ACM **International Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 160-165.

Ribeiro, Carlos Alberto S.; **Um Sistema de Apoio a Avaliação de Custos de Software;** Tese de Mestrado, Rio de Janeiro, COPPE/UFRJ, 1989.

Rich, Elaine; **"Artificial Intelligence"**; New York, Mc Graw Hill, 3ª Edição, 1985.

Robinson, A. E.; "Current Ideas in Knowledge-Base Management Systems"; **Information and Technology**; Volume 32, Nº 4; Maio de 1990, 266-273.

Rocha,A.R.; "**Um Modelo para Avaliação da Qualidade de Especificações**"; Tese de Doutorado; PUC-RJ; 1983.

Rocha, Ana Regina; Aguir, Teresa e Blashek, José; **Ambientes para Desenvolvimento de Software: Definição de Termos**; Relatório Técnico ES-137/87; COPPE/UFRJ; 1987.

Rocha, Ana Regina; **Análise e Projeto Estruturado de Sistemas**; Rio de Janeiro; Editora Campus, Ltda.; 1987a.

Rocha, A.R.C. da e Souza, J.M.; "TABA: Uma Estação de Trabalho para o Engenheiro de Software", **Congresso Latino Americano de Informática, Buenos Aires**, setembro de 1988.

Rocha, A.R.C. da; Aguiar, T.C. Aguiar; Souza, J.M. de; "Taba: A Heuristic Workstation for Software development"; **COMPEURO 90**; Tel Aviv, Israel; maio 1990.

Rocha,A.R.C.da e Werneck, V.M.B., **Ambiente de Manutenção**; Documento Interno IBM/IBL; não publicado, 1993.

Rocha,A.R.C.da et alli, Sistema de Planejamento Integrado: Processo de Desenvolvimento de Software; Publicação Interna do Projeto SPI; outubro 1993

Rocha,A.R.C. at alli "Uma Experiência na Definição do Processo de Desenvolvimento e Avaliação de Software segundo as Normas ISO"; **Workshop Qualidade de Software**, Curitiba, PR, outubro 1994.

Rolston, David W.; **"Principles of Artificial Intelligence and Expert Systems Developement"**; New York, Mc Graw Hill, 1988.

Ross, Douglas e Schoman, Kenneth; "Structured Analysis for Requirements Definition"; **IEEE Transactions on Software Engineering**; Volume SE-3, Número 1, Janeiro de 1977, 6-15.

Ross, Douglas; "Applications and Extensions of SADT"; **IEEE Computer**; Volume 18, Número 4, Abril de 1985, 25-34.

Rubin, Stuart H.; "Machine Learning and Expert Systems, **IEEE Expert**, Junho de 1993, 32-37.

Rumbaugh, James et alii, **"Object-Oriented Modeling and Design"**, Prentice-Hall, 1991

Saake, G, Jungclaus, R. e Hartmann, T.; "Application Modelling in Heterogeneous Environment using Object-Oriented Specification Language"; **Proceedings of International Conference on Intelligent and Cooperative Information Systems**, IEEE Computer Society Press, Maio de 1993, 309-318.

Sardinha, Eliziane; **"Gerência de Projetos na Estação TABA"**, Tese M.Sc., USP-São Carlos, nov 1993.

Schaschinger, H.; "Expert Supported Object-Oriented Analysis in Knowledge Engineering"; **Proceedings of Fourth International Conference on Software Engineering and Knowledge-Based**, IEEE Computer Society Press, Junho 1992, 116-122.

Schreiber,G.(ed.); "**Knowledge Acquisition: The KADS Approach to Knowledge Engineering**"; Academic Press; 1992.

Schreiber,G.; "Operationalizing Models of Expertise" em "**KADS: A Principled Approach to Knowledge-Based System Development**"; Schreiber,G., Wielinga, B. e Breuker, J.(ed.); Academic Press; 1993.

Schreiber,G., Wielinga, B. e Breuker, J.(ed.); "**KADS: A Principled Approach to Knowledge-Based System Development**"; Academic Press; 1993.

Schreiber,G., Wielinga, B., Breuker, J. et alii; "CommonKADS: A Comprehensive Methodology for KBS Development"; **IEEE Expert**; Dezembro de 1994, 28-37.

Scott, A, Carlisle, Clayton, Jan E. e Gibson, Elizabeth L.; **"A Practical Guide to Knowledge Acquisition"**; Massachusetts, Addison Wesley.

Selfridge, Peter G. (Panel Chair), Fischer, P.G., Ferguson G., Smith D., Johnson, W. L. e Hoebel, L.; "Assessing KBSE Research: Issues in Goals, Metrics and Transferability"; **Proceedings KBSE-92: Knowledge-Based Software Engineering Conference**; Setembro 1992, 246-248.

Selfridge, Peter G.; "Knowledge-Based Software Engineering", **IEEE Expert**, Volume 32, Nº 4; Dezembro de 1992.

Shadbolt, Nigel, Motta, Enrico e Rouge, Alain; "Constructing Knowledge Based Systems", **IEEE Software**, Novembro 1993, 34-38.

Sierhuis, M.; "An Object-Oriented Design Method for Knowledge Based Systems", **Proceedings of the IEEE/ACM International Conference on Developing and Managing Expert Systems Programs**, Setembro/Outubro 1991, 296-303.

Stylianou, G. et alii; "Selection Criteria for Expert Systems Shells: A Sociotechnical Framework"; **Communications of the ACM**, Volume 35, Número 10, Outubro de 1992, 30-48.

Tanik, Murat M. e Yeh, Raymond T.; "Expert Systems"; **IEEE Expert**, Novembro de 1988, 15-16.

Travassos, Guilherme Horta; **"O Modelo de Integração de Ferramentas da Estação TABA"**, Tese D.Sc., COPPE/UFRJ, março 1994.

Travassos,G. H. e Rocha, A.R.C.da; "Integração de Ferramentas na Estação TABA" **Congresso Latinoamericano de Informática**; Cidade do México, México; setembro 1994a.

Travassos, G.H. e Rocha,A.R.C.da "O Modelo de Integração de Ferramentas da Estação TABA"; **JAIIO**; Buenos Aires,Argentina; setembro 1994b.

Travassos,G.H.; Rocha, A.R.C.da; "Um Modelo para Construção e Integração de Ferramentas"; **VIII Simpósio Brasileiro de Engenharia de Software**; Curitiba, Paraná; outubro 1994c.

Travassos,G.H. e Werner, C.; da**; "Modelagem da Estação TABA: Especialização";** Relatório Técnico (em publicação), COPPE/UFRJ, 1995.

Turban, E "Managing Knowledge Acquisition from Multiple Experts"; **International Conference on Developing and Managing Expert Systems Programs**; Washington, USA; October 1991.

Vale, Marcus Antônio de Oliveira, **"PRODOC - Uma Ferramenta para Produzir Documentação de Produtos de Software"**, Tese M.Sc., COPPE/UFRJ, Março de 1988.

Velde, W. V.; "An Overview of CommonKADS" em **"The CommonKADS Library for Expertise Modeling"**; Breuker, J. e W. e Velde, Van de (ed.); Academic Press; 1994.

Vestli, M. et alii; "Modeling Control in Rule-Based Systems"; **IEEE Software**, Março de 1994, 77-81.

Vinson, Jonathan, Grantham, Stephen D. e Ungar, Lyle H.; "Automatic Rebuilding of Qualitative Models for Diagnosis"; **IEEE Expert**, Agosto 1992, 23-29.

Vob, A. e Karbah, W. "Implementing KADS Expertise Models with Model-K"; **IEEE Expert**, Agosto de 1993, 74-81.

Walker, Adrian, McCord, Michael, Sowa, John F. e Wilson, Water G.; **"A Logical Approch to Expert Systems and Natural Language Processing Knowledge Systemas and Prolog"**; Massachusetts, Addison Wesley, 1987.

Ward, P. T. e Mellor, S. J.; **"Structured Development for Real-Time Systems"**, Yourdon Press, 1985.

Warnier, J. D., **"LCP - Lógica de Construção de Programas"**, Editora Campus, 1986.

Waterman, Donald A.; **"A Guide to Expert Systems";** Massachusetts, Addison Wesley, 1986.

Weber, K. et alii; "**Qualidade e Produtividade em Software: Termo de Referência do Subprograma Saetorial da Qualidade e Produtividade em Software**"; Programa Brasileiro de Qualidade e Produtividade, QA&T, 1994.

Weitz, R. R.. e Meyer, A.; "Managing Expert Systems: A Framework and Case Study"; **Information & Management**, Volume 19, 1990, 115-131.

Weitzel, J.R. e Kerschberg, L.; "Developing Knowledge-based Systems: Reorganizing The System Development Life"; **Communications of the ACM**, Volume 32, Número 4, Abril de 1989, 482-488.

Werneck, Vera Maria B. et alii; "**Especificador de Ambientes da Estação TABA: Fase de Identificação**"; Relatório Técnico ES-202/89, COPPE/Sistemas, 1989.

Werneck, Vera Maria Benjamim; **"Taxonomia de Dominios de Aplicação",** MSc, COPPE/UFRJ, novembro 1990.

Werneck, Vera Maria B.; "Hiper-Aval: Uso de Hipertexto na Avaliação de Protótipos"; **XII Congresso de Metodologias en Ingenieria de Sistemas**, Chile, Julho de 1992, 94-98.

Werneck, Vera Maria B. e Carvalho, Luis Alfredo V. de; "**Sistemas Baseados em Conhecimento**"; Relatório Técnico ES-286/93, COPPE/Sistemas, 1993.

Werneck, Vera Maria B.; "**Uma Proposta para Ambiente de Desenvolvimento de Sistemas Baseados em Conhecimento**"; Relatório Técnico ES-300/94, COPPE/Sistemas, Março 1994a.

Werneck, Vera Maria B.; "**CRIANDO: Sistema do Estatuto da Criança e Adolescente**"; Exame de Qualificação, Março 1994b (não publicado).

Werneck, Vera Maria B. e Rocha, A.R.C.da; "**Processo de Desenvolvimento para Sistemas Baseados em Conhecimento**"; Relatório Técnico ES-299/94, COPPE/Sistemas; março, 1994.

Werneck,V.M.B., Oliveira,K.M.de; Rocha,A.R.C.da; **"Projeto Sistema Especialista em Cardiologia - Processo de Desenvolvimento de Software"**; Fundação Bahiana de Cardiologia, maio 1994a.

Werneck,V.M.B.; Oliveira,K.M.; Rocha,A.R.C.da "Uma Experiência na Definição do Processo de Desenvolvimento de Sistemas Especialistas em Cardiologia"; **Congresso Brasileiro de Informática em Saúde;** Porto Alegre,RS; outubro 1994b.

Wick, Michael R.; "Expert Systems Explanation in Retrospect: A Case Study in the Evolution of Expert System Explanation"; **J. Systems Software**, Volume 19, 1992, 159-169.

Wielinga, B. J., Breeqeg, B. e Breuker, J. A. ; "Knowledge Acquisition for Expert Systems"; **Proceedings of ACAI**, 1988.

Wielinga, B,. Schreiber,G. e Breuker, J.; "KBS Development through Knowledge Modelling" em **"Enhancing the Knowledge Engineering Process: Contributions from Esprit"**; North Holland; 1992.

Wirfs-Brooks, R.J. e Johnson, R.E.; "Surveying Current Research in Object-Oriented Design"; **Communications of the ACM**, Volume 33, Número 9, Setembro de 1990.

Woods, William A., "Important Issues in Knowledge Representation"; **Proceddings of IEEE**, Volume 74, Número 10, Outubro de 1986, 1322-1334.

Yen, John and Lee, Jonathan; "A Task-Based Methodology for Specifying Expert Systems; **IEEE Expert**, Fevereiro 1993, 8-15.

Yourdon, Edward, e Constantine, L.; **"Structured Design: Fundamentals of a Discipline of Computer Program and System Design"**, Yourdon Press, 1978.

Yourdon, Edward, **"Análise Estruturada Moderna"**, Editora Campus, 1990.

Zualkerman, I, Tsai, W.T. e Volovik, D.; "Expert Systems and Software Engineering: Ready for Mariage?"; **IEEE Expert**, Winter 1986, 25-30

Anexo I

Formulários para Avaliação do Processo de Desenvolvimento do SEC

Avaliação do Estágio de Demonstração da Viabilidade

(Desenvolvedores)

Neste momento em que concluimos a Versão X do Sistema, devemos avaliar a adequação do processo de desenvolvimento que utilizamos, com o objetivo de planejar melhor o desenvolvimento da próxima versão.

Assinale na escala o valor que lhe parece melhor representar o grau com que cada critério foi atingido.

Nome do Avaliador: _____

CRITÉRIOS DE AVALIAÇÃO	
1) As fases e atividades do modelo ciclo de vida adotado foram adequadas?	|___|___|___|___|___| 0 1 2 3 4
2) O volume de documentação foi adequado?	|___|___|___|___|___| 0 1 2 3 4
3) Os roteiros de documentação foram adequados?	|___|___|___|___|___| 0 1 2 3 4
4) Avalie as técnicas de aquisição do conhecimento considerando sua produtividade: • Entrevistas	|___|___|___|___|___| 0 1 2 3 4
• Caso Típico	|___|___|___|___|___| 0 1 2 3 4
. Protocolo Médico por Telefone	|___|___|___|___|___| 0 1 2 3 4
. Ordenação Conceitual	|___|___|___|___|___| 0 1 2 3 4
5) A técnica Delphi modificada foi adequada para resolução de conflitos de pontos de vista?	|___|___|___|___|___| 0 1 2 3 4

CRITÉRIOS DE AVALIAÇÃO	
6) O método KADS facilitou a representação do conhecimento no nível de complexidade necessário para o entendimento do problema?	0 1 2 3 4
7) O método KADS facilitou a identificação do conhecimento necessário para o projeto e implementação do sistema.?	0 1 2 3 4
8) O método KADS foi fácil de ser utilizado?	0 1 2 3 4
9) O método KADS foi fácil de ser aprendido pelos engenheiros do conhecimento?	0 1 2 3 4
10) A especificação do sistema, utilizando o método KADS, foi fácil de ser entendida pelos especialistas?	0 1 2 3 4
11) A especificação do sistema, utilizando o método KADS, foi fácil de ser validada?	0 1 2 3 4
12) O modelo físico foi útil na programação sistema?	0 1 2 3 4
13) O processo de avaliação da qualidade foi adequado?	0 1 2 3 4
14) O volume de avaliações foi adequado ?	0 1 2 3 4
15) O cronograma foi adequado?	0 1 2 3 4
16) A organização da equipe de desenvolvimento foi adequada?	0 1 2 3 4
17) A equipe de especialista foi adequada?	0 1 2 3 4

18) Que vantagens que você vê na utilização do método KADS?

19) Quais as desvantagens ?

20) Com relação ao processo de desenvolvimento da 2ª. versão:

	Manter	Rever
Fases e Atividades do Ciclo de Vida	⬭	⬭
Roteiros de Documentação	⬭	⬭
Técnicas de Aquisição do Conhecimento	⬭	⬭
Técnica de Resolução de Conflitos	⬭	⬭
Uso do Método KADS	⬭	⬭
Diagrama Estrutural do Conhecimento	⬭	⬭
Processo de Avaliação da Qualidade	⬭	⬭
Equipe de Desenvolvimento	⬭	⬭
Equipe de Especialistas	⬭	⬭
Equipe de Controle de Qualidade	⬭	⬭

21) Outras observações:

Avaliação do Estágio de Demonstração da Viabilidade
(Especialistas)

Neste momento em que concluimos a Versão X do Sistema, devemos avaliar a adequação do processo de desenvolvimento que utilizamos, com o objetivo de planejar melhor o desenvolvimento da próxima versão.

Assinale na escala o valor que lhe parece melhor representar o grau com que cada critério foi atingido.

Nome do Avaliador: _____

CRITÉRIOS DE AVALIAÇÃO	
1) O volume de documentação foi adequado?	0 1 2 3 4
2) Avalie as técnicas de aquisição do conhecimento segundo sua produtividade: • Entrevistas	0 1 2 3 4
• Caso Típico	0 1 2 3 4
• Protocolo Médico por Telefone	0 1 2 3 4
• Ordenação Conceitual	0 1 2 3 4
3) A técnica Delphi modificada foi adequada para resolução de conflitos de pontos de vista?	0 1 2 3 4
4) A especificação do sistema, utilizando o método KADS foi fácil de ser entendida?	0 1 2 3 4
5) A especificação do sistema, utilizando o método KADS foi fácil de ser validada?	0 1 2 3 4

CRITÉRIOS DE AVALIAÇÃO	
6) O processo de avaliação da qualidade foi adequado?	0 1 2 3 4
7) O volume de avaliações foi adequado ?	0 1 2 3 4
8) O cronograma foi adequado?	0 1 2 3 4
9) A organização da equipe de desenvolvimento foi adequada?	0 1 2 3 4
10) A equipe de especialista foi adequada?	0 1 2 3 4

11) Com relação ao processo de desenvolvimento da 2ª. versão:

	Manter	Rever
Fases e Atividades do Ciclo de Vida	⊂⊃	⊂⊃
Roteiros de Documentação	⊂⊃	⊂⊃
Técnicas de Aquisição do Conhecimento	⊂⊃	⊂⊃
Técnica de Resolução de Conflitos	⊂⊃	⊂⊃
Uso do Método KADS	⊂⊃	⊂⊃
Processo de Avaliação da Qualidade	⊂⊃	⊂⊃
Equipe de Desenvolvimento	⊂⊃	⊂⊃
Equipe de Especialistas	⊂⊃	⊂⊃
Equipe de Controle de Qualidade	⊂⊃	⊂⊃

12) Outras observações:

_____

_____

_____

_____

_____

_____

_____

Avaliação do Estágio de Demonstração da Viabilidade

(Coordenador do Projeto)

Neste momento em que concluimos a Versão X do Sistema, devemos avaliar a adequação do processo de desenvolvimento que utilizamos, com o objetivo de planejar melhor o desenvolvimento da próxima versão.

Assinale na escala o valor que lhe parece melhor representar o grau com que cada critério foi atingido.

Nome do Avaliador: Dr. Álvaro Rabelo Jr.

CRITÉRIOS DE AVALIAÇÃO	
1) As fases do modelo ciclo de vida adotado foram adequadas?	0 1 2 3 4
2) O volume de documentação foi adequado?	0 1 2 3 4
3) A especificação do sistema, utilizando o método KADS, foi fácil de ser entendida?	0 1 2 3 4
4) A especificação do sistema, utilizando o método KADS, foi fácil de ser validada?	0 1 2 3 4
5) O processo de avaliação da qualidade foi adequado ao desenvolvimento do sistema?	0 1 2 3 4
6) O volume de avaliações foi adequado ?	0 1 2 3 4
7) O cronograma foi adequado?	0 1 2 3 4
8) A organização da equipe de desenvolvimento foi adequada?	0 1 2 3 4
9) A equipe de especialista foi adequada?	0 1 2 3 4

10) Com relação ao processo de desenvolvimento da 2ª. versão:

	Manter	Rever
Fases e Atividades do Ciclo de Vida	⊂⊃	⊂⊃
Roteiros de Documentação	⊂⊃	⊂⊃
Uso do Método KADS	⊂⊃	⊂⊃
Processo de Avaliação da Qualidade	⊂⊃	⊂⊃
Equipe de Desenvolvimento	⊂⊃	⊂⊃
Equipe de Especialistas	⊂⊃	⊂⊃
Equipe de Controle de Qualidade	⊂⊃	⊂⊃

11) Outras observações:

Avaliação do Estágio de Demonstração da Viabilidade
(Outros Avaliadores)

Neste momento em que concluimos a Versão X do Sistema, devemos avaliar a adequação do processo de desenvolvimento que utilizamos, com o objetivo de planejar melhor o desenvolvimento da próxima versão.

Assinale na escala o valor que lhe parece melhor representar o grau com que cada critério foi atingido.

Nome do Avaliador: _____

CRITÉRIOS DE AVALIAÇÃO	
1) As fases e atividades do modelo ciclo de vida adotado foram adequadas?	|__|__|__|__|__| 0 1 2 3 4
2) O volume de documentação foi adequado?	|__|__|__|__|__| 0 1 2 3 4
3) Os roteiros de documentação foram adequados?	|__|__|__|__|__| 0 1 2 3 4
4) A especificação do sistema, utilizando o método KADS, foi fácil de ser entendida?	|__|__|__|__|__| 0 1 2 3 4
5) A especificação do sistema, utilizando o método KADS, foi fácil de ser validada?	|__|__|__|__|__| 0 1 2 3 4
6) O processo de avaliação da qualidade foi adequado?	|__|__|__|__|__| 0 1 2 3 4
7) O volume de avaliações foi adequado ?	|__|__|__|__|__| 0 1 2 3 4
8) O cronograma foi adequado?	|__|__|__|__|__| 0 1 2 3 4

9) Com relação ao processo de desenvolvimento da 2ª. versão:

	Manter	Rever
Fases e Atividades do Ciclo de Vida	⬭	⬭
Roteiros de Documentação	⬭	⬭
Uso do Método KADS	⬭	⬭
Processo de Avaliação da Qualidade	⬭	⬭

10) Outras observações:

Anexo II

Formulário para Identificação dos Critérios Relevantes para Definição do Ambiente de Programação

Formulário para Identificação dos Critérios Relevantes para Definição do Ambiente de Programação (Desenvolvedores)

Nome: _____

1) Critério: Interface com o Usuário Final

Característica	Impres-cindível	Impor-tante	Desejável	SemImpor-tância
Facilidade de explicação *(explicação clara das conclusões através de gráficos ou simples descrição do caminho que determinou a conclusão)*				
Documentação *(manual de ajuda e detalhamento de funções)*				
Tutorial *(ajuda direcionada)*				
Facilidades de Interface *(uso de janelas, mouse, menus desdobráveis, gráficos, tabelas, cores, modos de seleção do usuário (iniciante/ avançado), etc)*				
Facilidade de Armazenamento de Casos *(capacidade de guardar casos anteriores para poder consultá-los e/ou modificá-los)*				
Facilidade de Alteração de Casos *(capacidade de modificar um caso durante uma mesma execução do sistema)*				

2) Critério: Interface com o Desenvolvedor

Característica	Impres-cindível	Impor-tante	Desejável	SemImpor-tância
Ferramentas de edição/depuracão • editores internos com edição e execução interativa				
• facilidade de documentação *(permite o desenvolvimento de textos de ajuda e outras documentações junto com o código)*				
• "trace" *(característica de rastreamento para depuração)*				
• consulta às regras em tempo de execução *(esquema para mostrar estado das regras durante execução, como por exemplo codificação de cores)*				
Facilidade de Explicação • "how" (como) *(permite interrogar como chegou às conclusões)*				
• "what" (o que) *(permite interrogar o significado das questões perguntadas)*				
• "why"(porquê) *(permite interrogar o porquê das respostas dadas)*				
• formato livre *(permite ao desenvolvedor projetar explicações separadamente, usando um formato livre)*				
Projeto de telas *(facilidade de elaboração de telas amigáveis e de fácil entendimento)*				
Gráficos *(permitir capturar gráficos e textos para ligar à base de conhecimento e mostrar durante a execução do sistema)*				
Ambiente integrado *(possibilidade de ter mais de um processo, em janelas, na mesma tela, tais como, editor, depurador, e facilidade de explicação,...)*				
Gerador de Código *(poder traduzir para linguagens convencionais (como C, Pascal e Fortran) possibilitando, assim, portatilidade, embutibilidade e compatibilidade)*				

3) Critério: Interface do Sistema

Característica	Impres-cindível	Impor-tante	Desejável	SemImpor-tância
Hardware • uso em diferentes plataformas				
• uso em microcomputadores				
• suporte para multiprocessador				
• suporte para multiusuário				
• acesso a hardware especial				
Linguagem de Implementação • ser compilável				
• portatilidade *(capacidade de operar eficientemente em qualquer ambiente de HW e SW, acessando informações e aplicações existentes)*				
• embutibilidade *(capacidade do componente basedo em conhecimento ser construído dentro de uma aplicação convencional)*				
• compatibilidade com sistemas existentes *(capacidade de coexistir, cooperar e comunicar com outros programas, banco de dados e redes)*				
Segurança • evitar o acesso indevido do usuário				
• proteção do código				

4) Critério: Base de Conhecimentos

Característica	Impres-cindível	Impor-tante	Desejável	SemImpor-tância
Técnica de representação do conhecimento • técnicas simples de representação				
• várias possibilidades de representações alternadas				
• múltiplas representações em um mesmo sistema				
• Indução *(capacidade de gerar árvore de decisão e/ou conjunto de regras a partir de um conjunto de exemplos que demonstre o processo de decisão do especialista)*				

Característica	Lista de Técnicas Possíveis por Ordem de Adequabilidade ao Domínio do Problema
Tipo de Estrutura de Representação *(tipos de representações que se adaptam a esse tipo de problema: regras, "frames", "scripts", redes semânticas, objetos, lógica formal,...)*	

5) Critério: Máquina de Inferência

Característica	Imprescin-dível	Importante	Desejável	Sem importância
Mecanismo de Incerteza *(capacidade de trabalhar com incerteza)*				

Característica	Lista de Técnicas Possíveis por Ordem de Adequabilidade ao Domínio do Problema
Mecanismo de Inferência *("backward " ou "forward chaining", mecanismo híbrido ("forward chaining" para chegar às conclusões e "backward chaining" para confirmar), sistemas monotônicos ou sistemas não-monotônicos (permite que os fatos sejam modificados depois de terem sido estabelecidos))*	

5) Critério: Máquina de Inferência (cont.)

Característica	Lista de Técnicas Possíveis por Ordem de Adequabilidade ao Domínio do Problema
Estratégia de Busca *(técnica de navegação pela base de conhecimentos por busca exaustiva, por backtracking, por busca em largura,por busca em profundidade, etc)*	
Resolução de Conflito *(estratégia para determinar a prioridade das regras a serem executadas para uma situação. Geralmente, a prioridade é determinada pela regra antecedente ou consequente)*	
Mecanismo de Incerteza *(tipos de tratamento com incerteza: raciocínio por padrão, lógica nebulosa, lógica modal, fatores de certeza, probabilidade,...)*	

6) Critério: Interface de Dados

Característica	Imprescin-dível	Importan-te	Desejável	Sem importância
Acesso a linguagem usada *(acesso a linguagem usda para construção da própria ferramenta sem necessidade de um compilador a parte)*				
Acesso a outras linguagens *(permite chamadas a rotinas escritas em linguagens de terceira ou quarta geração)*				
Interface com banco de dados *(capacidade de acessar dados localizados em ambientes de banco de dados, internos ou externos à ferramenta)*				

7) Critério: Custo

Característica	Imprescin-dível	Importante	Desejável	Sem importância
Acesso a "Upgrades" *(possibilidade e direito a futuras melhorias da "shell")*				

Característica	Lista de Técnicas Possíveis por Ordem de Adequabilidade ao Domínio do Problema
Hardware *(tipo de hardware para operação do sistema)*	

8) Critério: Vendedor

Característica	Imprescin-dível	Importan-te	Desejável	Sem importância
Experiência *(tempo de trabalho com este ambiente e aplicações/serviços desenvolvidos)*				
Treinamento *(oferece cursos)*				
Manutenção *(oferece manutenção do ambiente)*				
Suporte Técnico *(oferece suporte técnico)*				
"Upgrade" • Inovação *(capacidade de introduzir constantemente produtos inovadores no mercado)*				
• Novas Versões *(oferece melhorias constantes dos seus produtos já no mercado)*				

9) Outras características importantes não mencionadas:

_____

_____

_____

Análise das Ferramentas de Programação através dos Critérios e Características Identificados como Relevantes

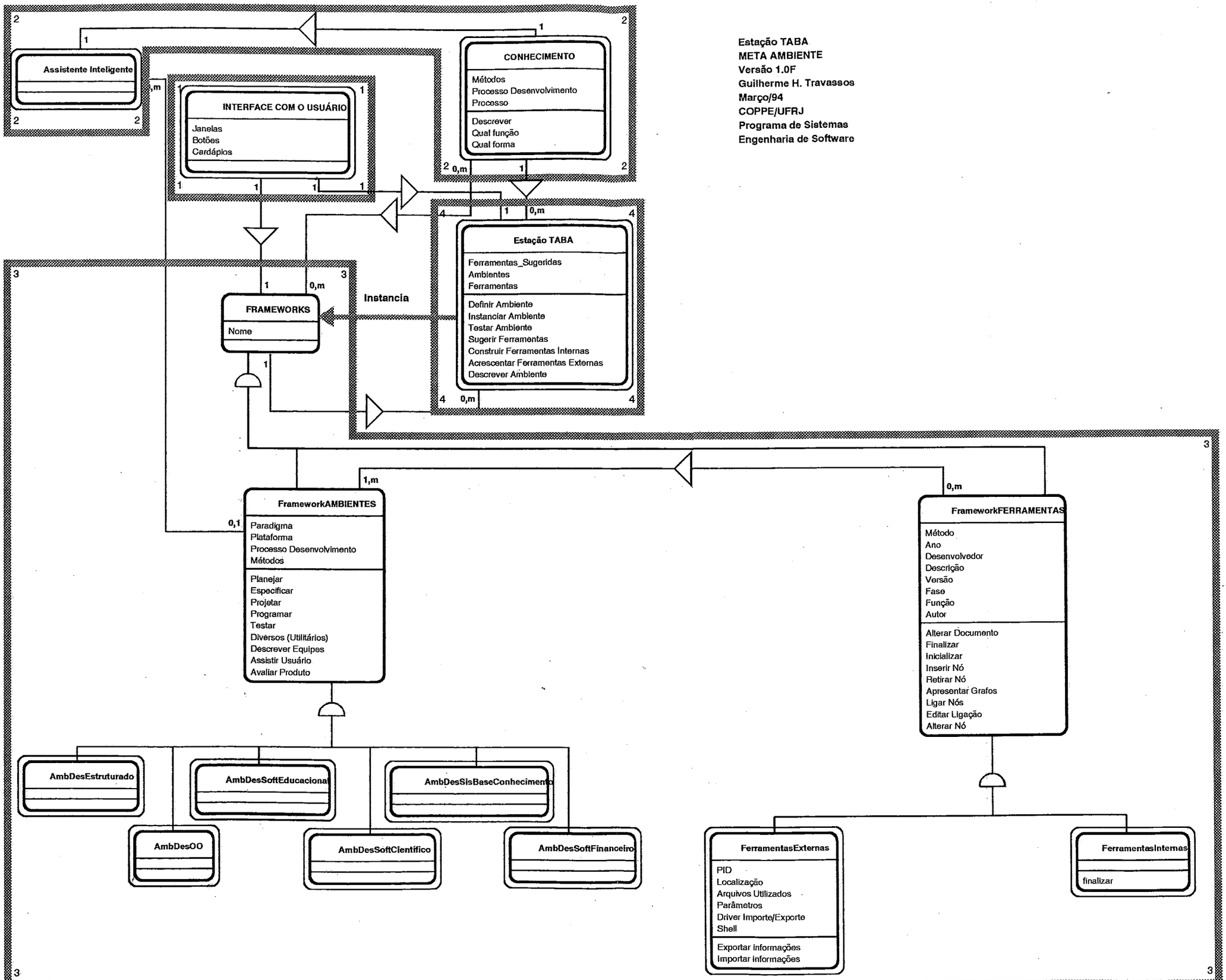
Data da Pesquisa: ____/____/____

Critérios	Características	FERRAMENTAS PESQUISADAS		
Interface com o Usuário Final	Facilidade de explicação			
	Documentação			
	Tutorial			
	Facilidades de Interface			
	Facilidade de Armaze-namento de Casos			
	Facilidade de Alteração de Casos			
Interface com o Desenvolvedor	Ferramentas de Edição / Depuração			
	Facilidade de Explicação			
	Projeto de Telas			
	Gráficos			
	Ambiente Integrado			
	Gerador de Código			
Interface do Sistema	Hardware Utilizado			
	Linguagem Compilável			
	Portatilidade			
	Embutibilidade			
	Compatibilidade com sistemas existentes			
	Segurança contra acesso indevido			
	Segurança de Proteção do Código			

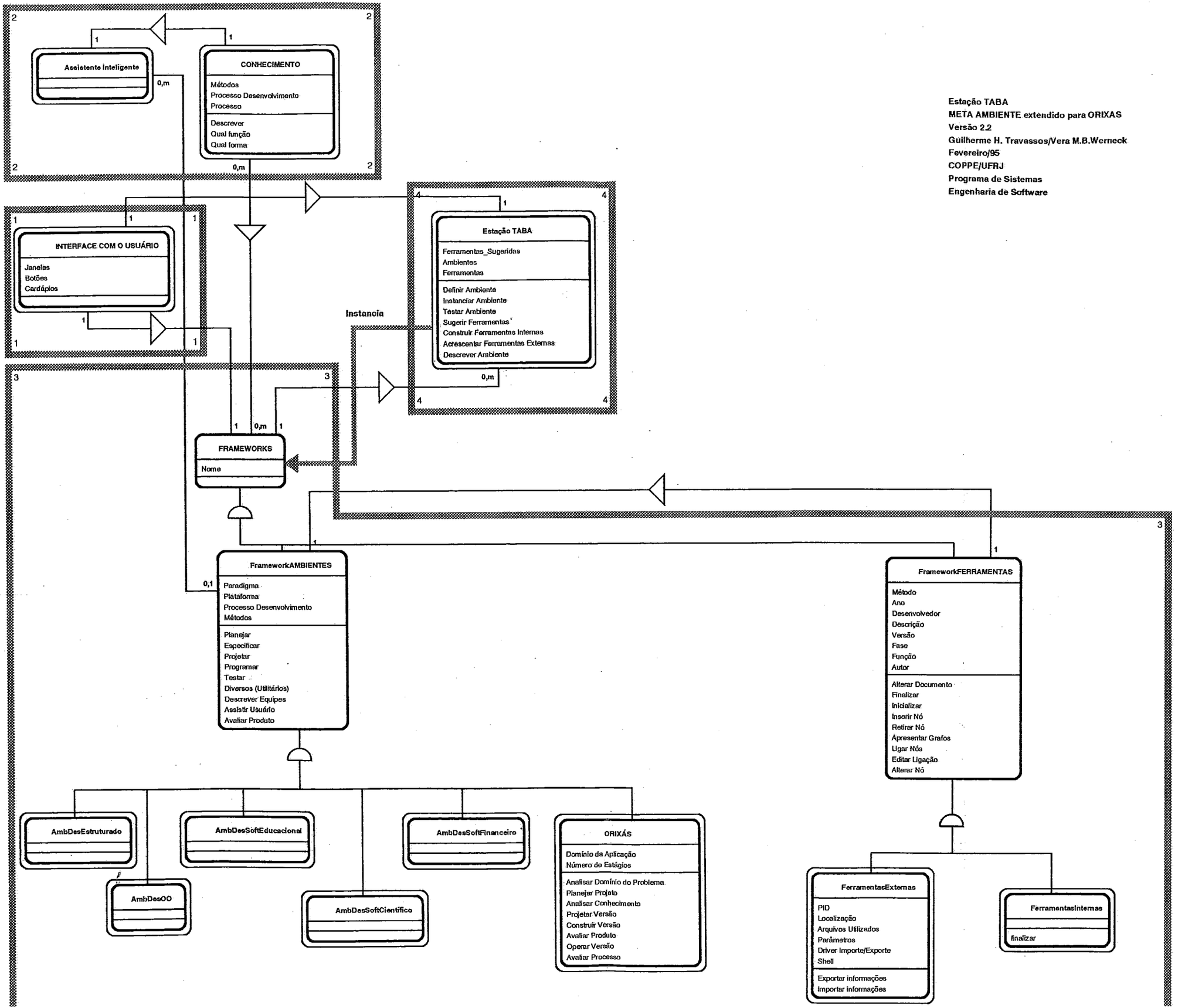
Critérios	Características	FERRAMENTAS PESQUISADAS		
Base de Conhecimentos	Técnicas Simples			
	Técnicas Alternadas			
	Técnicas Múltiplas			
	Tipos de Estruturas			
Máquina de Inferência	Mecanismo de Inferência			
	Estratégia de Busca			
	Resolução de Conflitos			
Interface de Dados	Acesso a linguagem usada			
	Acesso a outras linguagens			
	Interface com Banco de Dados			
Vendedor	Experiência			
	Treinamento			
	Manutenção			
	Suporte Técnico			
	Inovação			
	Novas Versões			
Custo	Preço			
	"Upgrades"			
	Hardware exigido			
	Suporte			
	Programa de treinamento			
	Instalação			
	Taxas de Consultoria			
	Cópias de "run-time"			

Anexo III

Modelagem da Estação TABA



Estação TABA
META AMBIENTE
Versão 1.0F
Guilherme H. Travassos
Março/94
COPPE/UFRJ
Programa de Sistemas
Engenharia de Software


Estação TABA
META AMBIENTE extendido para ORIXAS
Versão 2.2
Guilherme H. Travassos/Vera M.B.Werneck
Fevereiro/95
COPPE/UFRJ
Programa de Sistemas
Engenharia de Software
Asaistente Inteligente
CONHECIMENTO
Métodos
Processo Desenvolvimento
Processo
Descrever
Qual função
Qual forma
INTERFACE COM O USUÁRIO
Janelas
Botões
Cardápios
Estação TABA
Ferramentas_Sugeridas
Ambientes
Ferramentas
Definir Ambiente
Instanciar Ambiente
Testar Ambiente
Sugerir Ferramentas
Construir Ferramentas Internas
Acrescentar Ferramentas Externas
Descrever Ambiente
Instancia
FRAMEWORKS
Nome
FrameworkAMBIENTES
Paradigma
Plataforma
Processo Desenvolvimento
Métodos
Planejar
Especificar
Projetar
Programar
Testar
Diversos (Utilitários)
Descrever Equipes
Assistir Usuário
Avaliar Produto
FrameworkFERRAMENTAS
Método
Ano
Desenvolvedor
Descrição
Versão
Fase
Função
Autor
Alterar Documento
Finalizar
Inicializar
Inserir Nó
Retirar Nó
Apresentar Grafos
Ligar Nós
Editar Ligação
Alterar Nó
AmbDesEstruturado
AmbDesSoftEducacional
AmbDesSoftFinanceiro
ORIXÁS
Domínio da Aplicação
Número de Estágios
Analisar Domínio do Problema
Planejar Projeto
Analisar Conhecimento
Projetar Versão
Construir Versão
Avaliar Produto
Operar Versão
Avaliar Processo
AmbDesOO
AmbDesSoftCientifico
FerramentasExternas
PID
Localização
Arquivos Utilizados
Parâmetros
Driver Importe/Exporte
Shell
Exportar informações
Importar informações
FerramentasInternas
finalizar

Anexo IV

Definição e Extensão da *L*inguagem de *D*escrição e *M*anipulação de *M*étodos (LDMM)

Definição e Extensão da *L*inguagem de *D*escrição e *M*anipulação de *M*étodos (LDMM)

A definição da LDMM foi realizada em Travassos (1994), tomando-se por base métodos pertencentes a dois paradigmas distintos de desenvolvimento. Utilizou-se para sua construção os métodos SADT (Ross, 1985] e SSA (Gane, 1982), representando a classe de métodos pertencentes ao paradigma de desenvolvimento estruturado e os métodos OOA (Coad e Yourdon, 1992) e Booch (1991), representando a classe dos métodos orientados a objetos.

A partir do estudo destes métodos, foi possível identificar os papéis (funções) exercidos pelos elementos pertencentes a cada método em questão, que deveriam ser representados pela LDMM. Esta forma de representação permite que seja representada a semântica do documento gerado, e facilita, na medida do possível, o aproveitamento das informações geradas por uma ferramenta, num determinado momento do processo de desenvolvimento, pelo ambiente.

Foram identificadas, para os paradigmas estruturado e orientado a objetos, as seguintes funcionalidades para os elementos componentes dos métodos:

- ***DESCRITIVO***: elemento que permite a descrição de alguma informação textual complementar que auxilie a compreensão do documento.
- ***TRANSFORMADOR***: elemento responsável por receber um determinado conjunto de informações, transformá-las, de acordo com uma determinada lógica, e tornar estas informações transformadas disponíveis para outros elementos.
- ***FONTE***: elemento responsável por fornecer informações para o sistema. A informação fornecida por uma fonte é sempre recebida por um transformador.
- ***DESTINO***: elemento responsável por receber informações do sistema. A informação recebida por um destino é sempre fornecida por um transformador.

- ***ESTADO***: elemento que permite a representação de algum estado em que o outro elemento, ou o sistema, se encontra.
- ***ARMAZENADOR***: elemento responsável por guardar as informações relevantes ao sistema, e fornecê-las quando solicitado. Transformadores solicitam, aos armazenadores, a guarda ou recuperação das informações.
- ***ORGANIZADOR***: elemento que permite a organização de outros elementos. Pode ser utilizado para representar elementos que devem estar agrupados e apresentados de forma homogênea. Organizadores servem para agrupar encapsuladores e suas relações, salientando a característica, ou assunto principal, que eles representam.
- ***ENCAPSULADOR***: elemento que possui propriedades de encapsular informações e funcionalidades.
- ***HIERARQUIA***: elemento que permite mostrar a relação hierárquica existente entre elementos encapsuladores. Os encapsuladores envolvidos numa relação hierárquica não podem ser iguais.
- ***COMPOSIÇÃO***: elemento que permite relacionar elementos encapsuladores, demonstrando uma relação de composição (estruturação) entre eles.
- ***ESTÍMULO***: elemento que permite representar a ocorrência de algum evento (estímulo) que está acontecendo no sistema. Este estímulo pode ser representado por estímulos entre elementos encapsuladores, ou então, a partir de ocorrências de eventos que provoquem variações dos estados do sistema.
- ***DEPENDÊNCIA***: elemento que permite representar uma relação de dependência entre elementos encapsuladores.
- ***TRANSPORTADOR***: elemento responsável por transportar um conjunto de informações entre elementos transformadores, fonte, destino e armazenadores. Um transportador só pode transportar informações entre os seguintes elementos:

fonte → *transformador*;
transformador → *destino;*
transformador → *armazenador;*
armazenador → *transformador*;
transformador → *transformador*.

Esses elementos, no entanto, não são suficientes para representar semanticamente o *Modelo de Especialidade, Modelo Lógico* e o *Diagrama Estrutural da Base de Conhecimento (Modelo Físico)* do KADS-estendido. Assim sendo, foram identificadas as seguintes funcionalidades adicionais necessárias para definir os elementos componentes desses modelos:

- ***DECISOR*:** elemento responsável por decidir a partir de um determinado conjunto de informações, de acordo com uma determinada lógica, uma solução e tornar estas informações disponíveis para outros elementos.
- ***DEPENDÊNCIA CONJUNTO*:** elemento que permite representar uma relação de dependência entre encapsuladores e o conjunto de elementos de outro encapsulador.
- ***DEPENDÊNCIA LÓGICA*:** elemento que permite representar uma relação de dependência entre elementos encapsuladores com operadores lógicos.
- ***DEPENDÊNCIA LÓGICA CONJUNTO*:** elemento que permite representar uma relação de dependência entre elementos encapsuladores e o conjunto de elementos de outro encapsulador com operadores lógicos.
- ***ORDENADOR*:** elemento responsável por ordenar a execução de elementos decisores, mostrando o conjunto de elementos encapsuladores de entrada e saída. Um ordenador só pode ordenar informações entre os seguintes elementos:

  encapsulador → *decisor*;
  decisor → *encapsulador*;

A **LDMM** se compõe de dois blocos que são:

1) **Bloco de Definições:** permite a definição do conhecimento do método propriamente dito. É o responsável direto pela instanciação da classe Conhecimento;

2) **Bloco de Consultas**: permite a obtenção do conhecimento armazenado num objeto. É através dele que as ferramentas conseguem obter o conhecimento armazenado.

Na sintaxe da linguagem LDMM são encontradas as seguintes palavras reservadas:

Def_Método	Def_Doc	Def_Comp	Def_Função	Def_Forma
Sel_Método	Sel_Doc			
Qual_Forma	Qual_Tipo	Qual_Comp		
Importe				
Fim_Def_Método	Fim_Def_Comp	Fim_Sel_Método		Fim_Sel_Doc

A definição da sintaxe, feita em BNF, onde o sinal **?** representa o retorno de uma informação , é apresentada abaixo:

```
<Nome> ::= <Identificador> | <Identificador> <Nome>

<Identificador> ::= A | B | ... | Z

<Nome_Método> ::= <Nome>

<Nome_Documento> ::= <Nome>

<Nome_Componente> ::= <Nome>

<Método> ::=      Def_Método <Nome_Método>
                        <Definição_Documento>
                  Fim_Def_Método | &

<Definição_Documento> ::= Def_Doc <Nome_Documento>
                                <Definições>
                          Fim_Def_Doc | <Definição_Documento> | &

<Definições> ::=  Def_Comp <Nome_Componente>
                        <Define_função>
                        <Define_forma>
                  Fim_Def_Comp | <Definições> | &

<Define_função> ::=      Def_Função  <Função> | <Define_função> | &
```

<Define_forma> ::= Def_Forma <Forma> | &

<Forma> ::= Importe <Nome_Primitiva> <Lista_Parâmetros>

<Nome_Primitiva> ::= <Nome>

<Lista_Parâmetros> ::= <Parâmetro> | <Parâmetro> <Lista_Parâmetros>

<Parâmetro> ::= <Nome> | &

<Função> ::= DESCRITIVO | TRANSFORMADOR | FONTE | DESTINO | ESTADO | ARMAZENADOR | ORGANIZADOR | ENCAPSULADOR | HIERARQUIA | COMPOSIÇÃO | ESTÍMULO | DEPENDÊNCIA | TRANSPORTADOR | DECISOR | ORDENADOR | DEPENDÊNCIACONJUNTO | DEPENDÊNCIALÓGICA| DEPENDÊNCIALÓGICA CONJUNTO

<Seleções> ::= Sel_Método <Nome_Método>
<Consultas>
Fim_Sel_Método | <Seleções> | &

<Consultas> ::= Sel_Doc <Nome_Documento>
<Perguntas>
Fim_Sel_Doc | <Consultas> | &

<Perguntas>::= Qual_Forma {<Nome_Componente> | <Função>} ? <Forma> | Qual_Função {<Nome_Componente> | <Forma>} ? <Função> | Qual_Comp {<Função> | <Forma>} ? <Nome_Componente> | &

<Aplicação> ::= <Método> | <Consulta>

MODELO DE ESPECIALIDADE EM LDMM

Def_Método KADS-estendido
Def_Doc Estrutura-Domínio
Def_Comp Objeto
Def_Função *Encapsulador*
Def_Forma Importe DesRoundRect
Fim_Def_Comp
Def_Comp Sub-Tipo
Def_Função *Hierarquia*
Def_Forma Importe DesSeta
Fim_Def_Comp
Def_Comp RelConjunto
Def_Função ***DependênciaConjunto***[1]
Def_Forma Importe DesSetaLarga DesNome DesSeta
Fim_Def_Comp
Def_Comp Relação
Def_Função *Dependência*
Def_Forma Importe DesSeta DesNome DesSeta
Fim_Def_Comp
Def_Comp Expressão
Def_Função ***DependênciaLógica***
Def_Forma Importe DesSeta DesCaixa DesSeta
Fim_Def_Comp
Def_Comp ExpressãoConjunto
Def_Função ***DependênciaLógicaConjunto***
Def_Forma Importe DesSetaLarga DesCaixa DesSeta
Fim_Def_Comp
Fim_Def_Doc

[1]Em destaque os elementos novos definidos na LDMM

Def_Doc Estrutura-Inferência
    Def_Comp Meta-Classe
        Def_Função *Estado*
        Def_Forma Importe DesCaixa
    Fim_Def_Comp
    Def_Comp Inferência/Tarefa
        Def_Função ***Decisor***
        Def_Forma Importe DesElipse
    Fim_Def_Comp
    Def_Comp Fluxo
        Def_Função ***Ordenador***
        Def_Forma Importe DesSeta
    Fim_Def_Comp
Fim_Def_Doc
Def_Doc Estrutura-Tarefa
    Def_Comp Tarefa
        Def_Função ***Decisor***
        Def_Forma Descritivo
    Fim_Def_Comp
Fim_Def_Doc
Def_Doc Árvore-Tarefa
    Def_Comp Tarefa
        Def_Função ***Decisor***
        Def_Forma Descritivo
    Fim_Def_Comp
    Def_Comp Fluxo
        Def_Função ***Ordenador***
        Def_Forma Importe DesLinha
    Fim_Def_Comp
Fim_Def_Doc
Def_Doc Diagrama-Transição-Estado
    Def_Comp Estado
        Def_Função *Estado*
        Def_Forma Importe DesCaixa
    Fim_Def_Comp
    Def_Comp Mudanças-Estados
        Def_Função *Estímulo*
        Def_Forma Importe Deslinha DesNome
    Fim_Def_Comp
    Def_Comp Tarefa
        Def_Função ***Decisor***
        Def_Forma Descritivo
    Fim_Def_Comp
Fim_Def_Doc

MODELO LÓGICO EM LDMM

Def_Doc DHR
    Def_Comp Classe-Heurística
        Def_Função *Encapsulador*
        Def_Forma Importe DesCaixaColorida
    Fim_Def_Comp
    Def_Comp Classe-Entrada-Saída
        Def_Função *Encapsulador*
        Def_Forma Importe DesCaixa
    Fim_Def_Comp
    Def_Comp RelConjunto
        Def_Função ***DependênciaConjunto***
        Def_Forma Importe DesSetaLarga DesNome DesSeta
    Fim_Def_Comp
    Def_Comp Relação
        Def_Função *Dependência*
        Def_Forma Importe DesLinha DesNome DesSeta
    Fim_Def_Comp
    Def_Comp Expressão
        Def_Função ***DependênciaLógica***
        Def_Forma Importe DesCaixa DesSeta
    Fim_Def_Comp
    Def_Comp ExpressãoConjunto
        Def_Função ***DependênciaLógicaConjunto***
        Def_Forma Importe DesSetaLarga DesCaixa DesSeta
    Fim_Def_Comp
    Def_Comp Início
        Def_Função *Fonte*
        Def_Forma Importe CirculoI DesSeta
    Fim_Def_Comp
    Def_Comp Fim
        Def_Função *Destino*
        Def_Forma Importe CirculoI DesSeta
    Fim_Def_Comp
Fim_Def_Doc

Def_Doc Diagrama-Domínio-Problema
    Def_Comp Objeto
        Def_Função *Encapsulador*
        Def_Forma Importe DesRoundRect
    Fim_Def_Comp

Def_Comp Sub-Tipo
Def_Função *Hierarquia*
Def_Forma Importe DesSeta
Fim_Def_Comp
Def_Comp RelConjunto
Def_Função ***DependênciaConjunto***
Def_Forma Importe DesSetaLarga DesNome DesSeta
Fim_Def_Comp
Def_Comp Relação
Def_Função *Dependência*
Def_Forma Importe DesSeta DesNome DesSeta
Fim_Def_Comp
Def_Comp Expressão
Def_Função ***DependênciaLógica***
Def_Forma Importe DesSeta DesCaixa DesSeta
Fim_Def_Comp
Def_Comp ExpressãoConjunto
Def_Função ***DependênciaLógicaConjunto***
Def_Forma Importe DesSetaLarga DesCaixa DesSeta
Fim_Def_Comp
Def_Comp Classe-Heurística
Def_Função *Encapsulador*
Def_Forma Importe DesLinhaTracejada DesNome e/ou
Def_Forma Importe DesRoundRectCor
Fim_Def_Comp
Def_Comp Associação-Heurística
Def_Função ***DependênciaLógica***
Def_Forma Importe DesSeta DesCaixaCor DesSeta
Fim_Def_Comp
Def_Comp Classe-Interativa
Def_Função *Organizador*
Def_Forma Importe DesLinhaTracejada DesNome
Fim_Def_Comp
Def_Comp Classe-Início
Def_Função *Organizador*
Def_Forma Importe DesLinhaTracejada DesCirculoI
Fim_Def_Comp
Def_Comp Classe-Fim
Def_Função *Organizador*
Def_Forma Importe DesLinhaTracejada DesCirculoF
Fim_Def_Comp
Fim_Def_Doc

MODELO FÍSICO EM LDMM

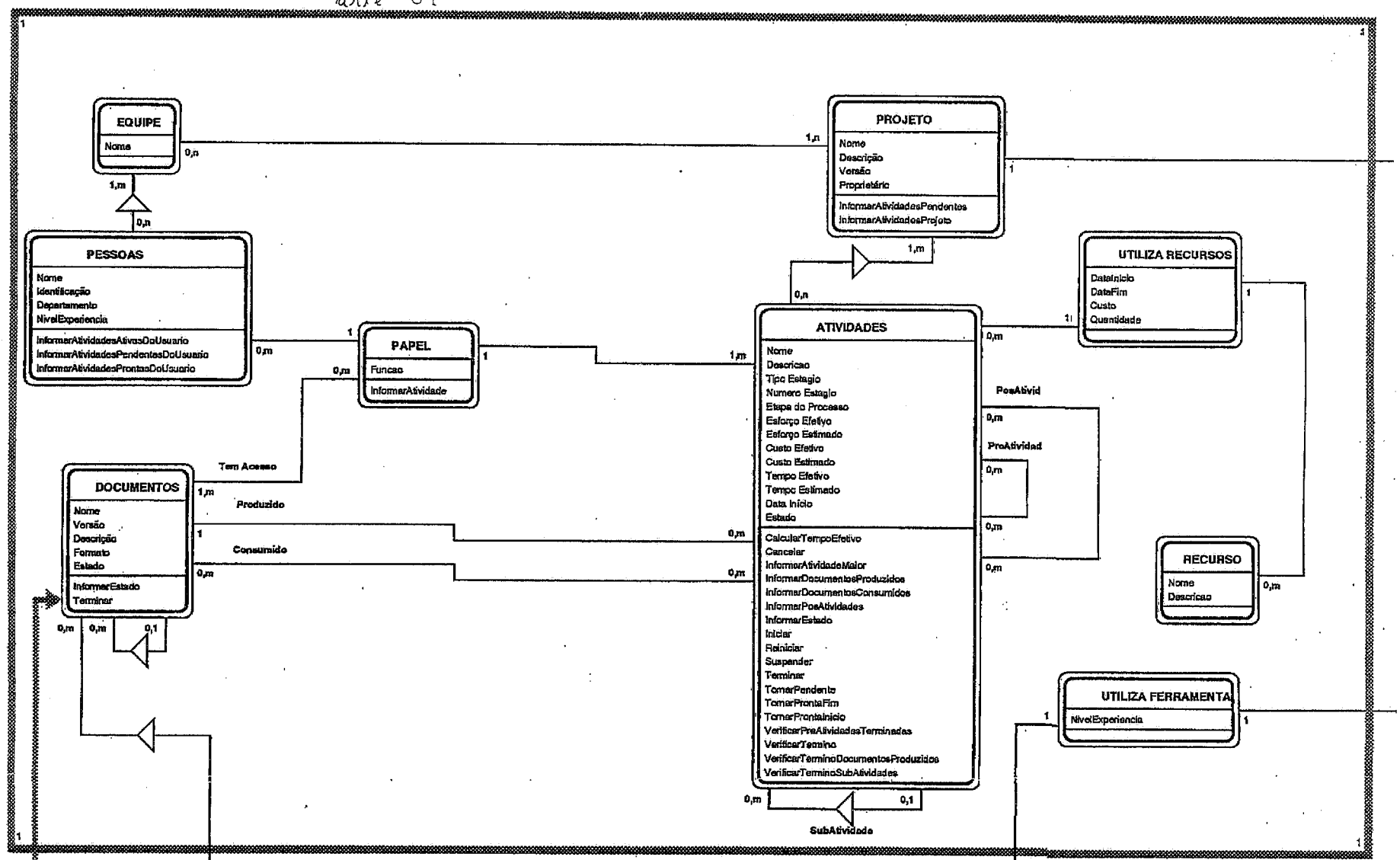
DIAGRAMA ESTRUTURAL DA BASE DE CONHECIMENTO

Def_Doc DEBC
Def_Comp Classe-Heurística
Def_Função *Encapsulador*
Def_Forma Importe DesCaixaColorida
Fim_Def_Comp
Def_Comp RelConjunto
Def_Função ***DependênciaConjunto***
Def_Forma Importe DesSetaLarga DesNome DesSeta
Fim_Def_Comp
Def_Comp Relação
Def_Função *Dependência*
Def_Forma Importe DesLinha DesNome DesSeta
Fim_Def_Comp
Def_Comp Expressão
Def_Função ***DependênciaLógica***
Def_Forma Importe DesCaixa DesSeta
Fim_Def_Comp
Def_Comp ExpressãoConjunto
Def_Função ***DependênciaLógicaConjunto***
Def_Forma Importe DesSetaLarga DesCaixa DesSeta
Fim_Def_Comp
Fim_Def_Doc
Fim_Def_Método

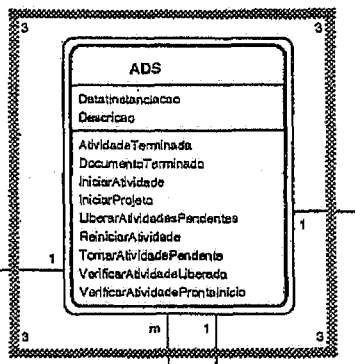
Anexo V

Modelagem da Estação TABA Especializada para o Ambiente ORIXÁS

Parte 01

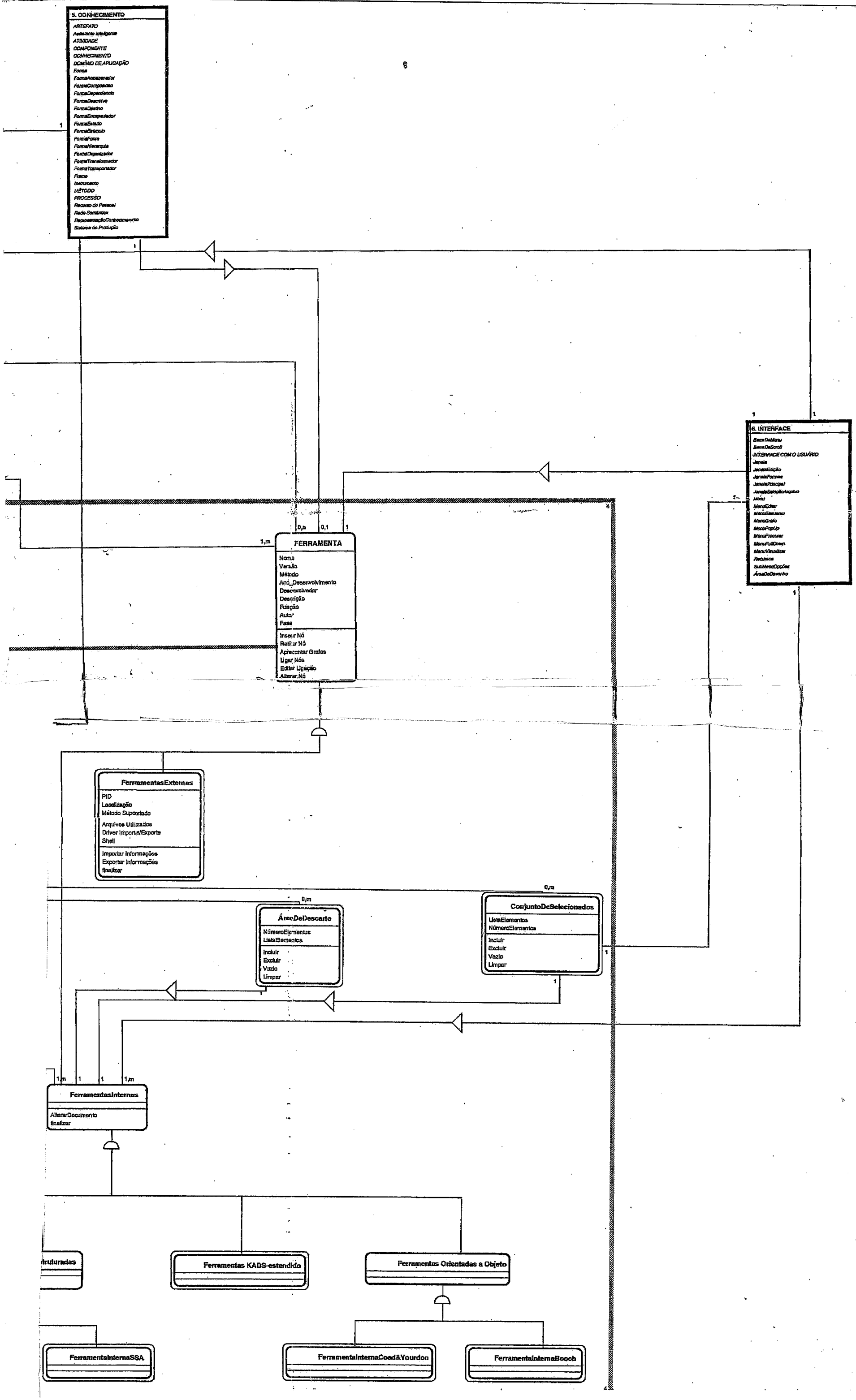

EQUIPE
Nome
PROJETO
Nome
Descrição
Versão
Proprietário
InformarAtividadesPendentes
InformarAtividadesProjeto
PESSOAS
Nome
Identificação
Departamento
NivelExperiencia
InformarAtividadesAtivasDoUsuario
InformarAtividadesPendentesDoUsuario
InformarAtividadesProntasDoUsuario
PAPEL
Funcao
InformarAtividade
UTILIZA RECURSOS
DataInicio
DataFim
Custo
Quantidade
ATIVIDADES
Nome
Descricao
Tipo Estagio
Numero Estagio
Etapa do Processo
Esforço Efetivo
Esforço Estimado
Custo Efetivo
Custo Estimado
Tempo Efetivo
Tempo Estimado
Data Início
Estado
CalcularTempoEfetivo
Cancelar
InformarAtividadeMaior
InformarDocumentosProduzidos
InformarDocumentosConsumidos
InformarPosAtividades
InformarEstado
Iniciar
Reiniciar
Suspender
Terminar
TornarPendente
TornarProntaFim
TornarProntaInicio
VerificarPreAtividadesTerminadas
VerificarTermino
VerificarTerminoDocumentosProduzidos
VerificarTerminoSubAtividades
PosAtivid
PreAtividad
SubAtividade
Tem Acesso
DOCUMENTOS
Nome
Versão
Descrição
Formato
Estado
InformarEstado
Terminar
Produzido
Consumido
RECURSO
Nome
Descricao
UTILIZA FERRAMENTA
NivelExperiencia
Modifica
GRAFOS
Método Original
Nome
Versão
IncluirElemento
ExcluirElemento
EncontraElemento
MostraGrafo
Modifica
ELEMENTOS
Nome
Descrição
PertenceaoGrafo
Mostra
NÓS
Localização
Identificação
ORGANIZADOR
Visibilidade
Descrição
ENCAPSULADOR
Serviços
Observações
DiagramaEstados
Atributos
Saída
Entrada
DECISOR
Método
Conhecimento de Suporte
DESCRITIVO
Texto
Observações
Editar Texto
ESTADO
Situação
ARMAZENADOR
Volume
Frequência de Atenção
Modelo de Armazenamento
Armazena
FONTE
Observações
Responsável
TRANSFORMADOR
DescriçãoLógica
Responsável
Ref_Físicas
Recebe
Entrega
DESTINO
Observações
Responsável
CODIGO-FONTE
Linguagem

Parte 02


ADS
DataInstanciacao
Descricao
AtividadeTerminada
DocumentoTerminado
IniciarAtividade
IniciarProjeto
LiberarAtividadesPendentes
ReiniciarAtividade
TomarAtividadePendente
VerificarAtividadeLiberada
VerificarAtividadeProntaInicio
3
3
1
1
3
m
1
3

LIGAÇÕES
Origem
Destino
Redirecionar
HIERARQUIA
2,m
COMPOSIÇÃO
Cardinalidade
2
DEPENDÊNCIA
Cardinalidade
Atributos
2
1,2
DependenciaConjunto
DependenciaLogica
Operador
ORDENADOR
Sentido
Repeticao
1
1,m
2
ESTÍMULO
Informações
Sincronismo
Frequência
1,m
1,2
Dependencia LogicaConjunto
0,1
0,1
0,1
1
TRANSPORTADOR
Volume
Observações
1,m
0,n
0,m
Estrutruras De Informação
Nome
Descrição
Sinônimos
Volume
0,m
0,n
1,n
Valores Atômicos
Nome
Tipo
Descrição
Possíveis Valores
Temporizador
IntervaloGravação
DataÚltimaGravação
HoraÚltimaGravação
GrafoMonitorado
HabilitadorDespertar
InicializarÚltimaGravação
SetarIntervaloGravação
Despertar
TrocarHabilitaçãoDespertar
1
1
1,m
1
1
Ferramenta
AlterarDocumento
finalizar
Ferramentas Estruturadas
ExplodirNó
FerramentaInternaSADT
Ferramentali

Parte 03


5. CONHECIMENTO
ARTEFATO
Assistente Inteligente
ATIVIDADE
COMPONENTE
CONHECIMENTO
DOMÍNIO DE APLICAÇÃO
Forma
FormaArmazenador
FormaComposicao
FormaDependencia
FormaDescritivo
FormaDestino
FormaEncapsulador
FormaEstado
FormaEstimulo
FormaFonte
FormaHierarquia
FormaOrganizador
FormaTransformador
FormaTransportador
Frame
Instrumento
MÉTODO
PROCESSO
Recurso de Pessoal
Rede Semântica
RepresentaçãoConhecimento
Sistema de Produção
6. INTERFACE
BarraDeMenu
BarraDeScroll
INTERFACE COM O USUÁRIO
Janela
JanelaEdição
JanelaFormas
JanelaPrincipal
JanelaSelecãoArquivo
Menu
MenuEditar
MenuElementos
MenuGrafo
MenuPopUp
MenuProcurar
MenuPullDown
MenuVisualizar
Rectangle
SubMenuOpções
ÁreaDeDesenho
FERRAMENTA
Nome
Versão
Método
And_Desenvolvimento
Desenvolvedor
Descrição
Função
Autor
Fase
Inserir Nó
Retirar Nó
Apresentar Grafos
Ligar Nós
Editar Ligação
Alterar Nó
FerramentasExternas
PID
Localização
Método Suportado
Arquivos Utilizados
Driver Importa/Exporta
Shell
Importar Informações
Exportar Informações
finalizar
ÁreaDeDescarte
NúmeroElementos
ListaElementos
Incluir
Excluir
Vazio
Limpar
ConjuntoDeSelecionados
ListaElementos
NúmeroElementos
Incluir
Excluir
Vazio
Limpar
FerramentasInternas
AlterarDocumento
finalizar
Ferramentas KADS-estendido
Ferramentas Orientadas a Objeto
FerramentaInternaSSA
FerramentaInternaCoad&Yourdon
FerramentaInternaBooch

Anexo VI

Modelo de Especialidade do KADS

Modelo de Especialidade do KADS

A metodologia KADS é o resultado de uma pesquisa desenvolvida em um projeto ESPRIT da comunidade européia. É uma proposta para o desenvolvimento e construção de sistemas baseados em conhecimento, prevendo a geração de vários modelos. (Schreiber, 1992), (Wielinga, Schreiber, e Breuker, 1992), (Schreiber, Wielinga e Breuker, 1993).

Entretanto, o modelo de especialidade é a atividade central do desenvolvimento de um sistema baseado em conhecimento e tem como objetivo especificar o conhecimento necessário para executar as tarefas associadas ao sistema. Este está baseado em um modelo de quatro camadas: domínio, inferência, tarefa e estratégia (Breuker e Wielinga, 1988), (Wielinga, 1988), (Sierhuis, 1991), (Schreiber, 1992), (Wielinga, Schreiber, e Breuker, 1992), (Schreiber, Wielinga e Breuker, 1993).

MODELO DE QUATRO CAMADAS

As quatro camadas do modelo têm interações limitadas e a Figura 1 mostra sua organização, seus elementos e relacionamentos entre si.

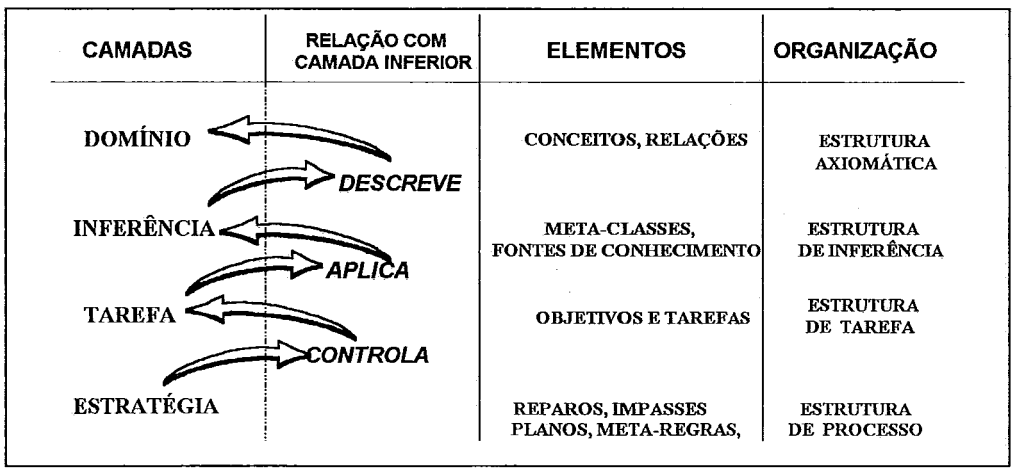


Figura 1- Camadas do Modelo Conceitual (Wielinga, 1988)

Camada de Domínio

Na camada de domínio é representado o conhecimento estático do domínio do problema, isto é, os conceitos e suas relações, independentes do raciocínio utilizado.

A análise da camada de domínio é importante e deve ser feita independente do processo de raciocínio pois esse não mostra claramente as relações entre os conceitos do domínio do problema.

Esta camada é definida utilizando a linguagem de descrição do domínio do problema cuja descrição é definida no final deste anexo.

Camada de Inferência

A camada de inferência contém o conhecimento necessário para inferir novos fatos a partir do conhecimento do domínio do problema. Essas inferências ou funções são denominadas *fontes de conhecimento*. Breuker e Wielinga (1988) definiram uma tipologia de fontes de conhecimento apresentada na Figura 2.

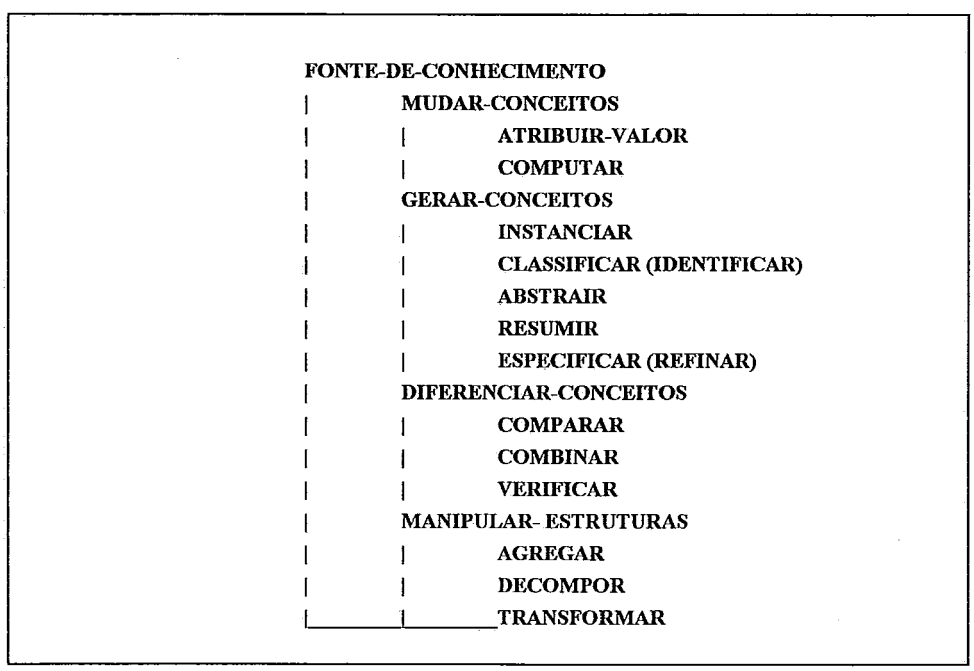


Figura 2 - Tipologia de Fontes de Conhecimento (Breuker e Wielienga, 1988), (Screiber, 1992)

As fontes de conhecimento podem ter várias entradas e possuem, normalmente, uma saída. As entradas são conceitos do domínio representados na camada de domínio. A saída dessas fontes é a instanciação de novos fatos e/ou mudanças na camada de domínio.

Assim as saídas e entradas da fonte do conhecimento são conceitos da camada do domínio e podem desempenhar papéis nesse processo. As meta-classes descrevem os papéis que os conceitos do domínio podem desempenhar no processo de raciocínio, sendo entradas e/ou saídas de uma fonte de conhecimento.

A vantagem de se dividir a inferência em fontes de conhecimento é permitir a decomposição funcional do comportamento do especialista ao solucionar o problema.

Alguns tipos de meta-classes (Figura 3) foram definidos por Breuker e Wielinga (1988).

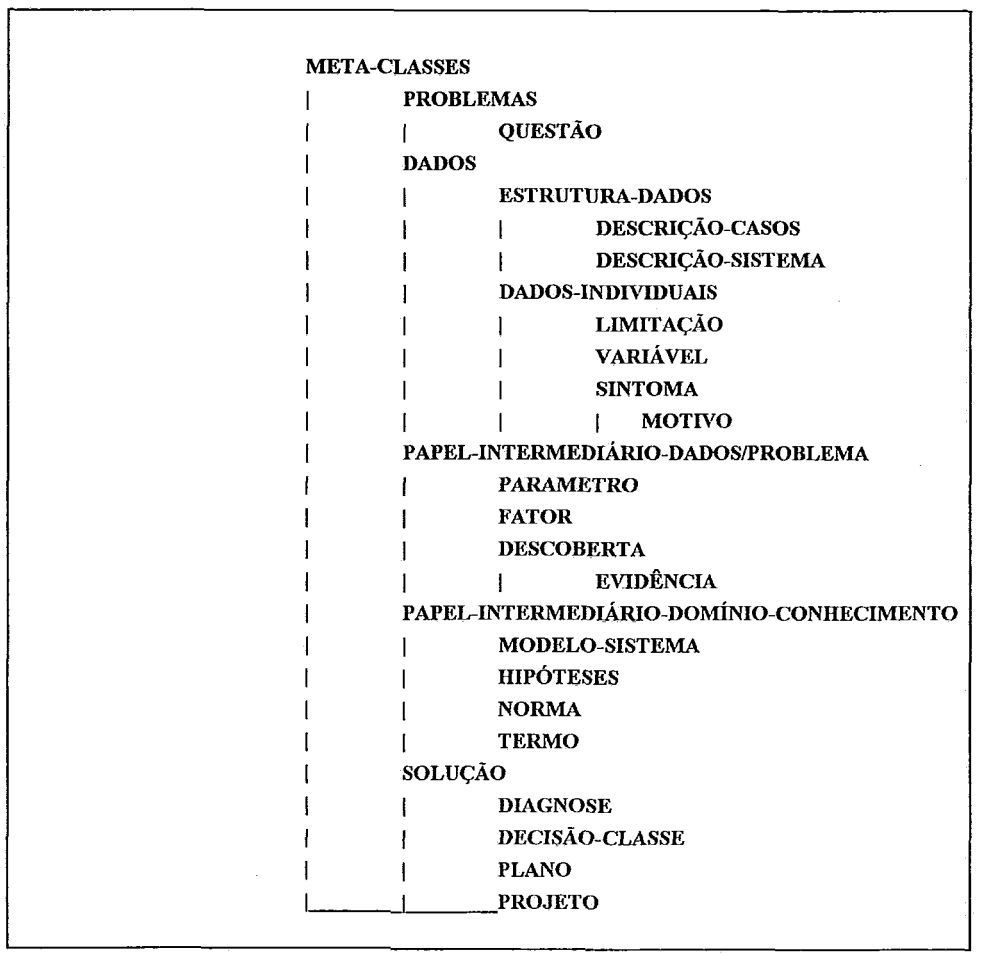


Figura 3 - Algumas Meta-Classes (Breuker e Wielienga, 1988)

Camada de Tarefas

A camada de tarefas representa a descrição de quando realizar inferências, descritas na camada de inferência.

Ao solucionar um problema, o especialista utiliza uma estrutura de controle que demonstra o seu conhecimento na solução do problema, pois ele deve ter o controle sobre o processo de realização da tarefa. Este nível de conhecimento das tarefas é representado na maioria das vezes, pelo paradigma de programação procedural, com estruturas de interação e comandos de seleção.

Breuker e Wielinga (1988) fornecem uma taxonomia de tarefas genéricas apresentada na figura 4.

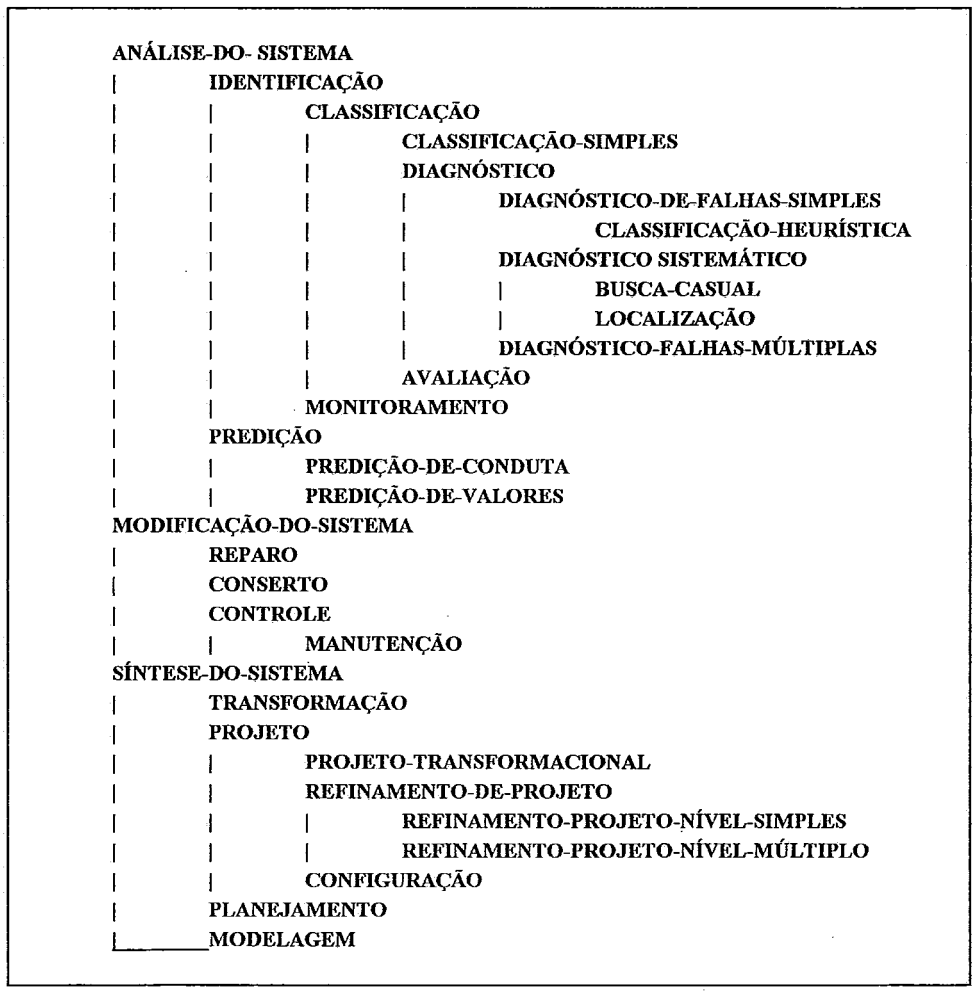


Figura 4 - Taxonomia de Tarefas Genéricas(Breuker e Wielienga, 1988)

Camada de Estratégia

A camada de estratégia representa o meta-conhecimento que o especialista tem sobre a estratégia usada para atacar um determinado problema, sendo que cada estratégia tem sua própria camada de tarefas.

Um mesmo problema pode ser resolvido utilizando diferentes estratégias e o meta-conhecimento é usado para se decidir as situações onde uma estratégia é melhor que outra. Esse tipo de conhecimento permite que sejam construídos sistemas baseados em conhecimento bem flexíveis.

CONSTRUÇÃO DO MODELO DE ESPECIALIDADE

O modelo especialidade do KADS é especificado na linguagem KCML ("KADS Conceptual Modelling Language"), que é uma simplificação da teoria de quatro camadas, tendo sua ênfase centrada nas estruturas de inferência e de tarefas.

No início da análise é usado um modelo de interpretação especificado na linguagem KCML, que define especificações gerais das estruturas de inferência e de tarefas.

Breuker e Wielinga (1988) fornecem além da taxonomia de tarefas genéricas, do conjunto de meta-classes e das fontes de conhecimento, uma biblioteca de modelos de interpretação. Em Hickman et alii (1989) encontram-se alguns desses modelos de interpretação.

Tarefas reais podem ser vistas como composições dinâmicas de tarefas genéricas, por isso o modelo conceitual pode ser construído através da combinação dos modelos de interpretação, provendo uma estrutura inicial de alto nível para a modelagem conceitual.

Ao se elicitar o conhecimento, identifica-se as tarefas do sistema através da taxonomia de tarefas genéricas para possibilitar a escolha do modelo de interpretação a ser utilizado. Este modelo deverá ser refinado para o domínio do problema tratado no modelo conceitual.

A figura 5 mostra o processo de construção do modelo e a figura 6 um exemplo de modelo de interpretação para a tarefa genérica de disgnóstico sistemático.

A linguagem KCML é um padrão para construção do modelo conceitual, não possuindo nenhum formalismo para a camada de estratégia. Para camada de domínio foi definida uma linguagem estruturada com descrição informal.

A fase de análise no KADS é suportada por um conjunto de ferramentas denominada SHELLEY (Anjewierden et alii, 1993). Esse sistema consiste num conjunto integrado de editores de manipulação direta para analisar os dados e construir o modelo conceitual. SHELLEY contém uma biblioteca de modelos de interpretação.

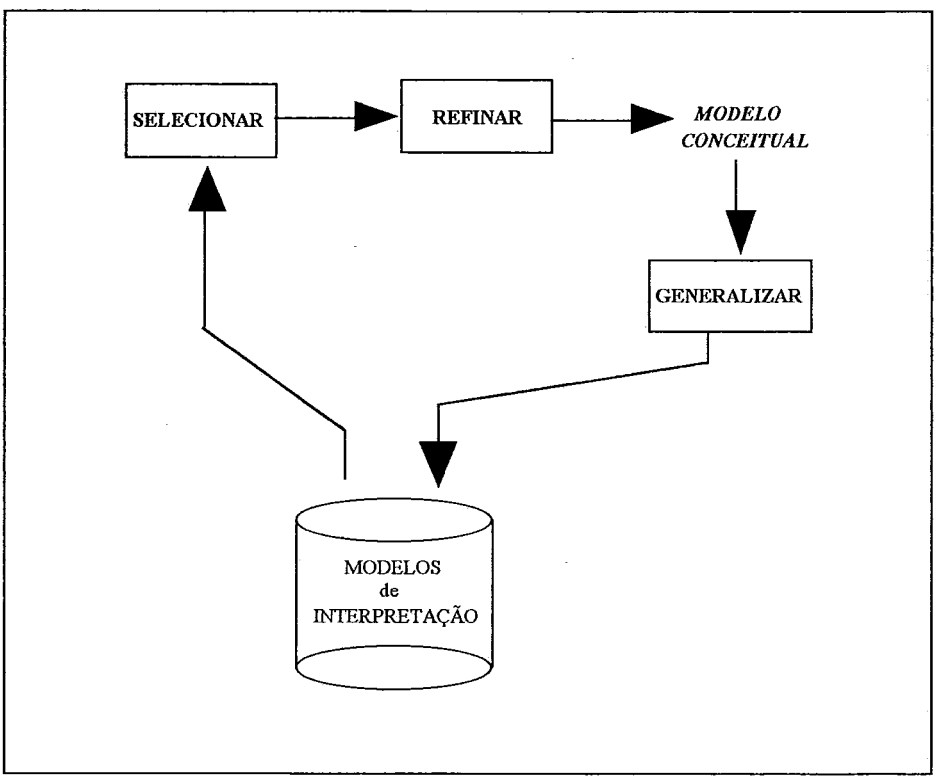


Figura 5 - Processo de Modelagem do Modelo de Especialidade do KADS

Apesar de vários estudos e aplicações mostrarem a utilidade desses modelos de interpretação, o modelo de especialidade não está totalmente desenvolvido e formalizado, não existindo modelos para todas as categorias de aplicação. Em Schreiber

et alii (1993) e Hickman et alii (1989) encontram-se alguns exemplos de modelagem de sistemas baseados em conhecimento utilizando o modelo de especialidade.

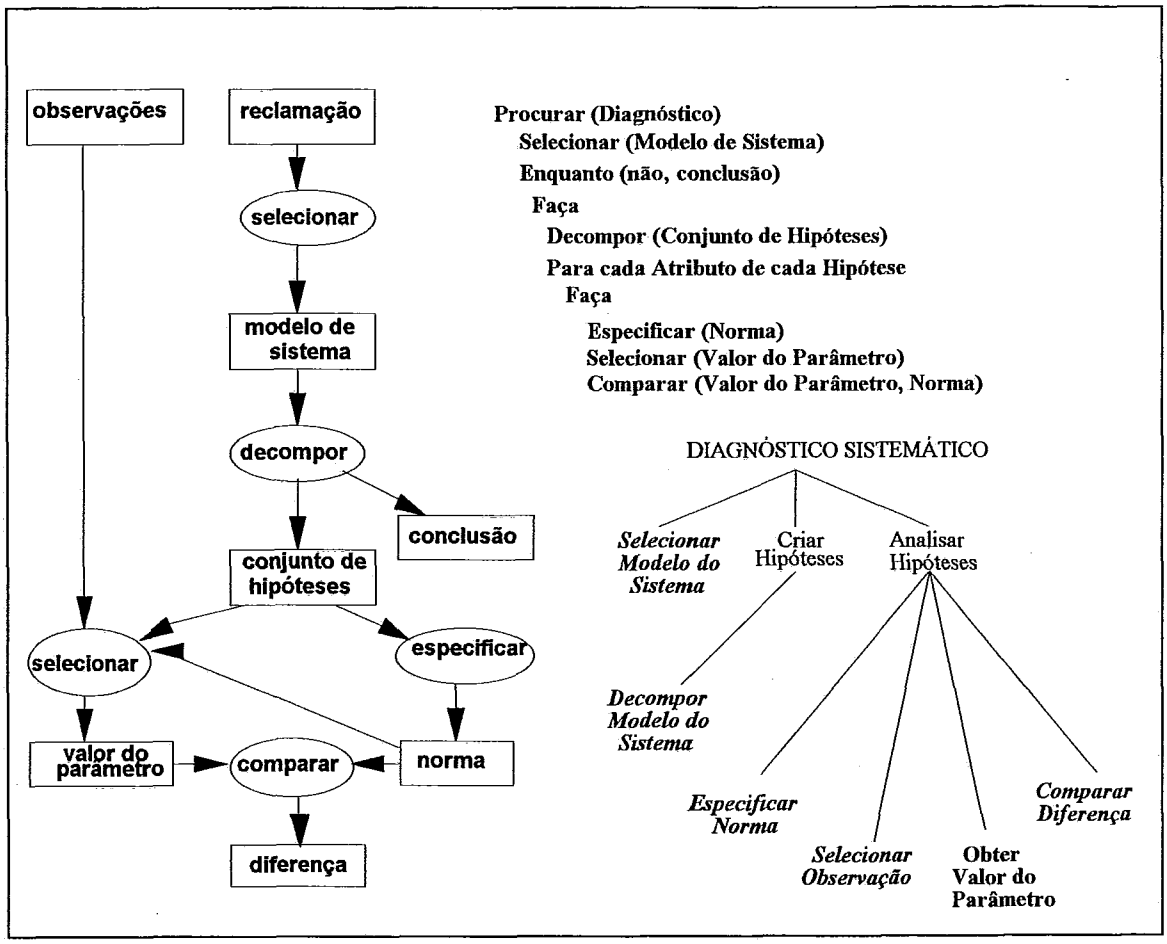


Figura 6 - Estruturas de Inferência e de Tarefas para o Dignóstico Sistemático (Breuker, 1988), (Schreiber et alii, 1993)

LINGUAGEM DE DESCRIÇÃO DO DOMÍNIO

A Linguagem de Definição de Domínio (DDL) foi desenvolvida com base no conhecimento do domínio em termos de conceitos de modelagem de dados, de orientação objetos e da linguagem Kl-ONE. Entretanto, nem todos esses conceitos são definidos graficamente e sim textualmente.

Esta linguagem expressa a conceitualização do domínio do problema na forma de uma teoria do domínio com as seguintes primitivas epistemológicas:

- *conceitos,* identificados pelo nome;
- *propriedades/valores:* conceitos possuem propriedades, que são definidas pelo nome e descrição de seus valores;
- *relações entre conceitos,* sendo bastante comum os tipos de relação de sub-classes (especialização) e de partes (agregação);
- *relações entre expressões de propriedades,* que são relações entre expresssões sobre determinados valores. Uma expressão é uma atribuição sobre valor(es) de uma propriedade de um conceito;
- *estrutura,* utilizadas para representar objetos complexos (vários objetos e relacionamentos).

DDL está dividida em 3 grupos de componentes de construção da modelagem: objetos intensionais, objetos extensionais e construções auxiliares.

Os objetos intensionais consistem nos elementos básicos da estrutura de dados representados por uma classe de objetos dos seguintes tipos:

- *conceito*: sinônimo de entidade ou classe;
- *conjunto*: representa uma construção de grupamento de objetos de um determinado tipo;
- *relação*: são as relações entre os diversos tipos: conceitos, instâncias, expressões, conjuntos;
- *estrutura*: agrega um número de partes onde essas podem ser conceitos, instâncias, conjuntos, relações, outras estruturas.

Os objetos extensionais são objetos definidos pela estrutura de objetos intensionais. Podem ser *instâncias* que correspondem a um elemento de conceito ou estrutura e *tuplas* que são elementos de uma relação.

As construções auxiliares são definidas nos objetos intensionais e correspondem a:

- *expressões:* relações com operadores lógicos, isto é, consistem de três partes, operando, operador lógico e propriedade de algum objeto;
- *subtipo*: hierarquias de objetos intensionais;
- *propriedade:* funções ou atributos que definem os objetos intensionais;
- *valores:* conjunto de valores possíveis das propriedades.

Esta linguagem possui uma representação gráfica e textual. A Figura 7 apresenta a definição gráfica da linguagem.

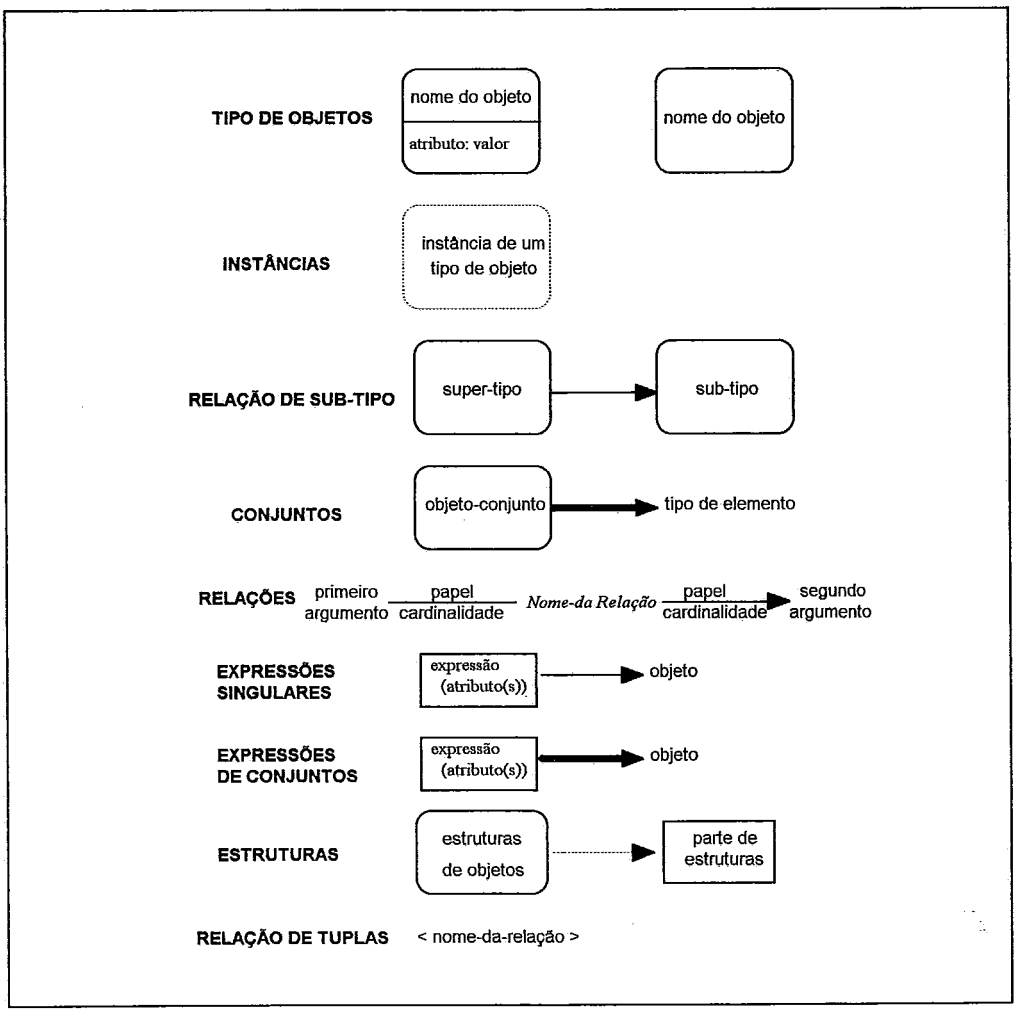


Figura 7 - Definição Gráfica da Linguagem DDL (Schreiber et alii, 1993)