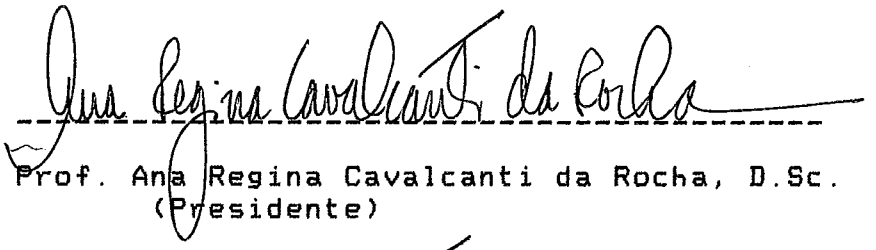
UM SISTEMA PARA REPRESENTAÇÃO DO CONHECIMENTO DE MÉTODOS DE DESENVOLVIMENTO DE SOFTWARE

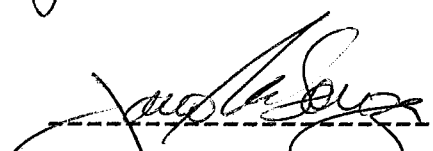
Marco Antonio Gomes Duarte

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

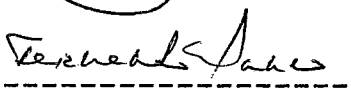
Aprovada por:



Prof. Ana Regina Cavalcanti da Rocha, D.Sc.
(Presidente)



Prof. Jano Moreira de Souza, Ph.D.



Prof. Fernando Silva Pereira Manso, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

JANEIRO DE 1991

DUARTE, MARCO ANTONIO GOMES

Um Sistema para representação do conhecimento de métodos de desenvolvimento de software [Rio de Janeiro] 1991

XII, 313 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1991)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Representação do Conhecimento I. COPPE/UFRJ

II. Título (série)

Para Beatriz, Daniela, Aloysio e Marcelo

AGRADECIMENTOS

À minha esposa e filhos pelo amor, apoio, incentivo e compreensão nos momentos difíceis e de privação de minha companhia.

Ao IBGE, pela oportunidade propiciada de realização deste curso e, ainda, aos apoios dos então Diretores de Informática Dr. Mário Aloysio Telles Ribeiro (em memória) e Dr. Paulo Sergio Braga Tafner.

O meu sincero agradecimento à Profa. Ana Regina C. Rocha pelo incentivo, orientação e sugestões, sem os quais, certamente, este trabalho não teria sido possível.

À Juan C. C. Pena, Guilherme H. Travassos e Teresa C. Aguiar pelo apoio na orientação e revisão deste trabalho.

Aos Profs. Jano M. Souza e Fernando S. P. Manso pela participação na Banca Examinadora.

Aos Professores, Funcionários e Colegas do Programa de Engenharia de Sistemas da COPPE com quem tive o privilégio de conviver.

Aos meus Pais, por me demonstrarem a razão da vida e da força de vontade.

À DEUS por tudo.

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

UM SISTEMA PARA REPRESENTAÇÃO DO CONHECIMENTO DE MÉTODOS DE DESENVOLVIMENTO DE SOFTWARE

Marco Antonio Gomes Duarte

Janeiro de 1991

Orientador : Ana Regina Cavalcanti da Rocha, D.Sc.

Programa: Engenharia de Sistemas e Computação

Este trabalho, integrante do projeto TABA, pretende contribuir com a busca de resultados que venham a aumentar a produtividade do processo de desenvolvimento de software e permitir o desenvolvimento de produtos de software com qualidade.

O TABA é um projeto, em desenvolvimento na COPPE/UFRJ, cujo objetivo é a construção de uma estação de trabalho configurável para desenvolvimento de software.

A partir de um estudo sobre representação do conhecimento, este trabalho apresenta a descrição de um sistema específico para representação do conhecimento de métodos de desenvolvimento de software. O sistema proposto foi desenvolvido baseado na utilização dos métodos de representação de Redes Semânticas, Quadros e Regras de Produção, sendo considerado para esta adoção híbrida, os tipos de conhecimentos adequados para o domínio tratado e a formação de uma base de conhecimentos para a manipulação do conhecimento armazenado.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M.Sc.).

A SYSTEM FOR KNOWLEDGE REPRESENTATION OF SOFTWARE DEVELOPMENT METHODS

Marco Antonio Gomes Duarte

January, 1991

Thesis Supervisor: Ana Regina Cavalcanti da Rocha

Department: Computer Systems Engineering

The purpose of this work, part of TABA project, is contributing with the search for results that increase the productivity of the software development process and the quality for the software products.

TABA, under development at COPPE/UFRJ, is a project which main objective is the building of a configurable workstation for software development.

Using as a starting point a study on knowledge representation, this work presents a description of a specific system for knowledge representation of software development methods. The proposed system was developed based on the using of the knowledge representation for Semantic Network, Frames and Production Rules and, for that hybrid implementation, it has been considered the types of knowledge suited for the explored domain and the set up of a knowledge base for the manipulation of the stored knowledge.

ÍNDICE

ÍNDICE DE FIGURAS

Capítulo I

INTRODUÇÃO

I.1 - EVOLUÇÃO HISTÓRICA.

É importante iniciar citando a evolução tecnológica dos computadores como fundamental, pois a cada dia a humanidade a usufrui e deve-se considerar privilegiada pelos serviços resultantes e essenciais dessa tecnologia. Isto significa dizer que além do advento dessa máquina - o computador - necessário se faz que ela produza resultados que venham a auxiliar as atividades diárias de pesquisadores, profissionais, estudantes, etc.

Embora alguns pesquisadores considerem o STONEHENGE o primeiro computador feito pelo homem, isto por volta de 2600 a.C. a 1700 a.C., foi na década de 60 que surgiu o primeiro computador programado, dando início assim, ao surgimento de áreas voltadas a fazer com que pudessemos obter, através do uso dos computadores, resultados úteis à humanidade.

Nesta oportunidade se faz necessário, entretanto, destacar os pontos mais relevantes na evolução da capacidade do homem para o desenvolvimento do ferramental que resultou nos computadores de hoje. São eles: o conceito de número; os primeiros métodos de cálculos (utilização dos dedos, o ábaco); os auxílios manuais nos cálculos escritos

(o método árabe, os logarítmos); os auxílios mecânicos para cálculos (a máquina de Pascal, os ossos de Napier, a régua de cálculo); auxílios mecânicos automáticos (máquina das diferenças, calculadoras mecânicas); automatismo completo (máquina analítica, máquinas tabuladoras); os primeiros computadores (MARKI, EDVAC, ENIAC etc). E, com relação aos pesquisadores que contribuíram de maneira significante para esta evolução, cita-se: Pascal, Napier, Leibnitz, Jacquard, Babbage, Aiken, Von Newmann e vários outros.

O rápido desenvolvimento da microeletrônica, permitindo a criação de computadores cada vez mais poderosos e a um custo decrescente, aliada ao avanço das telecomunicações, trouxe a chamada sociedade "informatizada", da qual somos contemporâneos.

O avanço tecnológico associado ao custo decrescente das máquinas incentiva cada vez mais a produção de sistemas de computação cuja aplicabilidade se presta a todas as áreas do conhecimento humano. Por outro lado, todas estas áreas demandam a concepção e o desenvolvimento de instrumentos baseados na mesma tecnologia a princípio desenvolvida para os computadores.

Com o advento dos computadores de terceira geração na década de 60 e a necessidade cada vez maior de técnicas e recursos para o desenvolvimento de programas, o surgimento de técnicas tais como, multi-programação e "time sharing", só foi possível com o desenvolvimento de recursos da tecnologia para sistemas interativos, multi-usuários, on-line e de tempo real.

Muitos destes sistemas tem sido construídos e liberados para uso. Dentre eles muitos apresentam problemas de custo excessivamente alto, demora na liberação do sistema, falhas de confiabilidade, ineficiência e problemas de aceitação dos sistemas pelo usuário.

À medida que os sistemas tornam-se maiores e mais complexos, torna-se aparente que a demanda de software está crescendo mais rapidamente do que a nossa habilidade em produzi-lo e mantê-lo.

Durante o decorrer dos últimos anos se comprovou, estatisticamente, que a maior parte do tempo usado pelas pessoas envolvidas em sistemas de software é ocupado em manutenção de sistemas já existentes, ao invés do desenvolvimento de novos sistemas numa razão de 30/70, 40/60 e até mesmo de 10/90, segundo estudos de (BOEHM-76).

A maioria dos problemas encontrados durante a manutenção de um sistema se deve a erros de projetos, os quais são muito mais difíceis de detectar e mais caros de corrigir do que os erros de implementação. Geralmente estes erros se devem ao entendimento incorreto da relação entre o sistema e seu ambiente e à má especificação dos requisitos do sistema, muitas vezes incorreta, inconsistente ou incompatível e confusa.

Com a finalidade de tratar esses tipos de problemas é que surgiu a área de tecnologia denominada de **ENGENHARIA DE SOFTWARE** .

I.2 - MOTIVAÇÃO.

Embora haja o reconhecimento dos esforços dedicados pela comunidade científica e pesquisadora da área de Informática, colocando disponíveis técnicas e recursos que objetivam atender às necessidades da tarefa de desenvolver sistemas, todo esse esforço não significa a eliminação de problemas nesse tipo de tarefa. Por exemplo, um tradicional problema vem resistindo a todo esse esforço, que é o das pessoas considerarem que a tarefa de desenvolver sistemas é uma atividade pessoal, sem que haja um planejamento e um método para o seu desenvolvimento, dificultando assim a manutenção pela própria pessoa que o desenvolveu e muito mais por pessoas não envolvidas com o sistema.

Por esta abordagem, relacionada ao desenvolvimento e manutenção de produtos de software em tempos específicos e dentro das estimativas de custo consideradas, a Engenharia de Software (ES) busca dentre seus objetivos a satisfação no trabalho das pessoas envolvidas.

Atualmente, não se pode ignorar o estágio alcançado pela Inteligência Artificial (IA), onde os campos de implementação estão cada vez mais se expandindo.

À medida que a pesquisa de IA progrediu e técnicas para tratar de quantidades maiores de conhecimentos do mundo foram desenvolvidas, conseguiu-se certos progressos nas áreas de jogos e provas de teoremas,

e novas áreas puderam ser razoavelmente exploradas. Elas incluem a percepção (visão e fala), compreensão de linguagem natural e resolução de problemas em domínios especializados, como a diagnose médica e a análise química (RICH-88).

(LEVINE-88) apresenta em seu livro que "um programa comum pode fazer tudo o que um programa de IA faz, mas não pode ser programado tão fácil ou rapidamente".

Esta característica de programas de IA ser desenvolvido com mais facilidade e rapidez, vêm de encontro com os interesses da ES. Assim, se forem consideradas as técnicas e recursos dessas áreas, resultados proveitosos poderão advir, vindo a contribuir com a solução de problemas que até hoje enfrentamos com o desenvolvimento de produtos de software.

Este trabalho, integrante do projeto TABA, pretende contribuir com a busca de resultados que venham a aumentar a produtividade do processo de desenvolvimento de software e permitir o desenvolvimento de produtos de software com qualidade.

O TABA é um projeto, em desenvolvimento na COPPE/UFRJ, cujo objetivo é a construção de uma estação de trabalho configurável para desenvolvimento de software.

A Estação TABA pode ser usada com quatro objetivos (ROCHA-89):

a) auxiliar o engenheiro de software na especificação e instanciação do Ambiente de Desenvolvimento de Software (ADS) mais adequado ao desenvolvimento de um produto específico;

b) auxiliar o engenheiro de software a implementar as ferramentas necessárias ao ADS definido em a);

c) desenvolver o produto (software) usando a estação através do ADS especificado em a) e produzido em b);

d) executar o software, dado que, eventualmente, o software-produto poderá ser executado na própria estação para ele configurada (isso é sempre verdade, pelo menos, na fase de testes).

I.3 - OBJETIVO.

O objetivo deste trabalho é desenvolver um estudo sobre representação do conhecimento e concluir apresentando um método que permita a representação das características e recursos de métodos de desenvolvimento de software.

O trabalho pretende contribuir para o desenvolvimento do Projeto TABA, no que se refere à sua base de conhecimentos.

A figura (I.1) mostra um diagrama de fluxo de dados da Estação TABA, baseado em (ROCHA-90).

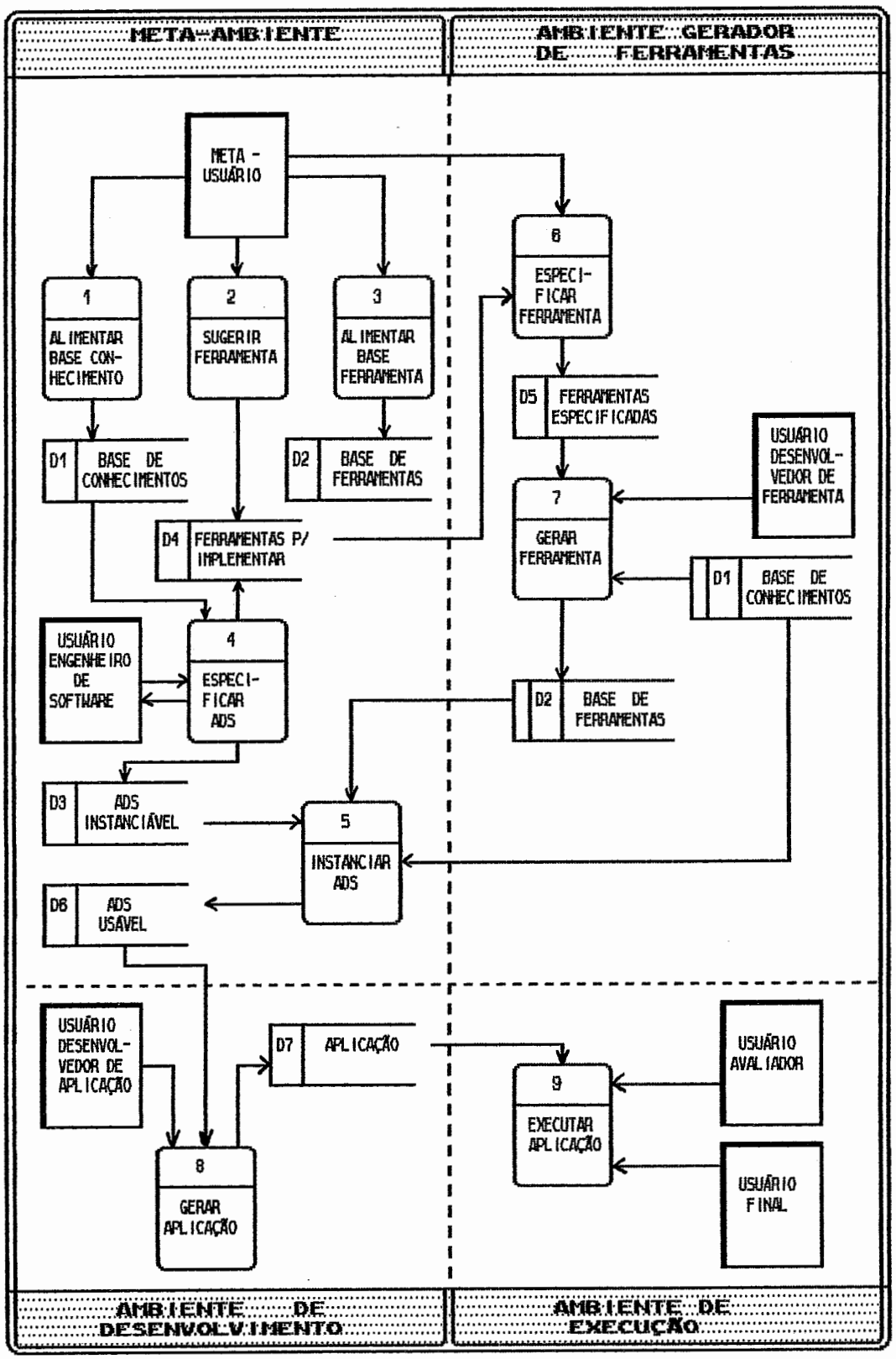


Figura 1.1: DFD da Estação TABA (ROCHA-90)

O meta-ambiente tem duas funções principais (ROCHA-89):

- especificar o ADS mais adequado para o desenvolvimento de uma determinada aplicação, e,
- instanciar o ADS especificado, isto é, torná-lo operacional.

A especificação pode ser realizada diretamente pelo especificador de ambientes como também através de um sistema que auxilia na tomada de decisões sobre os componentes desejáveis para um ADS, tendo em conta as características de um produto a ser desenvolvido.

O objetivo do sistema que auxilia o especificador de ADS é fornecer um acesso conveniente aos dados e conhecimentos, e ajudar os engenheiros de software na definição de ambientes apropriados a aplicações específicas.

A escolha de um ADS é um processo heurístico e não algorítmico, ou seja, os aspectos envolvidos na escolha não são definidos explícita e claramente para todos os casos, necessitando do conhecimento de especialistas. Estes especialistas devem ter conhecimento da área de aplicação e experiência no uso de métodos e ferramentas, o que muitas vezes não pode ser encontrado em uma única pessoa. Estas questões nos levaram a definir o especificador de ADS da Estação TABA como um sistema especialista de apoio à decisão.

A partir da especificação do ADS recomendado pelo sistema ou descrito pelo engenheiro de software, o instanciador gera o ADS.

O instanciador tem duas funções básicas:

- selecionar, entre os componentes disponíveis na Estação, os que foram indicados pelo ADS especificado, e,
- gerar um assistente especialista de auxílio à utilização do ADS instanciado.

O desenvolvimento do meta-ambiente da Estação TABA, implica na resolução de algumas questões que, no momento, são objeto de pesquisa da equipe envolvida no projeto. Neste sentido, além do sub-projeto que está sendo tratado nesta tese, estão em andamento os seguintes sub-projetos:

- Atributos de Qualidade para Ambientes de Desenvolvimento de Software;
- Taxonomia de Métodos e Ferramentas;
- Taxonomia de Domínios de Aplicação;
- Taxonomia de Modelos de Ciclo de Vida;
- Aquisição do Conhecimento em Engenharia de Software;
- Construção de um assistente especialista para ADS;

O método de representação proposto neste trabalho poderá ser utilizado pelo especificador de ambientes, quando houver necessidade de apresentar ao usuário um determinado método ou ferramenta. Poderá, ainda,

ser utilizado pela função do meta-ambiente que instancia o ADS especificado.

I.4 - CONTEÚDO.

Este trabalho está organizado em seis capítulos e um apêndice.

O **capítulo II** apresenta os objetivos da ES e da IA, com a finalidade de destacar os pontos principais destas duas áreas no processo de desenvolvimento de software.

É apresentada, também, as características básicas entre os programas de inteligência artificial e os programas convencionais e descrito um ciclo de vida que contempla as atividades essenciais dessas áreas.

O **capítulo III** estuda os principais métodos de representação do conhecimento descritos na literatura atual.

São, também, apresentadas definições, objetivos e características da representação do conhecimento, bem como, exemplos de sua utilização.

No **capítulo IV** apresenta-se, como resultado da pesquisa realizada no capítulo III, uma estrutura de fatores que descreve as funções consideradas importantes para análise das características de sistemas de representação do conhecimento.

O capítulo V descreve um sistema de representação do conhecimento para métodos de desenvolvimento de software. São analisadas as necessidades consideradas básicas para a concepção do sistema como, ainda, os resultados desejados em função das necessidades.

O capítulo VI apresenta as considerações finais do trabalho, ressaltando as idéias mais fortes deixadas ao longo do seu desenvolvimento, como ainda, sugestões para a elaboração de trabalhos futuros.

O apêndice A consta de uma descrição suscinta do método Análise Estruturada, onde nos restringimos à abordagem (GANE-83), como ainda, da representação do conhecimento deste método através do sistema de representação descrito no capítulo V.

Capítulo II

A INTELIGÊNCIA ARTIFICIAL
EM
AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE

II.1 - OBJETIVOS DA ENGENHARIA DE SOFTWARE.

A Engenharia de Software (ES) é uma disciplina, da tecnologia de computação, relacionada ao desenvolvimento e manutenção de produtos de software em tempos específicos e dentro das estimativas de custo consideradas (FAIRLEY-85).

Os principais objetivos da ES são o aumento da qualidade dos produtos de software, o aumento da produtividade e satisfação no trabalho das pessoas envolvidas.

A ES usa os fundamentos da ciência da computação, gerência, economia, habilidade de comunicação e modelos de solução de problemas da engenharia comum. Os fundamentos de gerência fornecem as bases para a gerência dos projetos. Os fundamentos de economia fornecem as bases para a estimativa e controle de custos. As habilidades de comunicação são essenciais para a comunicação entre usuários, gerentes, analistas de software, analistas de hardware e demais técnicos.

Devido, ainda, à ES estar envolvida com o desenvolvimento e manutenção de produtos tecnológicos, as técnicas de resolução de problemas comuns a todas as engenharias são usadas fornecendo bases para o planejamento, gerência, análise sistemática, desenvolvimento metódico, fabricação cuidadosa, validação externa e atividade de manutenção. São usadas notações, ferramentas e técnicas apropriadas a cada uma dessas áreas específicas.

De modo a alcançar os objetivos propostos pela ES, satisfazendo os atributos de qualidade já mencionados, torna-se necessário o uso de métodos para desenvolvimento e implantação de sistemas (FAIRLEY-85) (ARTHUR-85) (STAA-83) (TURNER-84) (DEMARCO-78).

Para disciplinar o desenvolvimento de software deve-se estipular uma sequência de etapas a serem vencidas. Esta sequência inicia-se ao conceber-se o produto e termina ao descontinuar-se o uso do mesmo, e em cada uma destas etapas, utilizam-se os resultados da etapa anterior, detalhando e/ou formalizando mais estes resultados. Esta sequência de etapas é conhecida como **CICLO DE VIDA** (FAIRLEY-85) (KELLER-87) (YOURDON-82) (PRESSMAN-82).

O desenvolvimento de produtos de software geralmente é realizado por refinamentos sucessivos, partindo-se de uma definição geral do sistema e, a partir daí, acrescentando-se detalhes à medida que se progride no desenvolvimento (WETHERBE-84) (GANE-83).

Os princípios básicos da **ENGENHARIA DE SOFTWARE**, definidos por (BOEHM-76) são:

1. Gerenciar usando um plano baseado no ciclo de vida do sistema;
2. Realizar validação continua;
3. Manter controle disciplinado do projeto;
4. Usar práticas modernas de programação;
5. Manter um registro claro dos resultados obtidos;
6. Utilizar pouco pessoal e de alta qualidade;
7. Manter equipe encarregada de melhorar o processo.

Os recursos disponíveis para auxílio ao desenvolvimento, através das sequências de tarefas pré-definidas por um método, são muitos.

Para que um método de desenvolvimento possa ter seu uso adequado às necessidades de seus usuários, é necessário algo mais que auxilie na sua utilização, tornando automáticas tarefas de geração de relatórios, documentos, verificação de cronogramas, acompanhamento e controle do sistema em elaboração. Além disto, utilizar ferramentas integradas para que o desenvolvimento do software seja realizado com maior rapidez e segurança, resultando em maior produtividade e qualidade no produto gerado.

Assim, surgiram os denominados Ambientes de

Desenvolvimento de Software (ADS), com objetivos tais que, em sua essência, buscam colocar à disposição do profissional, as ferramentas necessárias à elaboração de sua tarefa.

No projeto TABA, um ADS será gerado através do Meta-Ambiente, utilizando-se o especificador de ambientes para apoiar a escolha do melhor ADS em função das características da aplicação e do contexto onde o software será desenvolvido (ROCHA-89).

No contexto do Projeto TABA define-se um ADS como sendo composto de (ROCHA-87):

- um ciclo de vida, que define as fases do processo de desenvolvimento e as atividades a serem realizadas em cada fase;
- métodos, usados para organizar o pensamento e o trabalho do usuário, ao longo do processo de desenvolvimento;
- instrumentos, que tornam possível a utilização dos métodos, e;
- ferramentas, que automatizam os instrumentos.

O uso de um ambiente apresenta grandes benefícios em termos de:

- fazer cumprir os procedimentos de um método adotado para o desenvolvimento;

. ajudar no estabelecimento de requisitos para o software;

. melhorar a qualidade do software desenvolvido;

. simplificar o processo de manutenção;

. aumentar a produtividade no processo de desenvolvimento;

. ajudar no processo de controle e verificação;

. facilitar a comunicação entre os integrantes de uma equipe de desenvolvimento;

. facilitar as atividades de gerenciamento.

Sendo assim, um ADS produz sérias mudanças no ambiente e na forma de trabalho, o que afeta as equipes de desenvolvimento em dois sentidos (MASURKAR-82) (LAUBER-82) (HAUSEN-84):

1 - As equipes devem adaptar-se à nova forma de trabalho, e,

2 - As equipes devem trabalhar utilizando o ADS.

Um ADS deve ajudar a resolver o problema prático de elaboração de software, e portanto, deve abarcar todas as etapas do desenvolvimento, desde a sua definição até as etapas de manutenção, cobrindo as tarefas de controle (verificação e validação) e gerenciamento de projetos (planejamento, controle e execução). Portanto, o ambiente trás implícita a adoção de um modelo de ciclo de vida e a

incorporação de recursos que permitam levá-lo adiante, através da especificação e uso de ferramentas adequadas.

Um ADS deve ter as seguintes características (HAUSEN-84):

1. Suportar o processo completo de desenvolvimento, incluindo aspectos gerenciais e técnicos;

2. Incorporar ferramentas de software para suportar o processo de desenvolvimento adequado ao método;

3. Incorporar uma base de informações contendo as ocorrências do sistema em desenvolvimento;

4. Incorporar uma base de conhecimentos contendo a estrutura do método, a estrutura para o planejamento do projeto e as restrições de integridade a serem impostas;

5. Incorporar ferramentas para interface dos usuários com o ADS;

6. Incorporar um sistema gerenciador que permita tanto a manipulação das informações necessárias à tomada de decisões, como acesso aos recursos contidos no ADS.

Em função das características acima descritas para um ADS, destacamos os seguintes domínios envolvidos: o **método de desenvolvimento de software**, as **ferramentas de software** e as **características da aplicação**.

O **método** deverá apoiar atividades de

gerenciamento (planejamento, controle e documentação), quanto atividades de desenvolvimento propriamente ditas (especificação lógica, projeto de arquitetura, especificação de programas, etc), assim como atividades para teste e manutenção.

Com relação às **ferramentas de software**, estas deverão ser, portanto, notacionais e cognitivas, sendo as primeiras as linguagens utilizadas pelo usuário para a especificação de suas tarefas, e as últimas, os recursos oferecidos com o objetivo de aumentar a capacidade dos usuários, assim como auxiliar na utilização do método.

Com relação à **aplicação**, sua essência deverá ser conhecida, através de recursos que permitam identificar suas funções, objetivos e características específicas.

II.2 - OBJETIVOS DA INTELIGÊNCIA ARTIFICIAL

Não é de forma natural que a palavra inteligência aparece como destaque em diversos trabalhos, sendo de interesse de profissionais, técnicos e estudiosos nos dias de hoje. Anos atrás, cientistas da computação como VON NEWMANN, TURING, NEWEL, SIMON, McCARTHY e MINSKY reconheceram que o processo artificial poderia ser utilizado para que as máquinas se assemelhassem aos aspectos de pensamento do ser humano. A Inteligência Artificial (IA) permite o estudo de processos simbólicos e a tentativa de conhecer o que chamamos de "inteligência".

Quando tratamos de IA, estamos lidando com uma ciência que considera essencial a forma de pensamento do ser humano. Não querendo dizer com isso, que se saiba como exatamente a mente humana funciona. Porém, o que se sabe atualmente sobre a inteligência humana, embora ainda muito pouco, é fundamental. Este pouco conhecimento já nos permite fazer certas suposições sobre como pensamos (QUILLIAN-75) (MINSKY-75) (SHANCK-75) e aplicar essas suposições ao projeto de programas de IA, buscar novas soluções à problemas existentes, ou pelo menos, compreender melhor o problema (PARTRIDGE-86) (FRENZEL-87) (ROLSTON-88), dentre outros.

O objetivo de um programa de IA é fundamental, pois é a partir dele que buscamos uma solução (LEVINE-88). Ou seja, todo o pensamento nos ajuda a conseguir alguma coisa. Por exemplo: quando o despertador toca pela manhã, um processo de pensamento deve ser empregado para guiar nossa mão até ele e desligá-lo. Não é uma reação automática. Buscou-se uma resposta específica para a solução de um determinado problema.

Os resultados finais para os quais todos os nossos processos de pensamento estão dirigidos são chamados **objetivos**. Quando se projeta um sistema de IA o objetivo do sistema deve sempre ser mantido em mente. Lembre-se, diz (LEVINE-88): **"não fazemos as coisas porque pensamos; pensamos porque existem coisas que temos de fazer"**.

A lista seguinte, obtida de (RICH-88), contém um resumo de algumas das áreas que são tratadas pela IA:

. Jogos

. Prova de teoremas

. Resolução de problemas gerais

. Percepção: visão e fala

. Compreensão de linguagem natural

. Resolução de problemas especializados;
  - matemática simbólica
  - diagnose médica
  - análise química
  - projeto de engenharia

Assim, a "resolução de problemas gerais" pode ser explorada na prática, considerando alguma área específica de atuação, para centralizar os esforços do uso desses conceitos dentro dos problemas e objetivos desejados.

É assim que, após alguns anos, pesquisadores em IA conseguiram obter programas que jogam xadrês, provam teoremas matemáticos e projetam circuitos altamente complexos. Hoje, seus descendentes são os chamados **Sistemas Especialistas**, que a cada dia encontram novas aplicações nas mais diversas áreas (KELLER-87) (LEVINE-88) (HU-87) (BUCHANAN-84) (LUCAS-84) (ROLSTON-88) (GENARO-86) (ALTY-86).

Um sistema especialista lida com uma pequena área técnica que pode ser convertida da inteligência humana para a artificial. Por exemplo: sistema especialista em vendas,

em avaliação do aprendizado e planejamento financeiro (LEVINE-88). Em (ALTY-86) podem ser vistos outros tipos de aplicação.

Como o alcance dos objetivos é a meta de qualquer sistema de IA, a primeira etapa no planejamento de tal sistema é definir um conjunto de objetivos. É preciso saber que tipo de problema se quer solucionar e ser capaz de descrevê-lo em termos concretos antes de se começar a criar um programa para resolvê-lo.

(LEVINE-88) apresenta os componentes de um sistema de IA baseado em regras, conforme a figura (II.1).

Para auxiliar nas soluções dos problemas de IA, técnicas vêm sendo desenvolvidas com o objetivo de melhorar a qualidade e permitir que o desenvolvimento seja mais rápido e eficaz. Essas técnicas incluem estudos sobre:

. Representação do conhecimento;

. Algoritmos de pesquisas;

. Mecanismos de inferência;

. Aquisição do conhecimento;

. Manipulação do conhecimento;

. Base de conhecimentos.

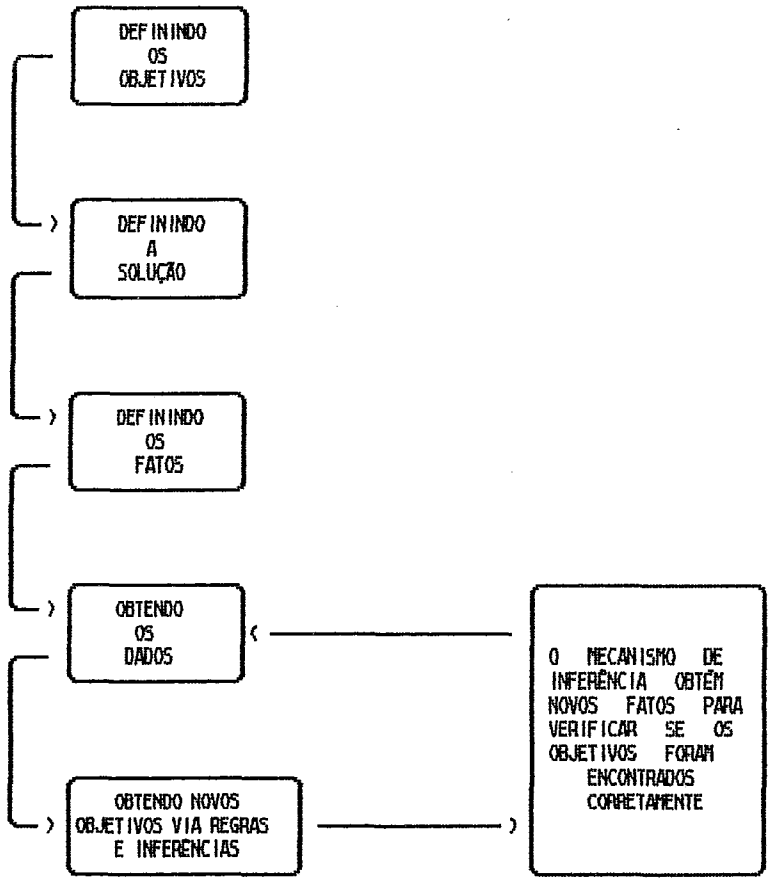


Figura II.1: Componentes de um sistema de IA baseado em regras (LEVINE-88).

A **representação do conhecimento** permite a visualização e relação dos conhecimentos do domínio tratado, como ainda, estruturar esse conhecimento para entrada e saída na base de conhecimentos.

O **algoritmo de pesquisa** utiliza técnicas apropriadas para as resoluções das inferências necessárias. Por exemplo: referência cruzada entre premissas e conclusões, marcando as premissas que aparecem nas conclusões e, durante a execução, marcando as regras utilizadas.

O **mecanismo de inferência** realiza os acessos de entrada e saída na base de conhecimentos, procurando resolver as inferências de forma eficiente.

A **aquisição do conhecimento** é o recurso que permite extrair o conhecimento de um domínio particular a partir de alguma fonte, geralmente de um ser humano especialista, e construir esse conhecimento numa forma computadorizada.

A **manipulação do conhecimento** é o recurso que permite o acesso à base de conhecimentos, para a consulta e atualização da informação obtida, de forma fácil, rápida e completa.

A **base de conhecimentos** é uma estrutura de dados utilizada para armazenar as informações características do domínio, permitindo o acesso de entrada e saída de forma fácil e eficiente.

Com relação à uma definição sobre IA, ainda não existe um consenso entre os cientistas desta área. MINSKY, por exemplo, diz não se aventurar e considera prematuro definir o que vem a ser IA, sua definição se resume que "IA é a habilidade de resolver problemas". Não obstante à preocupação desse conceituado cientista, conhecido como o "pai da IA", abaixo apresentam-se algumas definições sobre IA:

a) "É a transferência das características da inteligência humana para as máquinas." (LEVINE-88)

b) "É o estudo de como fazer os computadores realizarem tarefas em que, no momento, as pessoas são melhores." (RICH-88)

c) "A área da ciência da computação interessada no uso de computadores em tarefas que, normalmente, exigem conhecimento, percepção, raciocínio, aprendizado, entendimento e habilidades cognitivas similares." -N.J. NILSSON- fonte: (LUCENA-87)

Embora as três definições acima apresentem, em sua essência, um mesmo objetivo para a IA, é importante notar o nível de detalhes conceituais que cada uma delas traz.

Em a), observa-se uma definição bastante geral de IA, sem se preocupar com as dificuldades e, até impossibilidade, de se conseguir essa "tal" transferência para as máquinas, das caracaterísticas do ser humano.

De b), nota-se uma semelhança com a definição de a). No entanto, é posto como característica da IA, ser um estudo. Porém, é sabiamente colocado que, essas características são melhores realizadas pelo homem.

A definição c), no entanto, parece mais completa por externar claramente as necessidades que temos, ainda, de estudar e pesquisar sub-áreas derivadas da IA, que são necessariamente fundamentais para se, então, conseguir o que as definições apresentadas indicam a transferência para as máquinas das características do ser humano.

Com o advento da inteligência artificial, o conhecimento é visto agora como a ponta de lança da competição. O emprego de processos baseados em computadores e de conhecimento intensivo, acresce o nível de inteligência do usuário. Há semelhanças mas também diferenças entre a inteligência natural e a artificial. Isso será demonstrado na figura (II.2) e divide-se em três grupos. Em dois fatores a inteligência artificial e a natural são quase iguais; em seis outros a inteligência artificial domina. Em termos de criatividade e sendo parte de nossa cultura atual, a inteligência natural é superior.

Há outras questões na escala. É mais fácil interagir com uma máquina inteligente (ou pessoa) do que com uma não-inteligente. Tal interação deve seguir um método, e esses métodos e suas regras estão se tornando as palavras-chaves de nossa civilização.

Há também forças negativas trabalhando contra o uso da inteligência artificial no comércio e na indústria. Por exemplo:

1) Não se acredita que as máquinas possam ter inteligência;

2) A nossa cultura dominante não se adaptou ainda plenamente ao computador, muito menos a uma máquina inteligente.

3) Os especialistas em computadores ainda estão muito orientados para o "processamento de dados".

4) As linguagens disponíveis (Lisp, Prolog, Emycin) não são necessariamente transportáveis entre as máquinas. Isso cria uma separação danosa entre o desenvolvimento e o uso final.

INTELIGÊNCIA ARTIFICIAL	INTELIGÊNCIA NATURAL
Igual	
.Cresce com a experiência	.Cresce com a experiência
.Torna-se obsoleta se não for utilizada	.Torna-se obsoleta se não for utilizada
Positivo	**Negativo**
.Sobrevive ao ciclo de vida do especialista	.Desaparece após a morte do especialista
.É consistente	.É irregular
.Acumula-se permanentemente	.É efemera, oscilante
.É bem documentada	.É mal documentada
.É facilmente duplicada	.É difícil transmitir
.Seu custo é razoável	.Custa caro
Negativo	**Positivo**
.Atualmente não é criativa	.É criativa
.É ainda alheia a cultura atual	.É parte da cultura atual

Figura II.2: Semelhanças e diferenças entre a inteligência artificial e a natural (CHORAFAS-88).

À medida que o tempo passa, as experiências se acumulam e máquinas mais poderosas se tornam disponíveis; podemos esperar ver os sistemas de IA com milhares de regras. Também melhorará grandemente as abordagens explicativas que podem dizer por que o sistema fez o que fez, aconselhou determinado curso de ação e enfatizou questões de importância.

II.3 - CARACTERÍSTICAS ENTRE PROGRAMAS CONVENCIONAIS E DE INTELIGÊNCIA ARTIFICIAL.

As técnicas para criação de programas de IA variam de acordo com a associação do programa com o computador digital.

Os programas de computador convencionais são baseados em um algoritmo, claramente definido, com procedimentos passo-a-passo buscando a solução do problema. São utilizados dados tais como números, letras ou palavras.

No caso da IA, os programas não são baseados num processo algoritmico, eles utilizam representação e manipulação de símbolos, que podem ser uma letra, palavra ou número que são utilizados para representar objetos, processos e suas relações. Os objetos podem ser pessoas, coisas, idéias, conceitos, eventos ou declaração de fatos. Pelo uso de símbolos, é possível a criação de uma base de conhecimentos contendo fatos, conceitos e as relações entre eles.

Virtualmente todos os computadores digitais são de operações algoritmicas, baseados no conceito de Von Newmann: são executadas sequencialmentes as instruções armazenadas em memória. A questão é, como implementar o processamento simbólico numa máquina algoritmica ? Um programa é escrito de tal forma a permitir a representação e manipulação dos símbolos.

A figura (II.3) apresenta uma representação de como as operações dos programas convencionais e os de IA

são implementadas num computador digital.

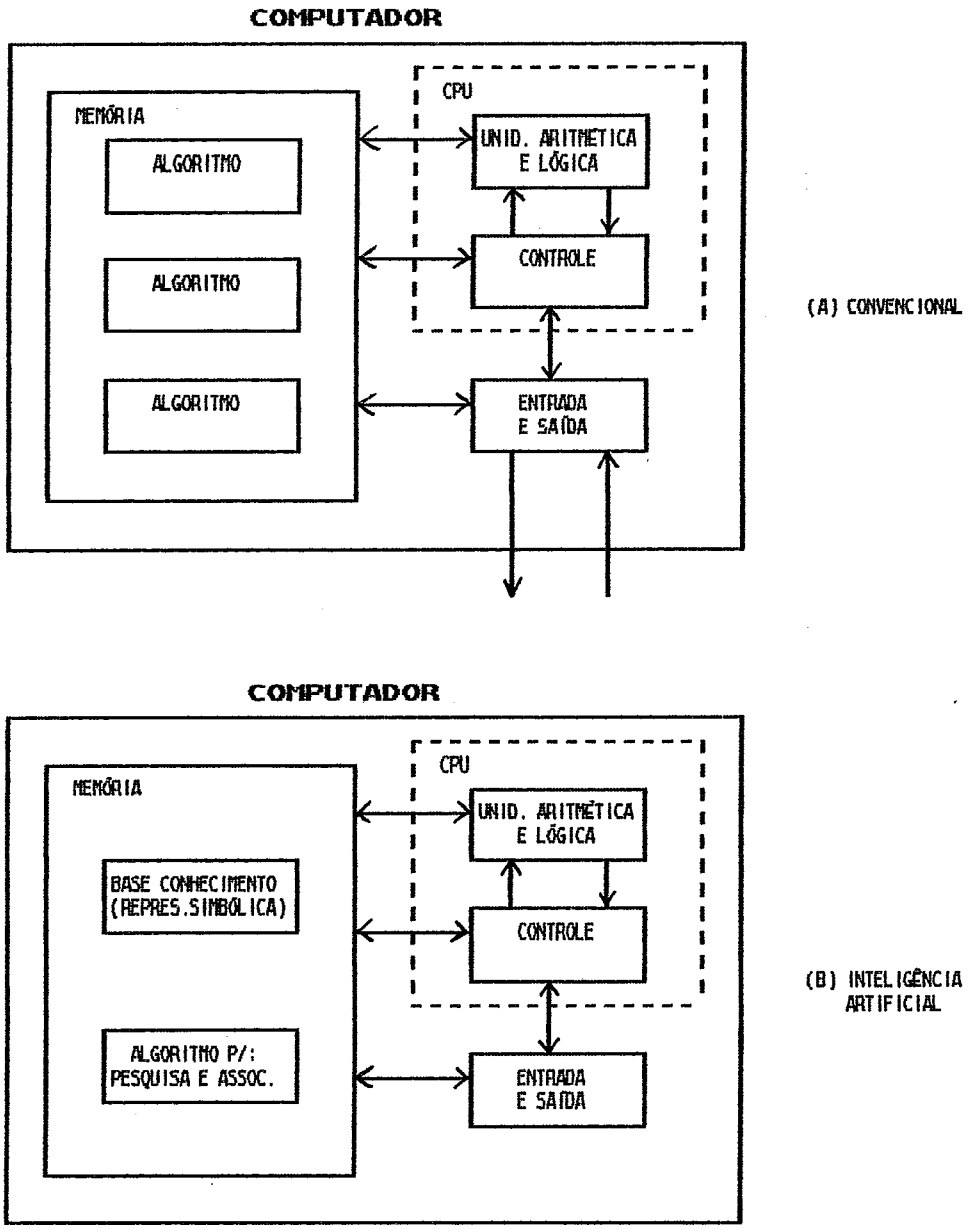


Figura II.3: Como os computadores digitais trabalham com métodos convencionais (a) e IA (b) (FRENZEL-87).

Pesquisadores como (LUCENA-87) e (LEVINE-88) citam a vantagem de programas desenvolvidos utilizando

técnicas de IA, serem fáceis de manutenção e rapidamente desenvolvidos. No entanto, sem considerações sobre essas características, o importante é perceber a transição evolutiva em que nos encontramos, onde a IA invade nossos dias com conceitos e técnicas, a ES busca se aprimorar e acompanhar as necessidades existentes buscando melhor forma de atendê-las e de percepções futuras da ciência da computação, que poderão ou não mudar até o que hoje estamos tratando como as mais modernas formas de encarar o desenvolvimento de sistemas.

(Lucena-87) apresenta, conforme figura (II.4), uma comparação dos programas de IA com os programas convencionais, como visto a seguir:

PROGRAMAÇÃO EM IA	PROGRAMAÇÃO CONVENCIONAL
(a) .busca heurística (passos de solução implicitos) (b) .estrutura de controle usualmente separada do domínio do conhecimento (c) .usualmente fácil de modificar, atualizar e ampliar (d) .algumas respostas incorretas são toleráveis (e) .respostas satisfatórias são, em geral, aceitas	(a) .algorítmica (passos de solução explicitos) (b) .informação e controle integrados (c) .modificação difícil (d) .respostas corretas são requeridas (e) .busca-se, em geral, a melhor resposta possível

Figura II.4: Comparação esquemática dos programas de IA com os convencionais (LUCENA-87).

Com a finalidade de ilustrar como as atividades de desenvolvimento de programas de IA podem ser integradas com as de programas convencionais, destacamos o ciclo de vida descrito em (KELLER-87) apresentado na figura (II.5). Este ciclo de vida representa uma estrutura de atividades voltadas a projetos que incluem algum processamento baseado no conhecimento.

Um ciclo de vida além da orientação ao desenvolvimento auxilia ao desenvolvedor raciocinar e modelar sua idéia para a solução do problema e, também, permitir que o usuário-final possa participar na fase de concepção e elaboração do projeto, entendendo e discutindo sobre as idéias do desenvolvedor e as suas próprias.

Atualmente revela-se de muita importância a existência de uma base de dados descrevendo e definindo os dados da Empresa, estabelecendo quais informações estão disponíveis e os requisitos para uso e criação. O projeto de um software tem a tendência de se desenvolver baseado numa base de dados, resultando, com isso, compatibilidade com os objetivos e necessidades da Empresa, além de, organização e segurança para o desenvolvimento do projeto.

(KELLER-87) acredita que a meta deveria ser um diagrama o qual suportaria as questões dos processamentos tradicional e o baseado em conhecimento, sem detrimento dos requisitos de ambos, através da integração das técnicas, conceitos e ferramentas.

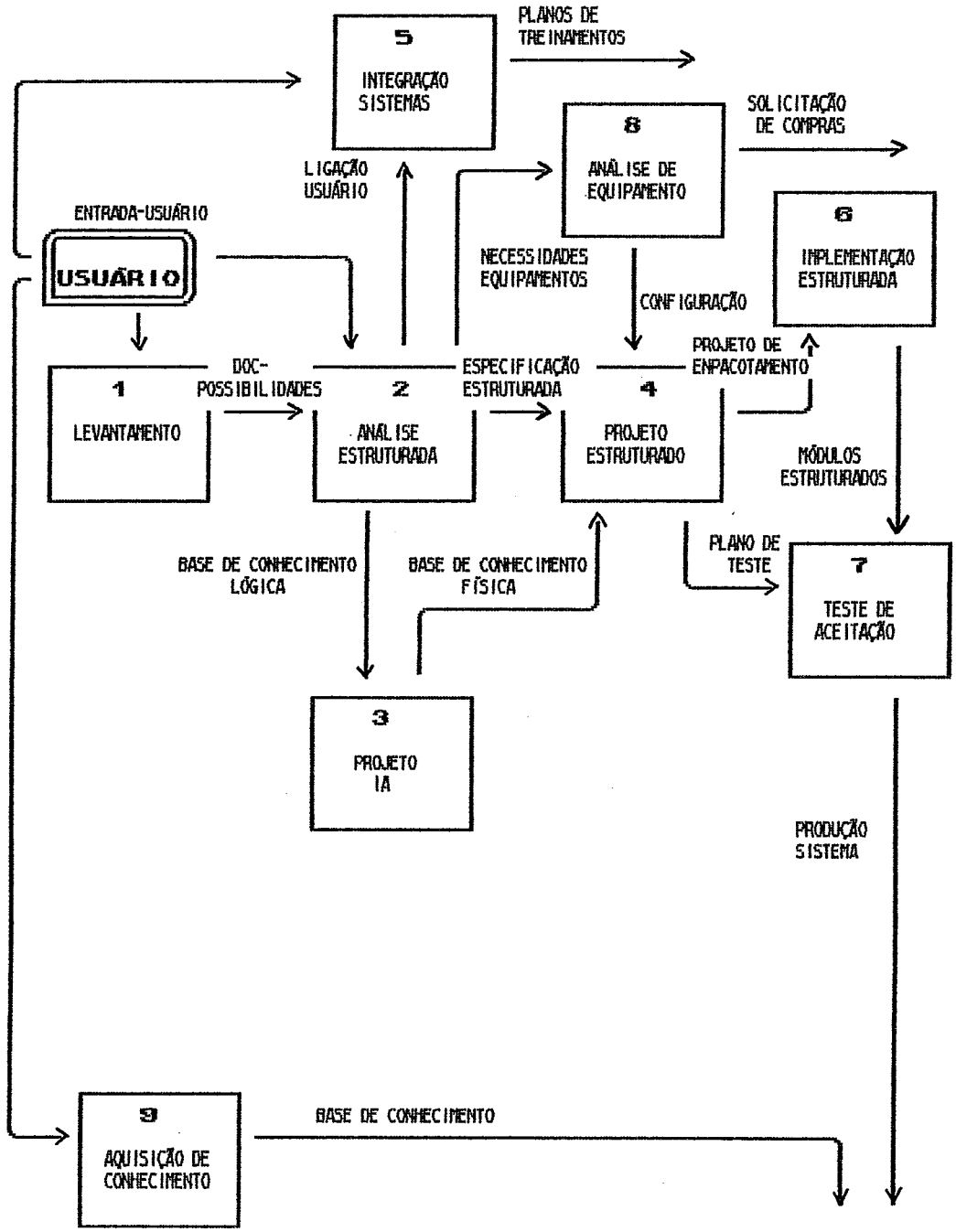


Figura II.5: Ciclo de vida para IA (KELLER-87)

Em linhas gerais, o projeto de um sistema de software inicia com um levantamento das características básicas do projeto, cuja primeira tarefa é decidir sobre a viabilidade de seu desenvolvimento. Quando o projeto é aprovado, ele é seguido para a atividade de Análise Estruturada, na qual se inicia a especificação das necessidades do usuário em termos de funções a serem executadas e dos relacionamentos dos dados entre elas. A Especificação Estruturada que resulta da análise é usada na fase de Projeto para especificar como as necessidades do usuário serão implementadas, e a fase de Implementação é dedicada para a construção do produto como especificado no Projeto de Empacotamento.

Durante a análise estruturada, as informações lógicas necessárias para o projeto, ou sejam, conhecimento e dados, são identificados sem interesse particular de como serão implementadas fisicamentes. A descrição da Base de Conhecimentos (KB) é utilizada no Projeto Físico da KB para especificar os detalhes da implementação da base de conhecimentos.

A colocação da atividade 9, Aquisição do Conhecimento, é uma parte especial desde que durante o levantamento e análise estruturada sejam colecionadas informações de um conhecimento especialista. Após a análise, as atividades de projeto são voltadas a focalizar o software como uma coleção de conhecimento, e o processo de aquisição de conhecimento continua de forma mais independente.

O sistema de Integração e a atividade de Análise de Equipamentos são importantes no ciclo de vida embora não representem uma parte direta da técnica de especificação do ciclo.

O que se destaca com o ciclo de vida apresentado pela figura (II.5), não é propriamente a questão do ciclo ideal para o desenvolvimento com técnicas de IA, é de como as atividades convencionais de desenvolvimento de programa se comportam quando consideradas, também, as características da IA. Isto, para nos levar a obtenção de resultados, ou procedimentos, como descritos na figura (II.4).

No entanto, devemos decidir se queremos ou não o desenvolvimento de um projeto de software que contemple essas atividades específicas da IA, tornando-se importante considerá-las nas fases de levantamento, análise dos requisitos do usuário e no desenvolvimento do projeto.

As diferenças entre os sistemas de software convencionais e os de IA são frequentemente uma questão de quantidade mais do que qualidade, apesar de existirem diferenças qualitativas. A maior diferença está na base de conhecimentos a qual contém não somente dados, mas também relações entre eles, numa máquina de inferência na qual se utilizam técnicas não encontradas nos sistemas de software tradicionais e nas pessoas capacitadas para o trabalho: o engenheiro de conhecimento e um especialista no domínio tratado.

Capítulo III

CARACTERÍSTICAS E MÉTODOS PARA A REPRESENTAÇÃO DO CONHECIMENTO.

III.1 - Definição e objetivos.

Todos os elementos nos quais consiste o processo humano de tomada de decisão -objetivos, fatos, regras, mecanismos de inferência e poda- devem ser reunidos em um programa de computador para que ele possa ser realmente qualificado como um programa que possui inteligência artificial. É preciso saber que tipo de problema se quer solucionar e ser capaz de descrevê-lo em termos reais antes da busca por uma solução (LEVINE-88).

Para a construção de um edifício, inicialmente, é elaborado um projeto contendo os objetivos e características que se deseja para tal projeto. Essa atividade é uma das primeiras utilizadas na área da construção civil, onde o desenho do edifício, plantas, relatórios e etc, permitem conhecer os objetivos e características do projeto.

Na ES, por exemplo, o método Análise Estruturada (GANE-83), utiliza o recurso do DFD (Diagrama de Fluxo de Dados) e Dicionário de Dados para apresentar ao elaborador de um sistema e seu usuário, uma representação do sistema a ser desenvolvido.

Da mesma forma, na área de IA o desenvolvimento de sistemas necessita de algo que permita ao seu elaborador expor um problema e a solução que considera adequada.

Por estarmos tratando e manipulando conhecimento sobre algo, o recurso utilizado é denominado como técnica de **REPRESENTAÇÃO DO CONHECIMENTO**.

A figura (III.1) apresenta os tipos de informações que podem residir numa base de conhecimentos e, por conseguinte, são elas passíveis de representação.

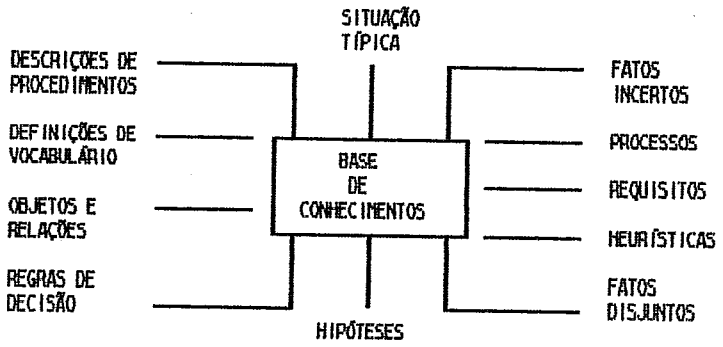


Figura III.1: Tipos de conhecimentos que podem ser representados (FIKES-85).

Representar o conhecimento significa organizar as informações requeridas numa forma tal que programas de IA possam ter rápido acesso a elas para tomar decisões, planejar, reconhecer objetos e situações, analisar cenas e outras funções cognitivas (LUCENA-87).

Existem diversas maneiras de se representar o conhecimento, cada uma apresentando uma característica particular e buscando resolver um determinado tipo de

problema. No entanto, alguns problemas se tornam mais difíceis de serem representados pelo fato da complexidade do problema afetar a estrutura de representação ou a técnica utilizada não ser compatível com o tipo de problema a resolver.

A representação do conhecimento para a IA acaba se tornando uma questão semelhante aos métodos para desenvolvimento de software na ES. Ou seja, não existe uma forma melhor, mais eficiente ou mais fácil. Existe sim, uma adequação melhor ao problema que se quer resolver.

O projeto de uma boa representação é a chave para transformar problemas difíceis em mais simples. Portanto, é razoável trabalhar muito para estabelecer que símbolos uma representação deve usar e como esses símbolos devem ser arrumados para realizar descrições significativas de coisas particulares. Entretanto, antes de descrever algumas possibilidades da representação, é importante nos familiarizar-mos com alguns termos (FRENZEL-87):

- Uma representação é um conjunto de convenções sintáticas e semânticas que torna possível descrever objetos;
- A sintaxe de uma representação especifica os símbolos que podem ser usados e as maneiras como esses símbolos podem ser arranjados;
- A semântica de uma representação especifica como o significado está incorporado nos símbolos e nas disposições de símbolos permitidos pela sintaxe.

As Linguagens de programação, por exemplo, são representações para os procedimentos. Como tais, elas têm sintaxe e semântica. A semântica é especificada por uma descrição de como construções sintáticas particulares se relacionam para produzir alguma coisa.

Em comparação com linguagens de programação, a distinção entre a sintaxe e semântica das linguagens naturais é obscura. A sintaxe de uma linguagem natural diz respeito a como os verbos e substantivos e outras partes do discurso podem se juntar para formar frases. A semântica de uma linguagem natural diz respeito à relação entre frases de uma lado e objetos, relações, ações e eventos do mundo, do outro.

III.2 - REPRESENTAÇÃO DO CONHECIMENTO: características.

Para se descrever uma aplicação de forma adequada e esboçar sua solução, é que os métodos de representação do conhecimento são úteis e fundamentais para o sucesso de um programa de software.

Segundo (RICH-88) a representação de conhecimento pode tornar-se difícil de se obter quando o problema tratado tem uma complexidade maior. Por exemplo, no caso de se representar um nó no espaço de busca de um robô. Isto faz com que seja importante dividir a questão da representação em três sub-questões:

1- Como os objetos e fatos individuais podem ser representados ?

2- Como as representações dos objetos individuais podem ser combinadas para formar a representação de um estado do problema completo ?

3- Como a sequência de estados de problema que surgem em um processo de busca podem ser representados eficientemente ?

As questões 1 e 2 são normalmente citadas como o problema de representação do conhecimento, embora não havendo uma solução completa.

A questão 3 é importante no contexto de um processo de busca. Suponha numa representação que a descrição de um objeto seja grande, fazendo com que uma quantidade grande desses objetos leve a um gasto elevado de memória. E ainda, o tempo necessário para tratar com cada um desses fatos, que pode se tornar desnecessário para aqueles fatos que não se modificam, ao contrário daqueles que ao longo do tempo sofrem modificações.

A mente humana possui uma quantidade grande de conhecimentos armazenados e relacionados a uma incontável lista de objetos e idéias. Nossa sobrevivência depende da habilidade de como aplicamos esses conhecimentos em qualquer situação que apareça, e aprender continuamente com as novas experiências, para que sejamos capazes de responder a situações similares no futuro. Aquilo que geralmente é considerado **inteligência** pode ser dividido em uma coleção de fatos e um meio de se utilizar esses fatos

para alcançar os objetivos. Isto é feito em parte pela formulação de conjuntos de regras relacionadas a todos os fatos armazenados no cérebro. A seguir, um exemplo do tipo de fatos e de regras relacionadas que usamos todos os dias.

Fato/regra conjunto 1:

Fato 1: O forno está aceso.

Regra 1: SE eu puser minha mão em um forno aceso, ENTÃO eu vou me queimar.

Fato/regra conjunto 2:

Fato 2: Durante a hora do "rush" as ruas ficam repletas de carros.

Regra 2: SE eu tentar atravessar uma avenida a pé durante a hora do "rush", ENTÃO eu posso ser atropelado por um carro.

A maioria dos sistemas de IA possuem duas partes: uma base de conhecimentos e um mecanismo de inferência. A base de conhecimentos contém fatos sobre objetos e as relações entre eles para um determinado domínio. A base de conhecimentos pode conter ainda, conceitos, teorias, procedimentos práticos e suas associações. A partir dessas informações, a base de conhecimentos deverá ser utilizada para que se possam obter conclusões através de um mecanismo de inferência.

O mecanismo de inferência é um conjunto de procedimentos que são utilizados para examinar a base de conhecimentos de forma ordenada para se obter respostas a perguntas, solução de problemas ou tomar decisão dentro

do domínio.

Para que um sistema de IA esteja bem construído em relação à sua base e seu mecanismo de inferência, é necessário que seja utilizado um método de representação de conhecimento adequado aos objetivos do sistema em construção. Para isso, vários métodos vêm sendo desenvolvidos através dos anos. Esses métodos de representação compartilham duas características comuns. Primeiro, eles podem ser programados, utilizando uma linguagem de computador existente, e carregados em memória. Segundo, eles são orientados para que os fatos e outros conhecimentos possam ser utilizados para raciocínio.

Portanto, estamos lidando com dois tipos diferentes de entidades que pertencem a todas as discussões de representação (RICH-88). São elas:

. Fatos, que são verdades relevantes em algum mundo. Essas são as coisas que queremos representar.

. Representação de fatos, com algum formalismo escolhido. Essas são as coisas que efetivamente seremos capazes de manipular.

Para que as representações sejam adequadas e de interesse em relação ao problema, deverá haver, também, funções que mapeiem desde os fatos até as representações e das representações de volta aos fatos.

Representar o conhecimento num programa de IA significa escolher um conjunto de convenções para descrever

objetos e relações. Escolhe-se uma estrutura conceitual para pensar sobre o problema e uma convenção numa determinada linguagem computacional para implementação e manipulação dos conceitos.

Tendo em vista que o objetivo é o uso da representação do conhecimento em computador, é necessário considerar as seguintes questões. A primeira está em encontrar ou definir tais convenções, de sorte a permitir o reconhecimento e identificação do domínio tratado. A segunda questão, a escolha ou definição de convenções computacionais, complementa a operacionalidade do método definido de representação, resultante da primeira etapa, permitindo sua implementação em computador e, por conseguinte, a manipulação do conhecimento alí tratado. Ambas as questões são tratadas neste trabalho no capítulo (V).

De certo modo encontrar uma representação para o conhecimento é como escolher as estruturas de dados de um programa convencional. As tabelas de dados, por exemplo, podem ser representadas por "arrays". Mas as estruturas definidas para a manipulação do conhecimento impõem requisitos adicionais. É preciso lembrar que alguns dos conhecimentos de especialistas são feitos através de inferências, necessitando, então, de convenções para que um programa possa utilizar adequadamente e interpretar essas estruturas. Outra questão, é que um especialista (ou engenheiro de conhecimento) precisa ser capaz de acessar e

manipular a estrutura de conhecimento de forma rápida e fácil, permitindo a redefinição da base de conhecimento do programa, de forma interativa.

A questão de como representar o conhecimento para uso inteligente por programas é uma das duas maiores questões que motivam a pesquisa em IA. A outra, é um tema pesquisado e debatido nos últimos 25 anos, que é de como usar o conhecimento para solução de problemas inteligentes.

De modo ideal, o próprio programa seria capaz de controlar a aquisição do conhecimento.

A base de conhecimento e o mecanismo de inferência, são partes básicas e fundamentais no projeto de sistemas de inteligência artificial.

É através do mecanismo de inferência, que se obtém conclusões e decisões cujas informações estão contidas na base de conhecimento. Esta base, por sua vez, suporta a realização das tarefas, contendo fatos e relações sobre um domínio escolhido. E, portanto, a representação de conhecimento precisa ser eficaz para que esse processo se dê de forma satisfatória, viabilizando a elaboração de tal projeto.

A figura (III.2) mostra uma visão funcional de um agente inteligente de propósito geral, usada por (NEWELL-82) e referenciada por (LUCENA-87), para motivar a relevância do problema da representação do conhecimento

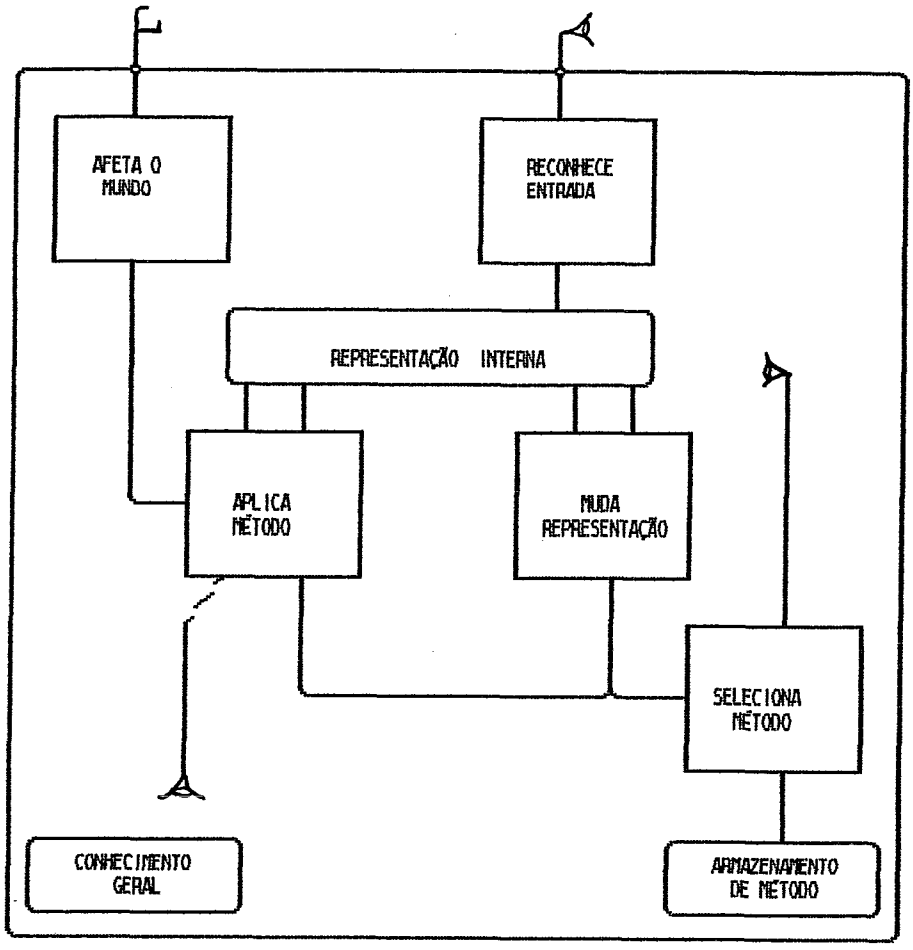


Figura III.2: Visão funcional de um agente inteligente (NEWELL-82) (LUCENA-87).

em IA. O agente inteligente está incorporado em um ambiente composto por tarefas; a "expressão de uma tarefa" entra através de um componente de "percepção" e é codificado em uma "representação inicial". A partir daí, começa um ciclo de atividades no qual ocorre um reconhecimento de um método usado para tentar a solução de um problema. O método recorre a uma memória do "conhecimento geral do mundo". No prosseguimento do ciclo, novos métodos e novas representações podem ocorrer, à medida que o agente procura resolver o problema.

Um método de representação do conhecimento é composto de lógica, semântica e descrições, que devem ser integrados para compor um sistema de IA, onde o acesso e manipulação das informações na base de conhecimento é voltado para as características da aplicação envolvida na tarefa.

Outras questões ainda relacionadas com a representação do conhecimento são fundamentalmente necessárias de serem consideradas, quer seja para escolha de um método quer seja para o projeto desse método. São elas:

- características e objetivos individuais do método, buscando uma melhor adequação do problema a ser tratado com o método a ser utilizado;
- observação das vantagens e desvantagens dos métodos buscando uma melhor apropriação dos recursos disponíveis: base de conhecimento, acesso à base e linguagem de manipulação;
- análise das características de métodos sintáticos e semânticos e, também, das características da aplicação, buscando com isso, a escolha mais adequada; e,
- consideração das características de métodos baseados em declarações e baseados em procedimentos, pois isto facilitará a escolha.

Os métodos de representação do conhecimento mais difundidos e que serão descritos, a seguir, com mais detalhes, são:

. Lógica de primeira ordem;

. Redes semânticas;

. Dependência conceitual;

. Quadros;

. Roteiros;

. Regras de produção; e,

. Redes neuronais.

III.3 - MÉTODOS TRADICIONAIS.

III.3.1 - LÓGICA DE PRIMEIRA ORDEM. (ISRAEL-83) (DAHL-83) (FRENZEL-87) (RICH-88) (LEVESQUE-88A)

Uma das formas mais antigas de representar o conhecimento é a LÓGICA. Considerada como sendo uma subdivisão da filosofia, o desenvolvimento e refinamento desse processo são geralmente creditados aos gregos.

A forma geral de qualquer processo lógico é ilustrado na figura (III.3).

Figura III.3: Usando a lógica para conclusões (FRENZEL-87).

Primeiramente, são fornecidas informações ao processo, declarações são elaboradas, ou observações são apontadas. Essas informações são as entradas para o processo lógico e chamadas de PREMISSAS. As premissas são usadas pelo processo lógico para a criação da saída a qual consiste de conclusões chamadas INFERÊNCIAS. Com este processo, fatos que são conhecidos como verdades podem ser usados para derivar novos fatos que também continuam sendo verdadeiros.

Neste processo há dois tipos básicos de raciocínio: **dedutível e indutível**.

Ambos os tipos são utilizados na lógica para fazer inferência a partir das premissas.

No raciocínio dedutível, em geral, as premissas são utilizadas para se obter uma conclusão específica. O processo é chamado de **raciocínio dedutível ou dedução**. O raciocínio caminha de um pricípio geral para conclusões específicas. Como ilustração, observa-se o exemplo a seguir:

O processo dedutível, geralmente, inicia com um silogismo, ou declaração de premissas e inferências, consistindo de três partes: uma premissa maior, uma premissa menor e uma conclusão.

. premissa maior: Eu não corro quando a temperatura excede 40 graus.

. premissa menor: Hoje a temperatura está 43 graus.

. conclusão : Logo, eu não correrei hoje.

O raciocínio indutível utiliza um número de fatos estabelecidos ou premissas para se chegar a alguma conclusão geral. Um exemplo a seguir ilustra este processo. Outra vez, um silogismo é usado para expressar o problema.

. Premissa: Diodos defeituosos causam falhas nos equipamentos eletrônicos.

. Premissa: Transistors defeituosos causam falhas nos equipamentos eletrônicos.

. Premissa: Cicuitos integrados defeituosos causam mal funcionamento em equipamentos eletrônicos.

. Conclusão: Portanto, dispositivo de semicondutor defeituoso é a causa de falhas em equipamentos eletrônicos.

Um detalhe interessante no raciocínio indutível é que a conclusão nunca é final ou absoluta. As conclusões podem se modificar se novos fatos forem descobertos.

Haverá sempre alguma incerteza na conclusão a não ser que todos os fatos possíveis sejam incluídos nas premissas, e isso é usualmente impossível. Como uma saída, o resultado de um processo de raciocínio indutível conterá sempre alguma avaliação de incerteza. Todavia, o grau de incerteza deverá ser reduzido com o uso de mais fatos ou premissas consideradas no processo de raciocínio. No exemplo apresentado, a incerteza está no fato de que no raciocínio indutível são necessários fatos/regras para que uma conclusão seja obtida. No entanto, não se pode ter uma certeza porque uma nova entrada de fatos/regras pode causar uma mudança na conclusão anterior, o que não acontece no raciocínio dedutível devido ao silogismo.

Para um computador executar um raciocínio usando a lógica, algum método deve ser utilizado para se converter o silogismo e o processo de raciocínio dedutível ou indutível em uma forma apropriada para a manipulação por um computador. O resultado é o conhecimento representado como um símbolo lógico ou matemático. Isto é, um sistema de regras e procedimentos que permite se delinear inferências de várias premissas usando uma variedade de técnicas lógicas. Esses métodos são geralmente conhecidos como LÓGICA COMPUTACIONAL.

Existem duas formas básicas de lógica computacional: a lógica proposicional e a lógica de predicados (também conhecidas como cálculo proposicional e cálculo de predicados).

Uma proposição nada mais é que uma declaração (ou sentença) que pode assumir somente dois valores: **verdadeiro ou falso**. Uma declaração é uma premissa que pode ser usada para se obter novas proposições ou inferências. Através do uso de técnicas se determina se a nova proposição é verdadeira (v) ou falsa (f).

Na lógica proposicional utilizam-se símbolos, tal como letras do alfabeto, para representar várias proposições, premissas ou conclusões. Por exemplo, considere as proposições usadas abaixo num processo simples de dedução:

A = O carteiro vem de segunda a sábado.

B = Hoje é domingo.

C = O carteiro não virá hoje.

Proposições simples como estas não são muito interessantes ou utilizadas. Problemas do mundo real envolvem relacionamento de diversas proposições. Para a formação de premissas complexas, duas ou mais proposições podem ser combinadas usando conectivos lógicos. Esses conectivos ou operadores são: **E , OU , NEGAÇÃO (NÃO), IMPLICAÇÃO E EQUIVALÊNCIA**. O significado de cada um desses conectivos e os símbolos usados para representá-los são dados a seguir na figura (III.4).

CONECTIVO	SÍMBOLO
E	$\wedge$, &, $\cap$
OU	v, u, +
NÃO	$\neg$, $\sim$
IMPLICA	$\supset$, $\rightarrow$
EQUIVALENTE	$\equiv$

Figura III.4: Conectivos ou operadores lógicos e seus símbolos (FRENZEL-87).

Esses símbolos são utilizados também na lógica digital. Eles são os mesmos que os utilizados na álgebra Booleana. Devido a lógica proposicional envolver somente a veracidade ou falsidade de proposições, a álgebra Booleana e todas as demais técnicas com essas características utilizam a lógica proposicional para análise, projeto e otimização de circuitos lógicos binários.

Os conectivos são usados para unir ou modificar proposições gerando-se novas proposições. A seguir serão mostrados alguns exemplos ilustrando esses conectivos.

1. NÃO

A = Está chovendo hoje.

não A = Não está chovendo hoje.

Uma tabela verdade pode ser utilizada para mostrar todas as combinações possíveis para este conectivo.

A	não A
T	F
F	T

Esta tabela verdade mostra que se A é verdade (v), então não A é falso (f). Se uma proposição é falsa (f), então não A torna essa proposição verdadeira (v).

2. E

Quando o conectivo E é usado para combinar duas proposições, a nova proposição resultante é verdade somente se ambas as proposições originais forem verdade.

D = O carro é preto.

E = O carro tem uma máquina de 6 cilindros.

F = O carro é preto e tem uma máquina de 6 cilindros.

F = D e E

Neste caso, F é verdade somente se D e E são verdadeiros.

A tabela verdade do conectivo E é mostrada abaixo, com todas as combinações possíveis para este conectivo.

D	E	F= D e E
F	F	F
F	T	F
T	F	F
T	T	T

3. OU

Quando o conectivo OU é usado para combinar proposições, a nova proposição gerada é verdade se uma ou outra ou ambas proposições originais forem verdadeiras.

P = A lua é um satélite

Q = A terra é um satélite

R = P ou Q, A lua ou a terra é um satélite

A tabela verdade para o conectivo OU é ilustrado à seguir:

P	Q	R=P ou Q
F	F	F
F	T	T
T	F	T
T	T	T

Observando a tabela verdade do OU nota-se que a nova proposição R é verdade se P é verdade, Q é verdade, ou ambas verdadeiras. Esta forma apresentada do conectivo OU é conhecida como OU INCLUSIVO. Uma outra forma do conectivo OU é conhecida como o OU EXCLUSIVO o qual significa que a proposição resultante só é verdade se uma ou outra proposição for verdade, para o caso de ambas serem verdade, a resultante é falsa.

A notação para o OU EXCLUSIVO pode ser, dependente do recurso de escrita, como (P e não Q) ou (não P e Q), isto porque (P e não Q) tem a mesma tabela verdade que (P ou exclusivo Q). Na lógica proposicional é frequentemente mais utilizada os conectivos E e NÃO

combinados para se obter a função resultante do OU EXCLUSIVO.

4. IMPLICA

O conectivo IMPLICA significa que se a proposição A é verdade, então a proposição B também é verdadeira: A ---> B. Exemplo:

A = A máquina do carro está com defeito.

B = Eu não posso dirigir hoje.

C = A ---> B

Uma outra maneira de explicar o conectivo IMPLICA é o uso da expressão "SE-ENTÃO". No exemplo acima, pode-se observar que "SE" a máquina do carro está com defeito, "ENTÃO" eu não poderei dirigir hoje.

A tabela verdade do conectivo IMPLICA é mostrada a seguir:

A	B	C=A--->B
F	F	T
F	T	T
T	F	F
T	T	T

O funcionamento do conectivo IMPLICA algumas vezes é difícil de se compreender. É interessante observar que a nova proposição C é verdade se A for falso (independente do valor lógico de B) ou se B for verdadeiro (independente do valor lógico de A). Isso escrito numa forma lógica tem-se " não A ou B ". Pode-se observar que B

é verdade na segunda e quarta entrada da tabela verdade. A é falso na primeira e segunda entradas. Como resultado em C, a primeira, segunda e quarta entradas são verdadeiras. Somente uma única vez a proposição resultante C é falsa e isto ocorre quando A é verdade e B é falso.

Utilizando o exemplo dado, temos que:

1a. linha t.v. - Se a máquina do carro não está com defeito, então eu posso dirigir hoje.
A ---> B é verdade

2a. linha t.v. - Se a máquina do carro não está com defeito, então eu não posso dirigir hoje.
A ---> B é verdade

3a. linha t.v. - Se a máquina do carro está com defeito, então eu posso dirigir hoje.

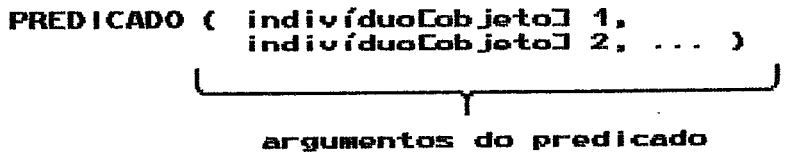
4a. linha t.v. - Se a máquina do carro está com defeito, então eu não posso dirigir hoje.
A ---> B é verdade

Embora sendo a lógica proposicional uma alternativa de representação de conhecimento, ela não é muito utilizada na inteligência artificial. A lógica proposicional se relaciona com declarações e se elas são verdadeiras ou falsas, esta habilidade para representar o mundo real é limitada.

Uma opção adotada em IA é o uso da **lógica de**

predicado. A lógica de predicados é uma forma mais sofisticada que utiliza os mesmos conceitos e regras da lógica proposicional. Também conhecida como cálculo de predicados, ela fornece maior habilidade para a representação do conhecimento em detalhes. O cálculo de predicado permite dividir uma declaração em partes, isto é um objeto, uma característica do objeto ou alguma afirmação sobre o objeto. O cálculo de predicado permite separar uma declaração ou proposição e se fazer afirmações sobre seus objetos e, ainda, permite o uso de variáveis e funções de variáveis. O resultado é um esquema mais poderoso de representação do conhecimento para a solução de problemas com o uso do computador.

No cálculo de predicados, uma proposição ou premissa é dividida em duas partes: o argumento (ou objeto) e o predicado (ou assertiva). Os argumentos são indivíduos ou objetos de uma proposição. O predicado é a parte da proposição que faz uma assertiva sobre os indivíduos ou objetos (os argumentos) e pode ser escrito como a seguir:



Por exemplo:

" O carro está na garagem. "

Poderia ser escrito da seguinte forma:

IN (carro,garagem) IN=está

IN = produto(afirmação)

carro = argumento(objeto)

garagem = argumento(objeto)

Outro exemplo:

Proposição : João gosta de Maria.

Expressão no cálculo de predicados:

GOSTA (joão,maria)

Um outro formato utilizado para representar é:

(está carro garagem) ou (gosta joão maria)

O predicado junto com seus argumentos é uma proposição. Todas as operações da lógica proposicional podem ser aplicadas a predicados.

O conhecimento contido no cálculo de predicados não está somente em palavras mas também nas sequências delas (isto é, predicado, indivíduo 1, indivíduo 2).

Em alguns casos a proposição pode ter somente um argumento. Alguns desses casos são:

. A porta está aberta.

ABERTA (porta)

. O pneu está furado.

FURADO (pneu)

. Chris é um homem.

homem (chris)

No último caso, o predicado "homem" é chamado uma função e é escrito em letras minúsculas.

Funções são diferentes de predicados. Uma função apresenta um relacionamento, enquanto o predicado expressa uma condição sobre o argumento.

No cálculo de predicados, as letras podem ser substituídas pelos argumentos. O símbolo "x" ou "y" pode ser usado para designar algum objeto ou indivíduo.

O exemplo dado "João gosta de Maria" poderia ser expresso em variável, onde x=joão e y=maria. Dessa forma a proposição poderia ser escrita como abaixo:

GOSTA(x,y)

Se variáveis são utilizadas, então a proposicão deve ser verdade para alguns nomes substituídos pelas variáveis.

Usando este sistema, uma base de conhecimento pode ser formada. O conhecimento expresso em cálculo de predicados, pode ser manipulado para a geração de inferências. Um exemplo simples:

João gosta de Maria.

Ramon gosta de Maria.

João=x , Maria=y , Ramon=z

"GOSTA(x,y) e GOSTA(z,y) implica não GOSTA(x,z)"

Nesta expressão se diz que se João gosta de Maria

e Ramon gosta de Maria, então João não gosta de Ramon.

Um quantificador é um símbolo que permite se declarar um alcance para as variáveis numa expressão lógica. Dois quantificadores básicos são utilizados na lógica: $\forall$, chamado de quantificador UNIVERSAL que tem o significado de "para todo"; $\exists$, chamado quantificador EXISTÊNCIAL, que significa "existe um". Por exemplo:

. $(\forall x) P(x)$

  significa: todo x satisfaz P.

. $(\exists x) P(x)$

  significa: existe um x que satisfaz P.

Para ilustrar melhor o uso desses quantificadores e o poder de sua representação, apresenta-se os seguintes exemplos:

. "Todos amam Maria." $(\forall x)$ AMA(x,maria)

. "Todos amam todos." $(\forall x)(\forall y)$ AMA(x,y)

. "Todos amam a si mesmo." $(\forall x)$ AMA(x,x)

. "Todos amam alguem." $(\forall x)(\exists y)$ AMA(x,y)

. "Existe alguem que todos amam." $(\exists y)(\forall x)$ AMA(x,y)

Os quantificadores são utilizados para determinar quando uma coisa é verdadeira e em que condições ocorre.

Essas expressões lógicas feitas de variáveis, predicados, quantificadores e conectivos como mostrados acima são referenciados como "fórmulas bem-formadas (Wff)".

O que foi mostrado até aqui, é um sistema que

permite se expressar fatos e conhecimentos numa forma simbólica. O que se espera realmente é utilizar o conhecimento para se fazer inferências. Como utilizar esse conhecimento para responder perguntas e fazer inferência? Utilizando várias regras de inferência para manipulação de expressões lógicas criando-se novas expressões. Se a proposição original é verdade, então a conclusão também será verdadeira. Por exemplo, uma regra de inferência bastante utilizada é a MODUS PONENS. Essa regra diz que se uma proposição A é verdadeira e se A implica B também é verdade, então a proposição B também é verdadeira.

Isto é expresso logicamente da seguinte maneira:

[A e (A implica B)] implica B

A e (A--->B) são proposições numa base de conhecimentos. Dada a expressão acima, pode-se recolocá-la com a expressão B. Ou seja, pode-se usar MODUS PONENS para obter a conclusão que B é verdade se as duas primeiras expressões são verdadeiras. Por exemplo:

A - está ensolarado

B - nós iremos a praia

A ---> B - Se está ensolarado, então nós iremos a praia.

A primeira premissa simplesmente declara que é um dia ensolarado. A segunda diz que nós iremos a praia. Além disso, A implica B.

Então se A e A implica B são verdades, B é verdadeira. Usando "modus ponens" pode-se então deduzir que

se irá a praia.

Essas deduções parecem bastantes naturais e comuns de serem obtidas, entretanto não o são para um computador. Simples deduções são feitas naturalmente pela inteligência do ser humano, porém para um computador deverá ser dito como fazer uma inferência simples como "modus ponens".

Numerosas técnicas são usadas para fazer deduções de premissas dadas ou para provar se uma proposição é verdadeira ou falsa. Todas elas requerem um grande conhecimento de lógica e sua implementação é complexa.

III.3.2 - REDES SEMÂNTICAS. (QUILLIAN-68) (BRACHMAN-83) (FRENZEL-87) (LUCENA-87) (RICH-88)

Muitos sistemas de representação do conhecimento baseados em Redes Semânticas, trazem a noção de uma hierarquia taxonômica explícita, uma estrutura em forma de rede ou de árvore que caracteriza as classes de elementos num domínio. A espinha dorsal da hierarquia reside num tipo de conexão de herança entre os objetos de representação, conhecida como **nós** em alguns sistemas e como **quadros** em outros. Esta conexão, frequentemente chamada de **IS_A** (também conhecida como IS, SUPERC, AKO, SUBSET, etc...), tem sido, provavelmente, o elemento mais estável das redes semânticas através dos anos, conforme (BRACHMAN-83).

A Rede Semântica, é uma forma de representação do conhecimento fácil e simples de ser entendida. Sua notação para representar o conhecimento é através de um gráfico contendo objetos e dos relacionamentos desses objetos.

Numa Rede Semântica os objetos são representados por nós (círculos ou retângulos), que descrevem as informações sobre tal objeto, e a relação entre os nós é representada através de uma ligação (arcos rotulados) desses nós, descrevendo um relacionamento. Um exemplo simples de uma rede semântica é apresentado na figura (III.5).

Os objetos numa Rede Semântica podem ser qualquer ítem físico como um livro, carro, cadeira ou também uma pessoa. Aos objetos pode-se acrescentar características que os descreva, através da inserção de uma descrição numa estrutura de dados, que permita o uso e programação.

Os relacionamentos representados por arcos significam o tipo de relação que envolve os objetos. Alguns arcos mais comuns em redes semânticas é o "É-UM" e "TEM-UM" e outros mais específicos como "FAZ-PARTE", "É-PARTE" e outros.

A rede semântica é um método bastante flexível de representação do conhecimento. São muito poucas as restrições impostas por esse método, permitindo definir uma grande variedade de objetos, atributos, conceitos e relacionamentos entre eles.

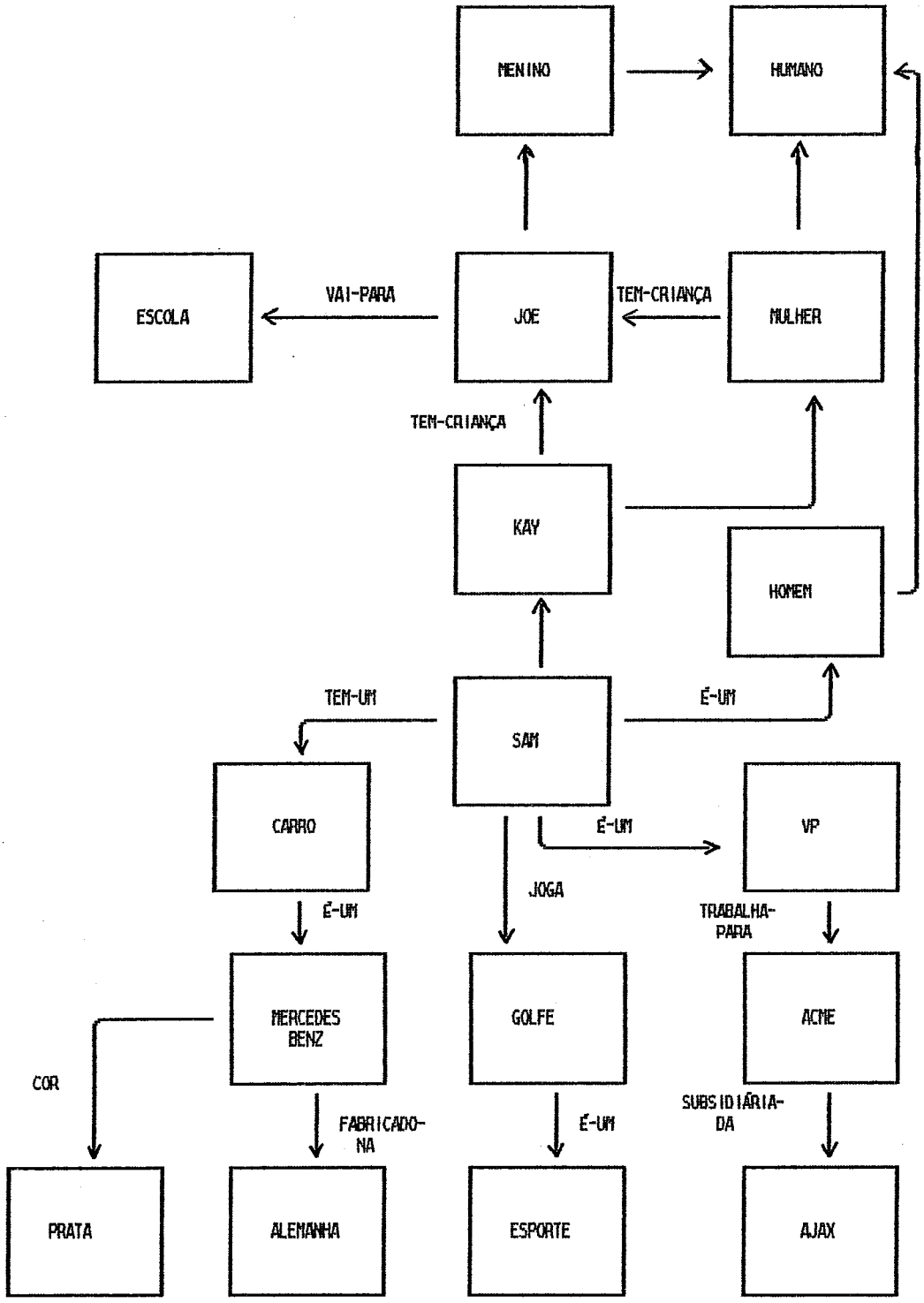


Figura III.5: Exemplo de uma rede semântica (FRENZEL-87).

Como ilustração a figura (III.6) apresenta mais um exemplo de rede semântica, destacando outros tipos de relações dos apresentados no exemplo anterior, onde o seu entendimento se torna bastante claro visualmente.

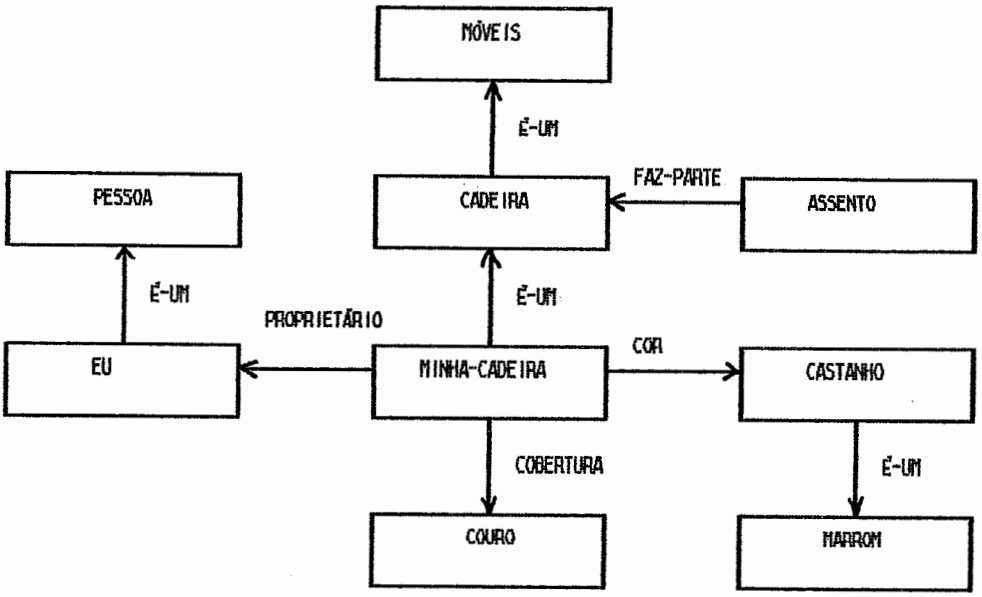


Figura III.6: Uma rede semântica (RICH-88).

Utilizando, por exemplo, a linguagem LISP como o meio de manipulação do conhecimento acima representado, cada nó da rede seria em LISP um átomo, os elos seriam propriedades e os nós, nas outras extremidades dos elos, seriam os valores. A rede acima apresentada seria representada em LISP como mostrado abaixo na figura (III.7).

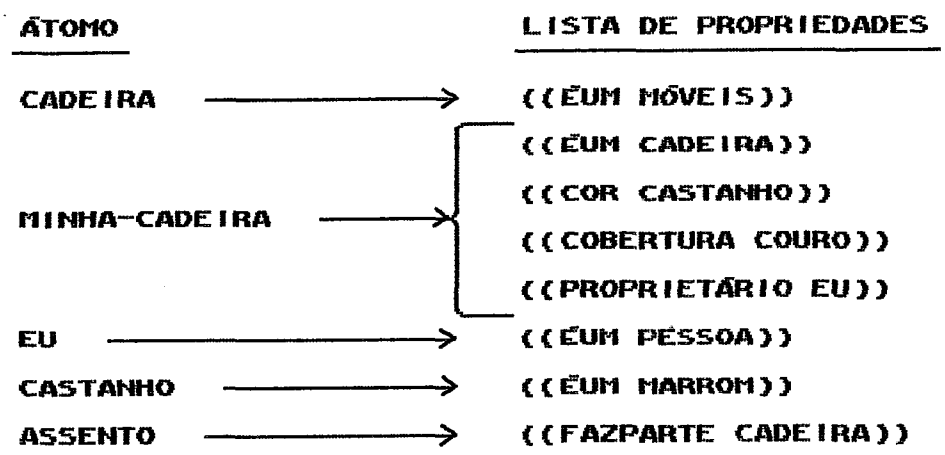


Figura III.7: A representação LISP de uma rede semântica (RICH-88).

Uma outra característica da rede semântica é a capacidade de representar relações da lógica de predicados, fazendo com que cada relação na rede apareça como predicado de dois argumentos. Por exemplo, alguns dos arcos da figura (III.9) poderiam ser representados na lógica como:

ÉUM (cadeira, móvel)

ÉUM (eu, pessoa)

COBERTURA (minha-cadeira, couro)

COR (minha-cadeira,castanho)

Consequentemente, predicados de N-argumentos podem ser representados numa rede semântica, utilizando o recurso da criação de nós adequados com arcos descrevendo a relação desejada.

Uma característica importante das redes semânticas, segundo (MYLOPOULOS-83), é seu ponto de vista associativo, o qual admite uma representação gráfica óbvia

que pode ser usada para definir caminhos de acesso conceituais ou de implementação. Alguns desses caminhos podem ser, e de fato têm sido, usados para organizar redes semânticas ao longo de diferentes domínios de aplicação. Entretanto, sua maior desvantagem é que seus elaboradores dependem mais das intuições advindas dos rótulos de conexão e do nó, do que da semântica formal.

Embora sejam claras e objetivas as características das redes semânticas, elas guardam conceitos fundamentais, desenvolvidos ao longo de sua experiência, que permitem o uso desse método de forma ampla em diversos domínios de aplicação.

Com relação aos domínios, eles afetam a característica da rede, em função dos nós e relacionamentos necessários e apropriados para a representação de seu conhecimento.

Nas figuras (III.5) e (III.6) já apresentadas, observamos que os nós e relacionamentos são de funções diferentes, em sua maioria, de forma necessária para apresentar o conhecimento desejado conforme seu domínio.

Assim, no surgimento das redes semânticas, os pesquisadores observaram que as duas formas de afirmação mais predominantes utilizadas por sistemas de representação de conhecimento eram a PREDICAÇÃO, expressando que um indivíduo (por exemplo, João) era de um certo tipo (por exemplo, homem solteiro), e a CONDICIONAL QUANTIFICADA UNIVERSALMENTE, expressando que um tipo (por exemplo, cão)

era sub-tipo de outro (por exemplo, mamífero). A maneira mais fácil de se colocar tais afirmações em um sistema de rede semântica, era possuir uma conexão que representasse diretamente as partes "é-um" (IS-A) de tais sentenças, nascendo assim a conexão **IS_A**.

A partir daí, se observou que as conexões IS-A formavam uma hierarquia (ou, em alguns casos, uma rede) dos tipos que estavam sendo conectados. A organização hierárquica facilitou a distribuição das "propriedades" de modo que aquelas que estivessem sendo divididas fossem armazenadas na hierarquia no local que cobrisse o maior sub-conjunto de nós que as estivessem dividindo. Sua habilidade de serem "herdadas" por todos os nós abaixo daqueles em que estão armazenadas é o que se denomina a noção de **herança de propriedades**, virtualmente tão mencionada quanto a conexão IS-A.

Numa notação gráfica típica daqueles que trabalham com redes semânticas, a distribuição de propriedades numa hierarquia , onde as propriedades que são comuns a mais de um conceito aparecem num nível geral, os relacionamentos são omissos, a favor dessa propriedade representar o conhecimento desejado.

No entanto, várias pesquisas sobre redes semânticas sugeriam, entre outras coisas, que não havia apenas uma conexão IS-A. (BRACHMAN-83) numa busca do que se pretende que signifiquem as conexões IS-A, ressalta que a estrita associação entre herança e IS-A, serve apenas para confundir as coisas. Só se pode esclarecer as alegações em

torno da IS-A colocando a herança em perspectiva (ela é um elemento de "implementação" e não de força expressiva).

Uma divisão básica das variedades das conexões IS-A, é entre interpretações genéricas e individuais de nós. Por alto, pode-se dizer que alguns nós são descrições aplicáveis a muitos indivíduos, enquanto que outros são representações ou daqueles indivíduos ou de descrições aplicáveis a um único indivíduo. Desta forma, divide-se a relação IS-A em dois sub-tipos principais, um que relaciona dois nós genéricos e outro que relaciona um indivíduo e um gênero. Para cada uma dessas relações, tem-se no mínimo os seguintes tipos de uso, conforme mostrado abaixo.

GENÉRICO/GENÉRICO:

1) Subconjunto/superconjunto
2) Generalização/especialização
3) AKO ("um tipo de")
4) Conteúdo conceitual
5) Restrição de valor
6) Conjunto

GENÉRICO/INDIVIDUAL:

1) Membro de conjunto
2) Predicação
3) Conteúdo conceitual
4) Abstração

Quando dois genéricos são relacionados por uma conexão IS-A, normalmente o propósito é que um esteja, de alguma forma, relacionado ao outro, mas que seja menos

geral.

Quando os nós representam conjuntos, a conexão entre dois nós genéricos representa a relação de subconjunto. Por exemplo, ao se construirem Submarinos Nucleares e Submarinos para representar os conjuntos de, respectivamente, todos os submarinos com força nuclear e todos os submarinos, "um Submarino Nuclear é um Submarino" significa que "para toda entidade x, se x for um membro de Submarinos".

A generalização, uma relação entre predicados, parece ser exprimível como uma condicional simples. Por exemplo, se Submarino (x) for um predicado considerado como a generalização de Submarino Nuclear (x), então a IS-A entre eles significa que "para toda entidade x, se Submarino Nuclear (x), então Submarino (x)".

Significando "um tipo de", AKO permite a relação entre "camelo" e "mamífero" em "o camelo é um tipo de mamífero". Embora AKO tenha muito em comum com a generalização, implica no "status" "tipo" para os nós conectados, ao passo que a generalização relaciona predicados arbitrários.

Para o caso do conteúdo conceitual, o propósito de uma conexão IS-A é expressar o fato de que uma descrição inclui outra. Ao invés de ler "um triângulo é um polígono" como uma simples generalização (isto é, há triângulos e polígonos, e esta é a relação entre eles), queremos ler que "ser um triângulo é ser um polígono com três lados", na

qual se utiliza um predicado para definir outro.

Uma outra relação entre genéricos, no caso a restrição do valor do papel, é a existente em "a tromba de um elefante é um cilindro com 1,3 metros de comprimento", ilustrando que, nas linguagens de representação, alguns termos genéricos referem-se a papéis ou "escaninhos" e não apenas a tipos.

Conjunto e seu tipo característico, é a relação que existe entre o conjunto de todos os elefantes e o conceito de um elefante.

Uma conexão IS-A entre um indivíduo e um genérico normalmente significa que um indivíduo pode ser descrito por alguma descrição genérica. Esta relação é frequentemente denominada instância. Para um membro de conjunto, se o genérico for interpretado como um conjunto, a relação é de membro. "Clyde é um camelo" significa "Clyde é um membro de (do conjunto de) camelo".

O uso da predicação, simplesmente atribui um predicado a um indivíduo, normalmente envolvendo um predicado de tipo como Camelo ou Submarino. Se o gênero for Camelo e o indivíduo for Clyde, a conexão IS-A expressa o fato de que Camelo (Clyde).

Para o conteúdo conceitual, quando o nó individual é considerado como uma descrição estruturada, o conteúdo conceitual pode ser a relação entre ele e um genérico. Esta é, por exemplo, a relação entre "rei" e "o

rei da França", onde a descrição genérica é utilizada para construir a descrição individual.

Outra relação existente entre um indivíduo e um gênero vai na direção oposta. Esta é a relação de abstração, na qual um tipo genérico é abstraído de um indivíduo, evidenciada em construções da linguagem natural como "a águia" em "a águia é uma espécie em extinção". A relação existe entre o indivíduo, A-águia e o predicado ou tipo (genérico), Águia.

Uma importante observação deixada por (BRACHMAN-83) é que a herança de propriedades nas conexões IS-A não representou nenhum papel para o entendimento do conhecimento expresso na rede. Embora muito tenha sido dito a respeito do significado da herança em redes semânticas, ninguém foi capaz de mostrar que ela faz alguma diferença na força expressiva do sistema que a anuncia. No máximo, as argumentações sobre a utilidade da herança foram feitas no terreno do pragmatismo: ela economizaria espaço de armazenamento numa implementação ou localizaria a informação a ser modificada.

Sem negar a importância do que diz respeito à implementação, admite-se que a herança é uma propriedade útil, mas que serve estritamente à implementação, sem trazer peso algum para a discussão sobre a superioridade comunicativa ou expressiva das redes semânticas.

III.3.3 - DEPENDÊNCIA CONCEITUAL. (SCHANK-77) (RICH-88)

A dependência conceitual, conhecida simplesmente como CD, é a teoria de como representar o significado de frases em linguagem natural de modo a:

. Facilitar a geração de inferências das frases.

. Ser independente da linguagem em que a frase foi originalmente declarada.

Baseada nesses objetivos, a representação CD de uma frase é construída não de primitivas correspondentes às palavras utilizadas na frase, mas de primitivas conceituais que podem ser combinadas para formar os significados de palavras em qualquer linguagem em particular.

Um exemplo simples nesta representação é dado na figura (III.8).

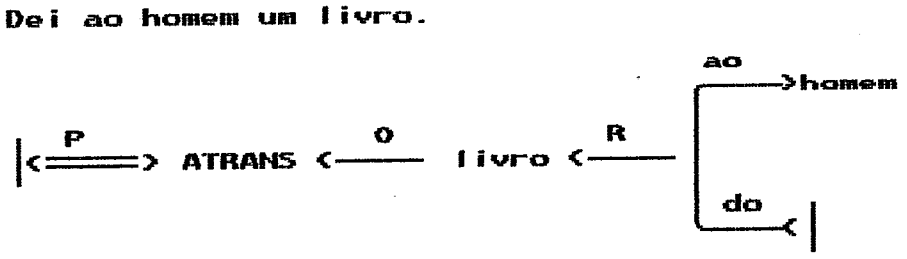


Figura III.8: Um exemplo de representação CD (RICH-88).

Os símbolos nessa representação possuem os seguintes significados:

. Setas, indicam a direção da dependência.

. Seta dupla, indica um elo de mão dupla entre o ator e a ação.

. p indica tempo passado.

. ATRANS, é um dos atos primitivos utilizados pela teoria. Ele indica transferência de posse.

. o, indica a relação de caso do objeto.

. R, indica a relação de caso do recipiente.

Uma representação em CD é construída através de blocos de construção, contendo um conjunto de primitivas.

As ações primitivas disponíveis variam de autores para autores. Um conjunto típico é o de SCHANK apresentado por (RICH-88).

. ATRANS Transferência de uma relação abstrata (por exemplo, dar)

. PTRANS Transferência da localização física de um objeto (por exemplo, ir)

. PROPEL Aplicação de força física a um objeto (por exemplo, empurrar)

. MOVE Movimento de uma parte do corpo de alguém (por exemplo, chutar)

. GRASP Segurar um objeto por um ator (por exemplo, jogar)

. INGEST Ingestão de um objeto (por exemplo, comer)

. EXPEL Expulsão de algo do corpo de um animal (por exemplo, chorar)

. MTRANS Transferência de informação mental (por exemplo, dizer)

. MBUILD Construir nova informação da antiga (por exemplo, decidir)

. SPEAK Produção de sons (por exemplo, dizer)

. ATTEND Enfocar o órgão de sentido em um estímulo (por exemplo, ouvir)

Um segundo conjunto de blocos de construção é o de dependências permitidas entre as conceitualizações descritas numa frase. Há quatro categorias de conceitualizações primitivas cujas estruturas de dependência podem ser construídas.

Acts	Ações
PPs	Objetos (produtores de imagem)
AAs	Modificadores de ações (auxiliadores de ação)
PAs	Modificadores de PPs (auxiliadores de imagem)

As conceitualizações que representam eventos podem ser modificadas de muitos modos para suprir informações normalmente indicadas na linguagem pelo tempo, modo ou aspecto da forma do verbo. O conjunto de tempos conceituais propostos por (SHANK-75) inclui:

p - Passado

f - Futuro

t - Transição

ts - Início de transição

tf - Fim de transição

k - Continuidade

? - Interrogativo

/ - Negativo

nil - Presente

delta - Intemporalidade

c - Condicional

Como um exemplo desses tempos, considere a representação CD da figura (III.9).

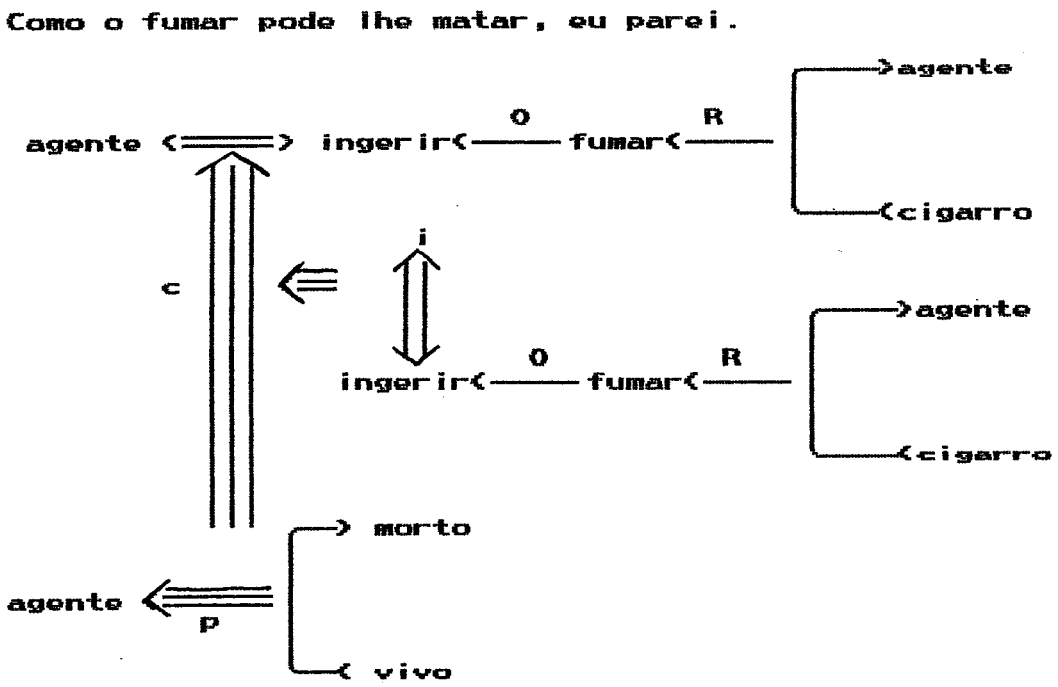


Figura III.9: Utilização de Tempos Conceituais em CD (RICH-88).

O elo de causalidade vertical indica que o fumar mata a gente. Como ele está marcado com c, entretanto, sabemos apenas que o fumar pode matar a gente, não que ele necessariamente o faça. O elo de causalidade horizontal indica que é essa primeira causalidade que me fez parar de fumar. A qualificação tfp afixada à dependência entre EU e INGERIR indica que o fumar (um caso de ingerir) parou e que a parada ocorreu no passado.

Há três meios importantes através dos quais a representação de conhecimento, usando o modelo de dependência conceitual, facilita o raciocínio com o conhecimento:

1 - São necessárias menos regras de inferência do que seriam se o conhecimento não fosse decomposto em primitivas.

2 - Muitas inferências já estão contidas na própria representação.

3 - A estrutura inicial, construída para representar a informação contida em uma frase, terá espaços que precisam ser preenchidos. Esses espaços podem servir como focalizadores de atenção para o programa que deverá compreender as frases posteriores.

A representação CD não é a única dessas teorias relacionada com a dependência de ações e, merece maiores discussões sobre suas vantagens e desvantagens.

III.3.4 - QUADROS. (MINSKY-75) (BRACHMAN-83A) (FRENZEL-87) (LUCENA-87) (RICH-88)

O quadro tem a característica de representar o conhecimento sobre um objeto, evento e situações estereotipadas em geral. Um conhecimento representado por este método e, já se encontrando armazenado e em uso, pode necessitar de modificações que alterem o conhecimento já armazenado, resultando numa forma de representação a partir de experiências anteriores.

O quadro é utilizado normalmente para representar conhecimentos estereotipados ou conhecimentos baseados em características e experiências bem conhecidas, permitindo um grau elevado de detalhes sobre o que se está representando.

Uma forma de raciocinar com este método é a procura sucessiva de experiências sobre um objeto, evento ou situação, até se obter o padrão desejado.

Desta forma torna-se mais fácil se fazer inferências acerca de novos objetos, eventos ou situações a partir de uma base de conhecimento já existente.

Um quadro é formado por "espaços a serem preenchidos (escaninhos)" para um determinado objeto e relações que são inerentes a uma certa situação. Associadas a cada quadro encontram-se as seguintes informações:

. como usar o quadro;

. o que fazer se ocorrer algo inesperado;

especificação de valores padrão para os espaços reservados ao quadro. Associado a esses espaços (escaninhos) existe um conjunto de condições que devem ser satisfeitas pelo objeto que vai preenchê-lo (caso contrário usam-se os valores de omissão).

Um exemplo do uso desse método é apresentado na figura (III.10), que busca representar um objeto sob vários pontos de vista.

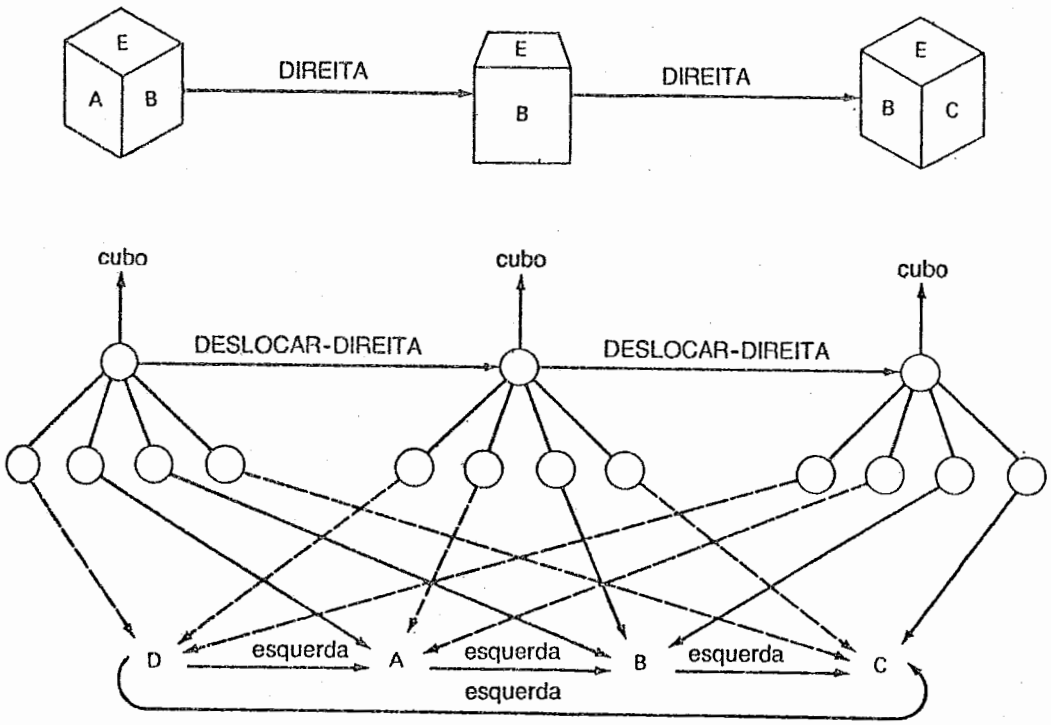


Figura III.10: Um sistema de quadro (RICH-88).

O exemplo anterior apresenta a perspectiva de um cubo em três visões. As linhas sólidas dos escaninhos representam os lados visíveis na visão do objeto. As linhas

serrilhadas indicam visões que são invisíveis daquele ponto de vista. Os elos entre os quadros descrevem a relação entre os pontos de vista que os quadros representam.

Um outro exemplo do uso do quadro é a descrição de um automóvel mostrado na figura (III.11) abaixo.

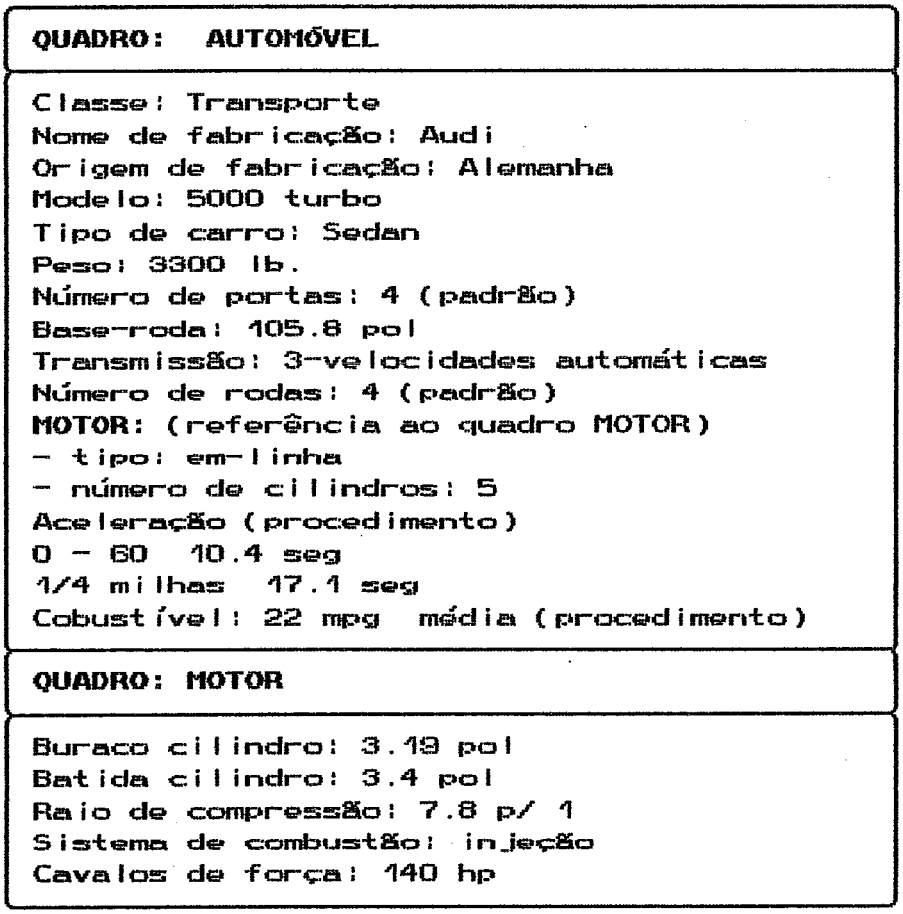


Figura III.11: Exemplo de uma representação em quadro (FRENZEL-87).

Observe que, os escaninhos estão descrevendo atributos do automóvel tal como nome do fabricante, modelo, origem do fabricante, tipo de carro, números de portas, motor e outras características. Note, ainda, que alguns dos escaninhos contém valores "de omissão". Por exemplo, um valor de omissão é que o número de rodas são quatro. Um valor de omissão significa que se pode assumir que o carro tem quatro rodas a menos que seja informado outro valor para esse atributo.

Um outro tipo de escaninho é de um procedimento dedicado, permitindo acrescentar novas informações às já existentes.

Neste exemplo, a aceleração e a média de combustível por milha são procedimentos dedicados. Um exemplo de um procedimento dedicado seria o de calcular o consumo por milha. Este procedimento controlaria o enchimento do tanque, em certo tempo de funcionamento por número de milhas, determinaria a quantidade de combustível usada, e então obteria a média por milha em termos de milhas por galão.

Um outro tipo de escaninho é um que faça referência a outro quadro. Por exemplo, o escaninho motor da figura (III.11) faz referência a descrição de seus atributos no quadro motor.

Muitos sistemas de IA que utilizam o método do quadro, são elaborados com uma coleção de quadros ligados hierarquicamente. Quando uma estrutura de quadros é

utilizada, então também um quadro deve herdar propriedades de outros quadros. Na figura (III.12) a seguir, é apresentado um esquema de representação utilizando o método dos quadros numa estrutura hierárquica.

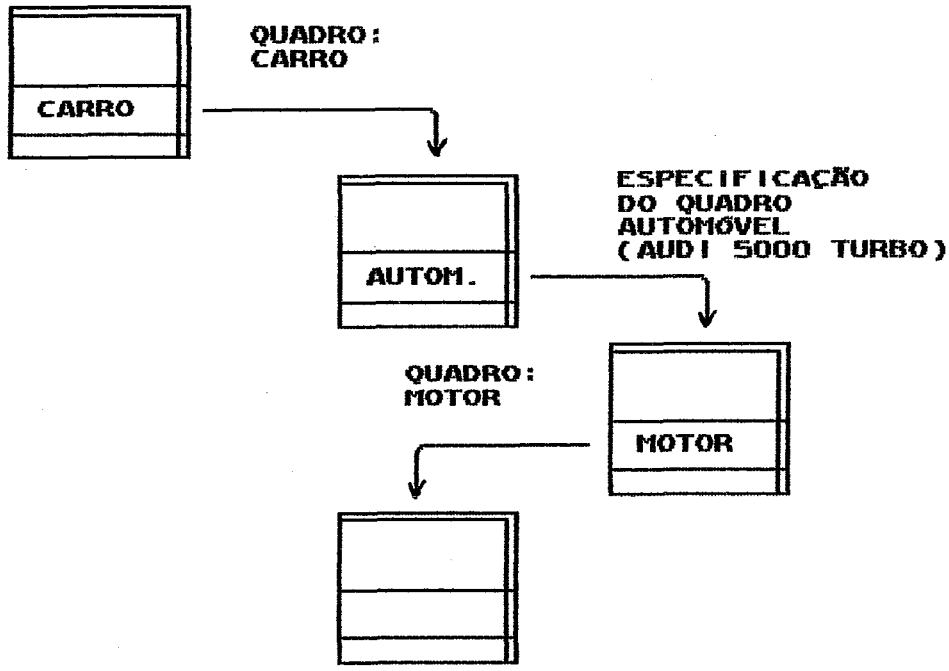


Figura III.12: Conhecimento representado numa hierarquia de quadros (FRENZEL-87).

Uma tendência que se tem observado, é a pesquisa centrada no desenvolvimento de linguagens baseadas em quadros, com as seguintes características:

. Os principais objetos de representação, ou quadros, são descrições não atômicas um tanto complexas.

. Os quadros são definidos como especializações de quadros mais gerais.

. Os indivíduos são representados por instanciações de quadros gerais.

. As conexões entre quadros resultantes formam taxonomias.

O grande apelo das taxonomias de quadros parece ser devido a sua proximidade com nossas intuições a respeito da maneira de estruturar o mundo. Os quadros também sugerem direções sedutoras para o processamento (herança, omissão, etc...) e encontraram aplicações em outras áreas, tais como, controle de base de dados e programação orientada por objeto.

(BRACHMAN-83A) coloca que normalmente surgem complicações na elaboração e uso dos quadros, são elas:

. as estruturas são interpretadas de diferentes maneiras em diferentes momentos (sendo que a principal ambiguidade situa-se entre as interpretações factuais e de definição); e,

. o significado da linguagem de representação é especificado somente em termos das estruturas de dados usadas para implementá-la (tipicamente redes de herança).

Essas questões vêm sendo tratadas em função das aplicações dos quadros para representação de conhecimento. O fato de se lidar com um domínio específico, o que é o nosso caso, faz com que as estruturas tenham uma interpretação pré-orientada e, a linguagem de

representação, com facilidades voltadas para a utilização adequada dessas estruturas.

O sistema KRYPTON, apresentado em (BRACHMAN-83A), representa uma tentativa de se lidar diretamente com estes problemas, em termos de uma estratégia de elaboração funcional. Certos erros de utilização podem ser minimizados com a severa limitação da interface entre o usuário e base de conhecimento. O usuário é obrigado a se concentrar na utilidade de sua base de conhecimento, ao invés de se preocupar com os detalhes de implementação que sustentam esta funcionalidade.

O KRYPTON também defende a divisão de um sistema de representação em dois componentes distintos: terminológico e assertivo. O componente terminológico sustenta a formação de descrições estruturadas organizadas taxonomicamente, enquanto que o componente assertivo permite que estas descrições sejam usadas para caracterizar algum domínio de interesse. Em ambos os casos, tem-se uma linguagem de composição que é usada para interagir com uma base de conhecimento.

III.3.5 - ROTEIROS. (SCHANK-77) (FRENZEL-87) (RICH-88)

O roteiro, é um método de representação de conhecimento parecido com o quadro, porém ao invés da descrição de um objeto, o roteiro descreve uma sequência de eventos. Da mesma forma que o quadro, o roteiro retrata uma situação estereotipada.

Para descrever uma sequência de eventos, o roteiro utiliza uma série de escaninhos contendo informações sobre pessoas, objetos, e ações que estão envolvidas no evento.

O roteiro apresentado como exemplo na figura (III.13), representa os procedimentos tradicionais num restaurante. Neste roteiro, cada parte é constituída de componentes que permitem descrever os procedimentos e, ainda, definir seus atributos. Os componentes importantes do texto são:

. **Condições de entrada**:

Condições que, em geral, deverão ser satisfeitas antes dos eventos descritos no roteiro poderem ocorrer.

. **Resultado**:

Condições que, em geral, serão verdadeiras após os eventos descritos no roteiro terem ocorrido.

. **Apoios**:

Escaninhos representando objetos que estão envolvidos nos eventos descritos no roteiro. A presença desses objetos pode ser inferida mesmo se eles não forem citados explicitamente.

. **Papéis**:

Escaninhos representando pessoas que estão envolvidas nos eventos descritos no roteiro. A presença dessas pessoas também pode ser inferida, mesmo se elas não forem citadas explicitamente. Se

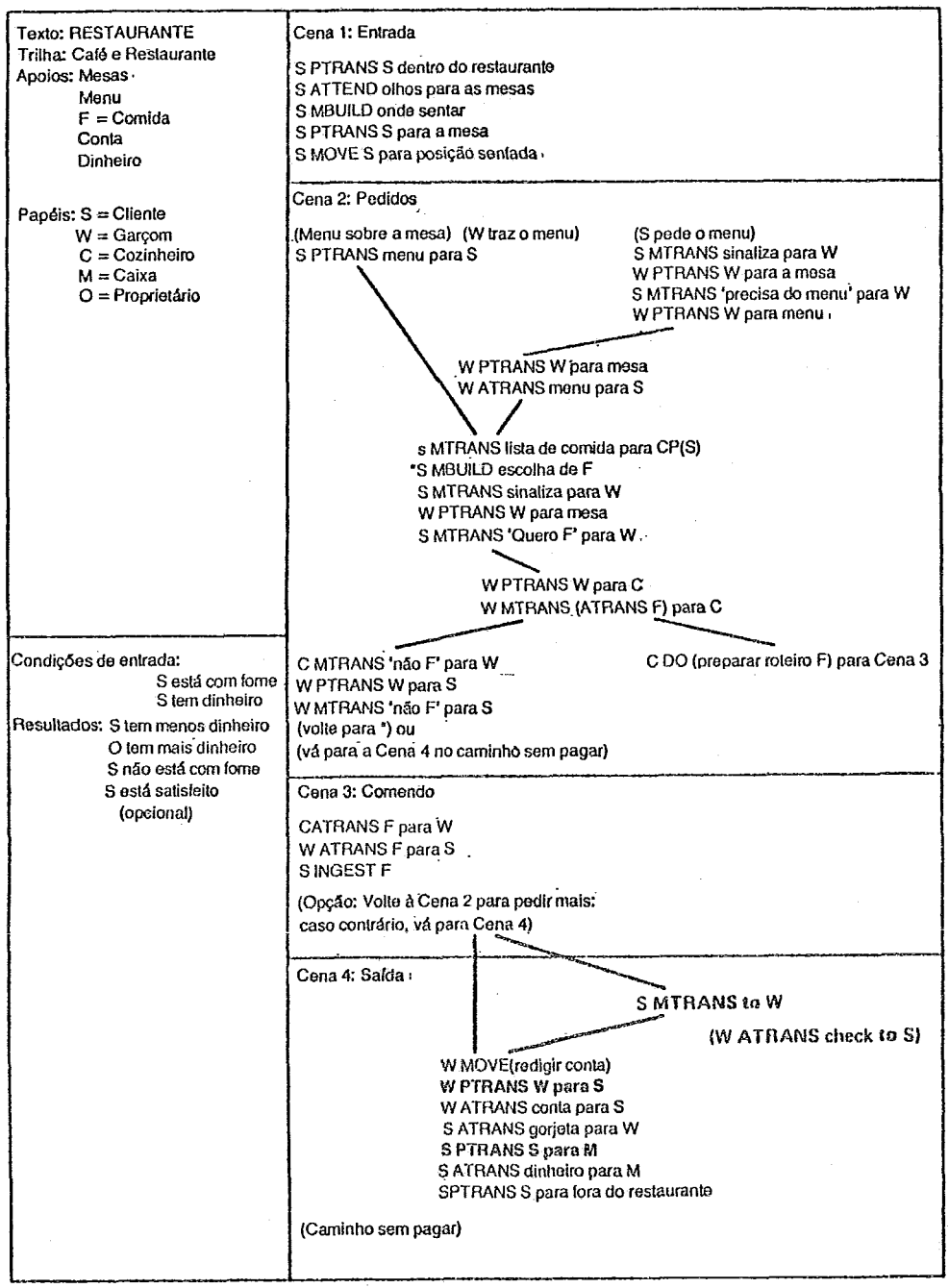


Figura III.13: Um exemplo do roteiro restaurante (RICH-88).

indivíduos específicos forem citados, podem ser inseridos nos escaninhos apropriados.

. **Trilha:**

Variação específica sobre um padrão mais geral que é representado por este roteiro em particular. Trilhas diferentes do mesmo roteiro partilharão muitos, mas não todos, os componentes.

. **Cenas:**

As sequências efetivas de eventos que ocorrem. Os eventos são representados no formalismo da CD.

Um roteiro é utilizado para predizer o que acontecerá em certas situações. Ainda que certos eventos não tenham sido observados, o roteiro permite ao computador predizer o que acontece, para quem e quando. Como no caso dos quadros, os roteiros são utilizados como forma de representar o conhecimento porque tratam de situações estereotipadas e eventos com os quais as pessoas lidam dia-a-dia.

III.3.6 - REGRAS DE PRODUÇÃO. (NEWELL-72) (FRENZEL-87) (LANGLEY-83)

A representação do conhecimento baseado em REGRAS DE PRODUÇÃO (ou sistema de produção), é um dos esquemas mais populares em IA. Refere-se a duas partes de uma declaração que incorpora pequenos pedaços do conhecimento: regra e produção. A primeira parte da regra é chamada de ANTECEDENTE que expressa uma situação ou premissa, enquanto

a segunda parte, chamada de CONSEQUENTE, expressa uma ação particular ou conclusão que são aplicadas se a situação ou premissa é verdadeira. As formas mais comuns de regras de produção seguem os formatos:

ANTECEDENTE	-	CONSEQUENTE
Situação Premissa		Ação conclusão

A primeira parte, lado esquerdo da regra, é uma declaração com o prefixo SE. A segunda parte, lado direito da regra, é uma declaração com o prefixo ENTÃO. Uma regra típica é ilustrada a seguir:

SE a quantidade do estoque de balas é inferior a 50
ENTÃO compre 100 porções.

A conclusão estabelecida na parte ENTÃO é válida se a parte SE da regra é verdadeira.

Regras de produção é uma das mais flexíveis formas de representação de conhecimento. Fáceis de se criar e entender, seu formato é compatível com o modo da mente humana armazenar e aplicar o conhecimento.

Uma grande quantidade de tipos de conhecimento podem ser representados no formato de regras de produção, permitindo converter o conhecimento humano num formato apropriado para representação em computador.

Alguns exemplos abaixo ilustram a flexibilidade

deste método representando diferentes tipos de conhecimentos.

1. SE a temperatura excede 25 graus
   ENTÃO ligue o ventilador para refrescar.

   Essa regra diz que se a temperatura estiver abaixo dos 25 graus então o ventilador não deverá ser ligado. Todavia, se a temperatura estiver acima, então o ventilador terá que ser ligado.

2. SE o carro não estiver regulado
   E a bateria estiver fraca
   ENTÃO o carro não vai pegar.

   Outra vez uma regra fácil de se entender. O ponto chave aqui é o uso da conjunção E. Aqui as duas condições prévias devem acontecer para que a ação seja satisfeita. Se ambas as condições não acontecerem, então a ação especificada não existirá.

3. SE um aluno teve nota na prova final maior que 8.0
   OU um aluno teve média final maior que 6.0
   ENTÃO o aluno é aprovado.

   Neste exemplo é importante observar o uso da função lógica OU (disjunção) entre as condições na parte SE da regra. Nesse caso um aluno é aprovado dependendo da nota na prova final ou da sua média final. Assim, para que seja aprovado basta que uma das condições seja satisfeita, independendo da

situação da outra.

Um sistema de regra de produção consiste de uma base de conhecimento, uma memória de trabalho e um mecanismo de controle ou inferência.

O exemplo mais comum de um sistema de produção é o sistema especialista, que utiliza as regras para codificar o conhecimento de um especialista num domínio particular, permitindo que um computador acesse esse conhecimento através de uma base de conhecimento nele armazenada.

Na inteligência artificial, é bastante utilizado a representação do conhecimento como um conjunto de regras de condição-ação, ou como sistemas de produção, como visto anteriormente. Esta abordagem, primeiramente proposta por (NEWELL-72) como um modelo da arquitetura cognitiva humana, foi usada em muitas áreas da IA, do processamento de linguagem aos sistemas especialistas (LANGLEY-83).

Um sistema de produção opera em ciclos. Em cada ciclo, as condições de cada produção são combinadas ao estado corrente da memória de trabalho. A partir de regras que combinam com sucesso, seleciona-se uma ou mais para aplicação. Quando se aplica uma produção, suas ações afetam o estado da memória de trabalho, fazendo com que novas produções combinem. Este ciclo continua até que mais nenhuma regra combine ou que se encontre um comando de parada.

Uma das características promissoras dos sistemas de produção é sua modularidade. Considerando que cada produção é relativamente independente das outras, podemos, a princípio, construir programas modulares que continuem a funcionar após a inserção de novas regras ou a remoção de regras antigas.

(LANGLEY-83) ressalta que a relação entre a modularidade e a aprendizagem é clara. A aprendizagem envolve a adição de conhecimento e o conhecimento deve ser adicionado de maneira que o novo conhecimento tenha boa interação com o conhecimento existente. Uma ilustração da integração da representação com a aprendizagem, pode ser visto no diagrama apresentado no capítulo (IV.3).

Embora a abordagem do sistema de produção possa não ser o único formalismo que ofereça modularidade, ele é um dos poucos modelos de representação testados com sucesso nesta dimensão.

III.3.7 - REDES NEURONAIS (PESSOA-89) (FRANÇA-89).

Redes neuronais (também chamadas de modelos conexionistas) é um outro modelo para se tratar com a representação do conhecimento.

Embora a história aponte sua origem na década de 50 com os trabalhos de Marvin Minsky e Dean Edmonds, a base das pesquisas foi com os trabalhos de McCulloch e Pitts em 1943 e de Donald Hebb em 1949.

Apresentaremos uma visão simplificada deste

modelo baseado em (PESSOA-89), com ênfase centrada na questão da representação do conhecimento.

Um dos objetivos mais importantes dos modelos conexionistas é tentar modelar tarefas inteligentes como o entendimento de linguagem natural e o reconhecimento de imagens. Sua característica básica é o tratamento mais próximo de como o ser humano raciocina para a solução de problemas. Ao invés de executar uma série de instruções sequencialmente, como um computador tradicional, modelos conexionistas exploram diversas hipóteses alternativas simultâneamente, utilizando redes altamente paralelas de diversos elementos computacionais simples conectados por ligações de pesos e variáveis.

Apesar da grande quantidade de características, modelos conexionistas podem ser especificados através de três elementos básicos: um conjunto de unidades de processamento, uma topologia de interligação e um esquema de aprendizado.

Conforme apresentado em (FRANÇA-89), é imprescindível o conceito de **estado de ativação** $a_i(t)$ do neurônio N_i no instante t, que sendo um número inteiro ou real dentro de determinados limites, fornece o grau de excitação ou inibição do neurônio. O conjunto de estados de ativação de todos os neurônios da rede neuronal é o seu **padrão de atividade**. Cada célula possui uma resposta ou saída $o_i(t)$ no tempo t em função de seu estado de ativação $a_i(t)$, por definição uma regra de saída f_i dada por $o_i(t)=f_i(a_i(t))$.

São as conexões entre os neurônios que determinam as características da rede neuronal, incluindo o que os conexionistas chamam de inteligência. Dessa forma, a especificação do padrão de conexão é essencial e pode ser realizada através do estabelecimento dos valores W_{ij} das eficiências sinápticas existentes entre o axônio do neurônio N_j e um dendrito do neurônio N_i. As eficiências sinápticas podem ser positivas, negativas ou nulas, representando correspondentemente sinapses excitatórias, inibitórias, ou a sua inexistência. O conjunto de valores W_{ij} pode ser coletivamente denotado W.

Também é preciso definir uma regra de propagação g_i, cuja tarefa é integrar os impulsos recebidos pelo neurônio N_i em um impulso total de entrada $u_i(t)$, que corresponderia ao potencial pós-sináptico do neurônio biológico. Geralmente a função g_i depende da saída o(t) e do padrão de conexão W, ou seja, $u_i(t)=g_i(o(t),W)$.

Ainda, o modelo conexionista necessita de uma regra de ativação h_i capaz de fornecer o novo estado de ativação $a_i(t+\_t)$ no instante de tempo futuro t+_t. De um modo geral, $a_i(t+\_t)=h_i(a_i(t),u_i(t))$.

Os modelos de redes neuronais possuem ainda uma regra de aprendizado que fornece os meios necessários para que as eficiências sinápticas sejam alteradas e a rede neuronal modifique sua "inteligência", ou seja, aprenda ou se adapte a novas situações.

Para ilustrar um exemplo prático de um modelo

conexionista, utilizaremos um sistema cujo objetivo é tratar a ambiguidade verbal de um subconjunto do português (PESSOA-89).

Um dos problemas mais complexos no processamento de linguagens naturais é o problema de resolver o significado específico de uma palavra dentre o conjunto de significados possíveis. Vejamos alguns significados do verbo "passar":

1) João passou a faca no pão

2) João passou no concurso

3) João passou o fim de semana no Rio

Desta forma o verbo "passar" possui três significados distintos.

Uma maneira de se resolver a ambiguidade é fazer uso do contexto como forma de auxiliar na decisão de que significado é o mais apropriado. Em muitos casos, o contexto da própria frase é suficiente para resolvermos o problema. Em outras situações, no entanto, é preciso saber o contexto mais geral no qual a frase está inserida. O exemplo mais simples a ser ilustrado resolve o problema da ambiguidade no contexto da frase.

Vejamos agora o sistema proposto para resolver a ambiguidade do verbo "bater", conforme figura (III.14).

O sistema possui onze unidades representadas por retângulos. Cada unidade possui um rótulo indicando o que a unidade representa. As unidades estão interligadas por dois

tipos de ligação: flechas são usadas para indicar ligações excitatórias (pesos positivos) e a linha com uma extremidade arredondada indica uma ligação inibitória (pesos negativos). No sistema em questão, encontramos linhas com ambas as extremidades arredondadas indicando uma inibição mútua.

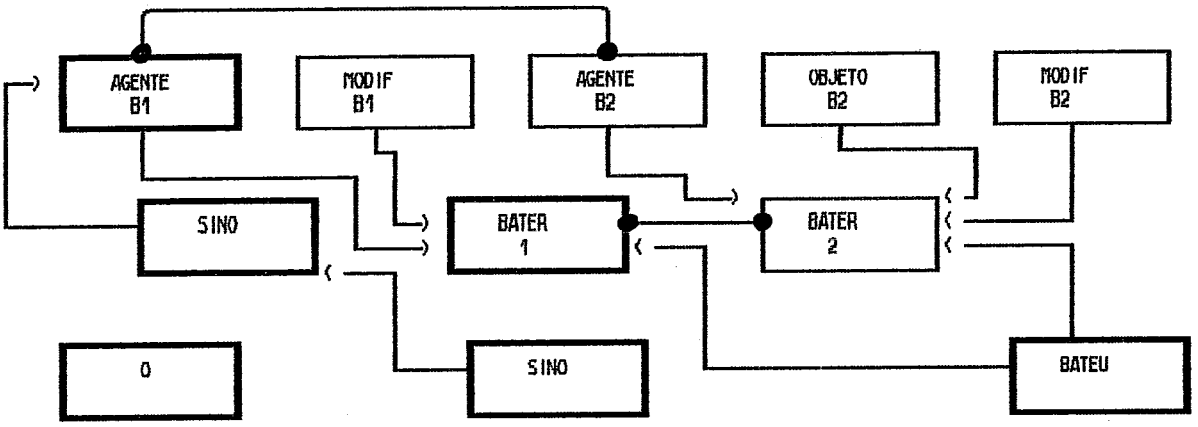


Figura III.14: Sistema para resolver a ambiguidade do verbo "bater" (PESSOA-89).

O sistema está estruturado em três níveis: léxico, significado das palavras e casos verbais. Cada unidade do nível léxico representa uma palavra existente e estas unidades são ativadas externamente, ou seja, constituem a entrada do sistema. No caso em questão, as palavras "o", "sino" e "bateu" foram "percebidas" e encontram-se ativadas (retângulo grosso).

As unidades do nível léxico possuem ligações com as unidades do nível de significado das palavras, onde cada unidade representa um significado possível da palavra. As unidades "bater-1" e "bater-2" são os dois significados conhecidos neste sistema. "bater-1" representa o significado usual de "bater", e "bater-2" representa "vencer alguém no jogo". Assim, a palavra "bater" do nível léxico está ligada aos dois possíveis significados do nível acima.

O último nível é o dos casos verbais (provenientes da linguística). Os casos utilizados são os seguintes:

. agente: aquele que executa ou é responsável pela ação;
. objeto: aquele que sofre ou é alterado pela ação;
. instrumento: instrumento utilizado na execução da ação;
. modificador: modo como foi executada a ação.

São os casos verbais que vão, em última instância, definir o significado correto do verbo em questão.

Mas como o sistema chega à conclusão de que o significado correto é "bater-1" ? A unidade "sino" do nível léxico, ao ser estimulada ativa a unidade "sino" do nível do significado. Esta unidade, por sua vez, ativa a unidade "agente b-1" que representa um agente inanimado. Mas "agente b-1" e "agente b-2" são mutuamente exclusivos e

desta forma "agente b-2" é inibido. Finalmente, "agente b-1" ativa "bater-1", o qual fica ativo dando o significado desejado. A unidade "modificador b-1" é opcional para a ativação de "bater-1", e não precisou estar ativa. Em outras palavras, a unidade "bater-1" ficou ativa mesmo sem que a unidade "modificador b-1" estivesse.

A partir da ilustração de um exemplo do modelo conexionista, indicamos os principais aspectos desse modelo, apresentados detalhadamente em (PESSOA-89), quais sejam:

. um conjunto de unidades de processamento;
. um estado de ativação;
. uma função de saída;
. um padrão de interconexão;
. uma regra de propagação;
. uma regra de ativação;
. uma regra de aprendizado.

Apresentaremos agora um outro exemplo cujo objetivo é ilustrar como implementar redes semânticas na abordagem conexionista. A maneira mais direta seria fazer uma correspondência direta entre os nós da rede semântica e as unidades de um modelo conexionista. Analogamente, as relacões entre os nós corresponderiam à ligações entre unidades. Esta forma corresponderia, basicamente, à abordagem local (seria então bastante semelhante ao sistema do exemplo anterior).

A estratégia proposta por Hinton, referenciada em

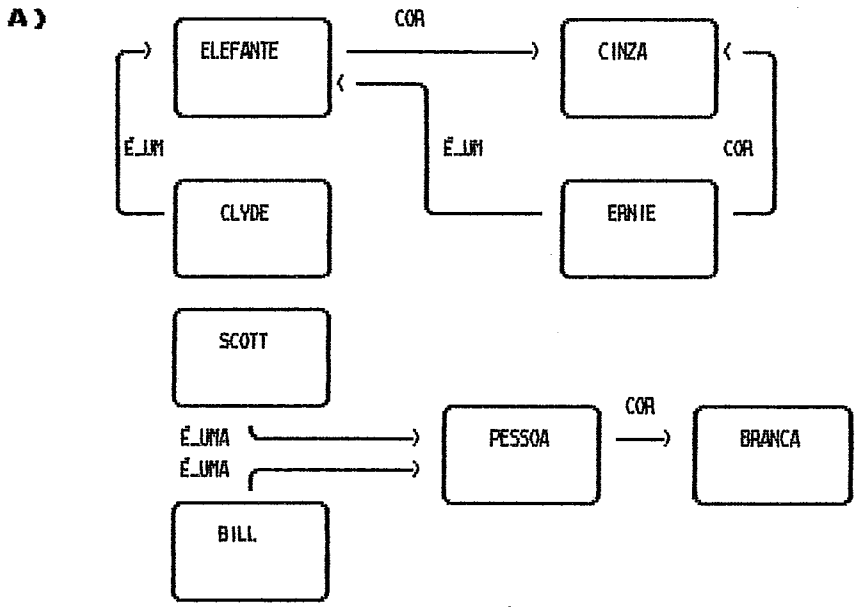
(PESSOA-89), para resolver este problema é bastante diferente. Nela cada nó da rede semântica corresponderia à um padrão de ativação específico em uma vasta coleção de unidades do sistema conexionista. Em outras palavras, Hinton adotou a abordagem distribuída. As diversas unidades do sistema representarão "microcaracterísticas" que juntas formarão os conceitos do sistema. Assim, diferentes nós da rede semântica serão representados por diferentes padrões de ativação no mesmo conjunto de unidades.

Hinton argumenta que ao formalizarmos as interações entre conceitos com um único elo estamos fazendo uma aproximação (de como os conceitos são realmente representados no sistema nervoso) que não leva em conta a importância das milhões de interações simultâneas no nível das microcaracterísticas, que segundo ele existem. Segundo Hinton, é desta forma que devemos entender como conceitos são lembrados, como relações entre eles mudam e como conceitos novos surgem.

É importante destacar que uma rede semântica consistindo de nós conectados por arcos direcionados rotulados possui a mesma informação de um conjunto de triplas, cada qual constituída de três campos correspondente aos dois nós e ao arco de rótulo, conforme figura (III.15) ilustrações "a" e "b". Neste mesmo raciocínio, para implementarmos uma rede semântica em um conjunto de unidades, dividimos o total de unidades em três grupos, correspondendo aos dois nós e ao arco (relacionamento). Os padrões de ativação destes grupos

serão usados para representar os dois nós e o arco de rótulo envolvidos na relação, figura (III.15) ilustração "c".

Uma característica bastante interessante de representações distribuídas, e consequentemente do sistema sendo ilustrado, é que propriedades como "generalização" e "herança" simplesmente emergem da organização do sistema, não precisando ser explicitamente implementadas. Nas abordagens tradicionais de IA, entretanto, é preciso prover mecanismos explícitos para obtermos estas propriedades.



B)

(ELEFANTE	COR	CINZA)
(ERNIE	COR	CINZA)
(CLYDE	É_UM	ELEFANTE)
(ERNIE	É_UM	ELEFANTE)
.		
.		

C)

111000	000000	ELEFANTE
000111	000000	PESSOA
111000	111000	CLYDE
111000	000111	ERNIE
000111	101010	SCOTT
000111	010101	BILL

Figura III.15: Uma abordagem conexionista para as redes semânticas (PESSOA-89).

III.4 - INFORMAÇÕES TECNOLÓGICAS.

Apresentaremos nesta seção, alguns sistemas de representação do conhecimento desenvolvidos e que utilizam alguns dos métodos descritos na seção anterior. Estes sistemas nos demonstram como foram utilizados os métodos de representação para um objetivo pré-definido e apresentam uma característica interessante, de associar mais de um método de representação num mesmo sistema.

III.4.1 - SISTEMA CENTAUR (BUCHANAN-84).

Para melhor compreender os objetivos e importância da representação do conhecimento, é apresentado um resumo, baseado em (BUCHANAN-84), do método de representação utilizado pelo sistema Centaur.

O Centaur é um sistema de consulta que realiza uma interpretação e produz um diagnóstico baseado num conjunto de resultados de testes. As informações de entrada para o sistema são os resultados dos testes das funções pulmonares e um conjunto de dados do paciente incluindo nome, sexo, idade e um diagnóstico inicial para referência. A saída consiste de um conjunto de declarações interpretadas, que explicam ou comentam os resultados dos testes das funções pulmonares, e um diagnóstico final da doença pulmonar do paciente.

O sistema Centaur utiliza hipóteses dirigidas em busca da solução do problema, onde as hipóteses são representadas por um protótipo. O objetivo do sistema é

encontrar um ou mais protótipos existentes num conjunto de protótipos, a partir dos dados fornecidos num caso atual. Os protótipos representam várias doenças pulmonares, seus rigores e sub-tipos, de tal forma que o protótipo confirmado representa o diagnóstico da doença pulmonar de um paciente.

A figura (III.16) apresenta um exemplo de uma interpretação de um conjunto de resultados de testes de um paciente.

No sistema Centaur o conhecimento é representado utilizando dois métodos de representação: regras de produção e quadros. Cada protótipo contém dois grupos de informações: **componentes** de um domínio específico que expressa a característica real de cada protótipo, e **escaninho** que especifica a informação usada na execução do sistema. Cada componente deve ter escaninhos de informações associados com ele, incluindo um escaninho de **regras** que permita o direcionamento dos componentes para regras que determinem os valores desses componentes.

As regras consistem de uma ou mais cláusulas de premissas seguidas por uma ou mais cláusulas de ações. Em geral, as cláusulas de premissas especificam um conjunto de **valores-limites** para alguns dos componentes de um protótipo, e as cláusulas de ação fazem conclusões sobre os valores de outros componentes. Junto dessas estruturas de dados, existem também estruturas de dados que fornecem informações sobre os valores atuais dos dados obtidos durante a consulta.

```
PROTÓTIPO CORRENTE: SOPRO

A hipótese corrente é o desejo de uma interpretação
dos testes de função pulmonar.

[ inicio de execução do protótipo SOPRO ]

------  Paciente 7448  ------

1) Número de identificação do paciente

** 9007

2) Diagnóstico

** ASMA

[ Disparo para o protótipo ASMA e CM 900 ]

O protótipo ASMA é disparado pelo valor referido
no diagnóstico. A medida de certeza (CM) indica
numa escala numérica o grau de certeza com o qual
o protótipo é indicado por um dado.

3) RV/RV - prognosticado:

** 261

4) TLC (quadro) observado/prognosticado:

** 139

5) FVC/FVC prognosticado:

** 81

[ disparo para Normal e CM 500 ]

( Obs: O questionamento continua e outros protótipos
  são disparados em função dos valores dos dados. )
```

Figura III.16: Um exemplo da interpretação de um conjunto de resultados de testes de um paciente (BUCHANAN-84).

Um protótipo pode ser visualizado como mostra a figura (III.17).

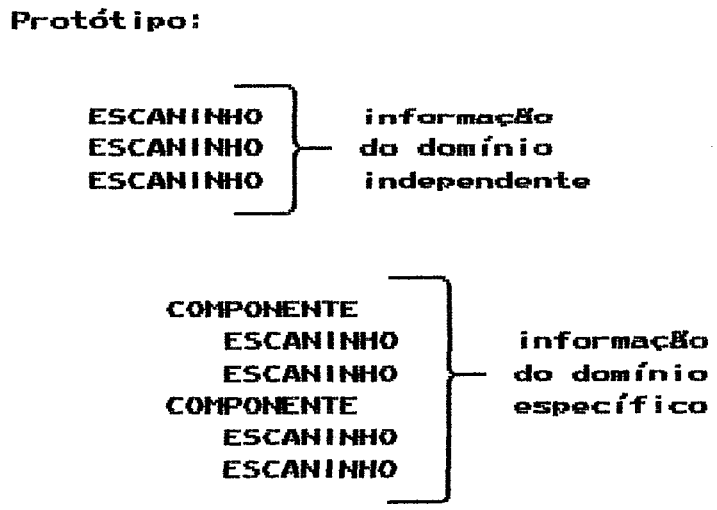


figura III.17: Representação de protótipo (BUCHANAN-84).

As regras do Centaur são agrupadas dentro de quatro conjuntos de acordo com suas funções. Elas se referem aos valores dos componentes nas suas cláusulas de premissas e, a partir daí, fazem conclusões sobre os valores dos componentes nas suas cláusulas de ações. Um exemplo de uma das regras é fornecido na figura (III.18).

No Centaur, cada parte de um dado que é adquirido em outras iniciações de testes do paciente ou posteriormente durante um processo de interpretação, é chamado de um fato. Cada fato é constituído de seis campos de informações a eles associados. Quando um primeiro fato é introduzido no sistema, seu nome, valor, e o fator de certeza são iniciados.

```
REGRA 013:

   PREMISSA:
   ($AND($OR($AND(LESSP(VAL1 CNTXT MMF)20)

   (GREATERP(VAL1 CNTXT FVC)

   ($AND(LESSP(VAL1 CNTXT MMF)15)

   (LESSP(VAL1 CNTXT FVC)80))

   AÇÃO:
   (DO-ALL(CONCLUIR CNTXT DEGMMF REGISTRO 900)

   (CONCLUIRTEXTO CNTXT ENCONTRADOOAD(TEXT $MMF)

   REGISTRO 1000))

   se:
   1) A: O MMF/MMF prognosticado e razão é menor
         que 20, e

      B: O FVC/FVC prognosticado e razão é maior
         que 80, ou

   2) A: O MMF/MMF prognosticado e razão é menor
         que 15, e

      B: O FVC/FVC prognosticado e razão é menor
         que 80

   então:
   1) Existe uma evidência de .9 que o grau de
      obstrução indicado por MMF é grave.
```

Figura III.18: Exemplo de regra no Centaur (BUCHANAN-84).

Por exemplo, se o usuário especifica que a capacidade total do pulmão do paciente é 126, com um fator de certeza de 0.8, então o fato é criado:

```
NOME             : CAPACIDADE TOTAL DO PULMÃO
VALOR            : 126
FATOR DE CERTEZA: 0.8
```

O quarto campo associado com o fato indica onde ele foi obtido: do usuário (isto inclui os resultados iniciais dos testes), das regras, ou como um valor padrão associado com um protótipo. Então, no fato sobre a capacidade total do pulmão, o quarto campo deverá ter o valor "USUÁRIO".

O quinto campo de cada fato torna-se iniciado uma vez que os valores dos fatos são classificados como valores plausíveis, valores passíveis de erros ou valores surpresa para um determinado protótipo. Os valores surpresa são todos aqueles que não são valores plausíveis nem valores passíveis de erro. Eles indicam fatos que não podem ser obtidos pelas hipóteses representadas pelo protótipo.

O último campo associado com fato indica qual tipo de protótipo confirmado contém o valor dado.

A estrutura de controle no Centaur é outro ponto interessante de ser citado. A informação de controle é contida em cada escaninho ou numa simples interpretação. Algumas estratégias de controle são específicas para um protótipo individual e necessárias para serem associadas com ele, enquanto sistemas mais gerais de informações de controle são mais eficientes expressos numa interpretação. O objetivo da estratégia de controle é de encontrar valores para os componentes do protótipo, ou seja, instanciar o protótipo.

III.4.2 - SISTEMA KRYPTON (BRACHMAN-83a).

Embora haja muito desacordo no que diz respeito ao trabalho de representação do conhecimento, parece que algumas tendências estão surgindo. A pesquisa tem-se centrado no desenvolvimento de linguagens baseadas em quadros, com as seguintes características:

- Os principais objetos de representação, ou quadros, são descrições não atômicas um tanto complexas.
- Os quadros são definidos como especializações de quadros mais gerais.
- Os indivíduos são representados por instanciações de quadros gerais.
- As conexões entre quadros resultantes formam taxonomias.

O grande apelo das taxonomias de quadros parece ser devido a sua proximidade com nossas intuições a respeito da maneira de estruturar o mundo (como ilustrado em taxonomias de povo, por exemplo). Elas também sugerem direções para o processamento (herança, omissões, etc.) e encontraram aplicações em outras áreas da ciência computacional, tais como, controle de base de dados e programação orientada a objeto.

Embora as idéias básicas dos sistemas de quadros sejam diretas, surgem complicações em sua elaboração e uso. Essas complicações normalmente surgem porque:

1- As estruturas são interpretadas de diferentes maneiras em diferentes momentos (sendo que a

principal ambiguidade situa-se entre as interpretações factuais e de definição).

2- O significado da linguagem de manipulação é especificado somente em termos das estruturas de dados usadas para implementá-la (tipicamente redes de herança).

(BRACHMAN-83a) desenvolveu uma estratégia de elaboração para evitar estes tipos de problemas e implementou um sistema de representação baseado nela. O sistema, chamado KRYPTON, distingue claramente a informação factual da de definição. O KRYPTON possui duas linguagens de representação, uma para formar termos descritivos e outra para fazer declarações a respeito do mundo utilizando estes termos. Além disso, o KRYPTON fornece uma visão funcional de uma base de conhecimento caracterizada em termos do que pode ser perguntado ou dito a ela e não em termos das estruturas particulares utilizadas para representar o conhecimento.

A estratégia definida para o KRYPTON é de centrar-se numa especificação funcional da base de conhecimento, substituindo a pergunta "Quais estruturas os sistemas deverão manter para o ususário ?" pela questão "O que, exatamente, o sistema deveria fazer para o ususário ?". Em outras palavras, decidiu-se o que era necessário para se interagir com uma base de conhecimento, sem nenhuma consideração a respeito de sua estrutura interna.

As operações no KRYPTON são divididas em dois tipos separados, produzindo dois componentes principais: um

componente terminológico, ou "T Box", e um componente assertivo, ou "A Box". O componente "T Box" permite estabelecer taxonomias de termos estruturados e responder perguntas a respeito das relações analíticas entre estes termos. O componente "A Box" permite construir teorias descritivas de domínios de interesse e responder perguntas a respeito destes domínios.

A separação entre os dois componentes surge naturalmente nos dois tipos de expressões utilizadas para representar conhecimento: termos (nominais) e sentenças. O "T Box" lida com equivalentes formais de frases nominais do tipo "uma pessoa com pelo menos três filhos", e compreende que esta expressão é incluída por (pela versão formal de) "uma pessoa com pelo menos um filho", e está desligada de "uma pessoa com, no máximo, um filho". O "A Box", por outro lado, opera com o equivalente formal de frases do tipo "Toda pessoa com pelo menos três filhos possui um carro", e compreende as implicações (no sentido lógico) de asserções deste tipo.

A linguagem "T Box" sustenta particularmente dois tipos de expressões: expressões de conceito, que correspondem superficialmente aos quadros (ou aos conceitos do KL-ONE) e expressões de regras, as contra-partes dos escaninhos (ou regras do KL-ONE).

De uma forma geral, os conceitos e regras são formados pela combinação ou restrição de outros conceitos e regras. Por exemplo, a linguagem inclui um operador "Conj Generic" (para gêneros conjugados), que têm um número

qualquer de conceitos e forma o conceito correspondente a sua conjugação. Este operador poderia ser usado para definir o símbolo "solteiro" designando para este símbolo a expressão "(Conj Generic pessoa-não-casada homem)", considerando que os símbolos pessoa-não-casada e homem tenham definições apropriadas como conceitos.

Também pode-se formar conceitos pela restrição de outros conceitos, utilizando-se regras. Por exemplo, o operador VR Generic (gênero restrito por valor) que toma dois conceitos c_1 e c_2 e uma regra r e produz o significado de termo "um c_1 de qualquer r é um c_2", como em (VR Generic pessoa filho solteiro) para "uma pessoa da qual todos os filhos são solteiros". A linguagem também possui um operador NR Generic (gênero restrito por número) que restringe a cardinalidade do conjunto de preenchedores para uma dada regra. Por exemplo, (NR Generic pessoa filho 1 3), para "uma pessoa com pelo menos um e não mais do que três filhos".

As regras e os conceitos podem ser definidos como especializações de outras regras. Um operador básico de especialização de regra "VR Diff Role" (diferenciação restrita por valor) toma uma regra r e um conceito c, e define a regra derivativa correspondente à frase "um r que é um c". Por exemplo, "filho" poderia ser definido como "VR Diff Role filho homem", dados os termos "filho" (uma regra) e "homem" (um conceito).

Para se observar a correspondência da linguagem "T Box" com uma linguagem de quadros, considere um quadro

"família":

```
família
    é-uma (IS-A) social estrutura-social
    genitor-masculino: homem (exatamente 1)
    genitor-feminino: mulher (exatamente 1)
    filho: pessoa
```

Considerado como um conceito, este quadro seria expresso no KRYPTON como:

```
(PRIM Generic (Conj Generic
    (NR Generic
    (VR Generic estrutura-social genitor-masculino
    homem) genitor-masculino 1 1)
    (NR Generic
    (VR Generic estrutura-social genitor-feminino
    mulher) genitor-feminino 1 1)
    (VR Generic estrutura-social filho pessoa)))
```

Através de diversas experiências realizadas com linguagens no "T Box", os principais operadores considerados foram os seguintes:

EXPRESSÃO	INTERPRETAÇÃO	DESCRIÇÃO
CONCEITOS:		
(Conj Generic c1...cn)	" um c1 e... um cn"	Conjunção •
(VR Generic c1 r c2)	" um c1 de qq r é um c2"	Restrição de valor
(NR Generic c n1 n2)	" um c entre n1 e n2"	Restrição de número
(Prim Generic c i)	" um c do i-ésimo tipo"	Primitivo de subconceito
(Decomp Generic c i j disjunção ?)	"um c do i-ésimo tipo do j-ésimo (disjunção) decomposição"	Decomposição
REGRAS:		
(VR Diff Role c r)	" um r que é um c"	Diferenciação de regra
(Role Chain r1...rn)	" um rn de ... de um r1"	Cadeia de regra
(Prim Role r i)	" um r da i-ésima classe"	Primitiva subregra
(Decomp Role r i j disjunção ?)	" um r do i-ésimo tipo do j-ésimo (disjunção) decomposição"	Decomposição

Da mesma forma que as expressões da linguagem "T Box", as sentenças da linguagem "A Box" são construídas por composição, a partir de outras mais simples.

A linguagem "A Box" é estruturada como uma linguagem de cálculo de predicados de primeira ordem, ou seja, são utilizados os operadores formadores de sentença usuais: Não, OU, Existe, etc.

A maior diferença entre a linguagem "A Box" e uma linguagem lógica de primeira ordem reside nas sentenças primitivas. Os símbolos não lógicos de uma linguagem lógica padrão (isto é, os símbolos de predicado e os de função, quando há) são considerados como termos independentes, primitivos e dependentes do domínio. O sistema KRYPTON possui uma facilidade para especificar uma coleção de termos dependentes de domínio, qual seja, a "T Box". A

abordagem do KRYPTON é fazer com que os símbolos não lógicos da linguagem "A Box" sejam os termos da linguagem "T Box".

O KRYPTON defende a divisão de um sistema de representação em dois componentes distintos: terminológico e assertivo. O componente terminológico sustenta a formação de descrições estruturadas organizadas taxonomicamente, enquanto que o componente assertivo permite que estas descrições sejam usadas para caracterizar algum domínio de interesse. Em ambos os casos, se têm uma linguagem de composição que é usada para interagir com uma base de conhecimento.

III.4.3 - SISTEMA PSN (MYLOPOULOS-83).

A pesquisa sobre a representação do conhecimento pode envolver um estudo da forma de representação de noções semânticas particulares como tempo, causalidade, crenças e intenções. Por outro lado, ela pode tomar a forma de um projeto de elaboração de linguagem, onde a linguagem serve para a representação de conhecimentos e os "programas" escritos na linguagem são bases de conhecimento armazenando conhecimento a respeito de algum domínio. Uma outra alternativa é que a pesquisa sobre a representação de conhecimento possa envolver o desenvolvimento de um meio de programação, como o InterLisp ou o Unix, para construir e utilizar bases de conhecimento. O projeto de Redes Semânticas de Procedimento ou PSN (Procedure Semantic Networks), iniciou-se na universidade de Toronto, em 1976,

como um projeto de implementação e elaboração de linguagem.

Desde os influentes trabalhos sobre quadros de Minsky, muitas tentativas têm sido feitas para integrar as características das linguagens de representação declarativa e de procedimento. PSN é uma destas tentativas, objetivando a integração das noções de rede semântica e de procedimento.

A principal característica das redes semânticas é seu ponto de vista associativo, o qual admite uma representação gráfica óbvia e pode ser usado para definir caminhos de acesso conceituais ou de implementação. Alguns destes caminhos podem ser, e de fato têm sido, usados para organizar redes semânticas ao longo de diferentes dimensões. Tradicionalmente, sua maior desvantagem é que seus elaboradores dependem mais das intuições advindas dos rótulos de conexão e do nó, do que da semântica formal. PSN lida com esse problema associando programas a cada objeto genérico de uma base de conhecimento da PSN, programas que especificam como operar nas instâncias daquele objeto genérico. Esta abordagem está no espírito dos tipos de dados abstratos e de algumas das idéias da proposta dos quadros. A base de conhecimento da PSN consiste de objetos, que podem ser sinais, classes, conexões, relações e programas. Sinais representam entidades particulares -tais como a pessoa John Smith, o número "7" e a série "estória"- e se interrelacionam através de conexões que representam relações binárias entre entidades. Classes representam conceitos gerais -como pessoa, número e série- enquanto que

relações representam relacionamentos genéricos como a relação "irmão-de" entre uma pessoa do sexo masculino e outra pessoa. Cada sinal é uma instância de pelo menos uma (geralmente várias) classe, enquanto que cada conexão é uma instância de pelo menos uma relação.

A figura (III.19) é uma configuração simples envolvendo dois sinais, John e 23; duas classes, Pessoa e Número; uma relação, Idade; e uma conexão relacionando John e 23. A figura também indica os relacionamentos "instância-de" entre John, 23 e a conexão "john-23", bem como aquelas entre Pessoa, Número e Idade. Quatro programas fornecem o "significado" de cada classe pela especificação de como inserir, remover e alcançar as instâncias da classe, e como testar se um objeto está em uma instância da classe. Da mesma forma, cada relação possui quatro programas associados que especificam como inserir ou remover conexões de instância, como alcançar todos os objetos na faixa da relação associados a uma instância particular do domínio, e como testar se dois objetos estão interrelacionados através de uma instância da relação.

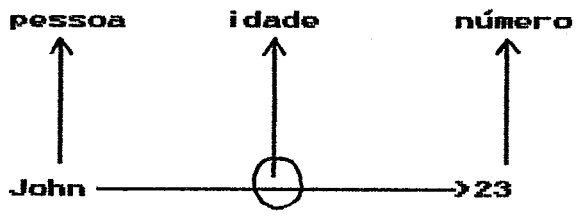


Figura III.19: Uma configuração PSN (MYLOPOULOS-83).

Uma grande base de conhecimento, assim como um grande programa, precisa ser estruturada através de princípios organizacionais intuitivos para ser compreendida por seus elaboradores e usuários. É responsabilidade da linguagem de representação de conhecimento fornecer estes princípios. A PSN oferece três destes princípios nas relações primitivas INSTANCE-OF (instância-de), PART-OF (parte-de) e IS-A (é-um).

Uma INSTANCE-OF entre sinais e classes que constituem seus tipos. Um sinal pode ser uma instância de diversas classes (por exemplo, John Smith pode ser uma instância de Pessoa e Estudante). Na PSN, a relação INSTANCE-OF é usada para relacionar todos os objetos - sinais, classes, conexões, relações ou programas- a seus tipos. Assim, podemos ter metaclasses, que têm outras classes como instâncias, como indicado na figura (III.20).

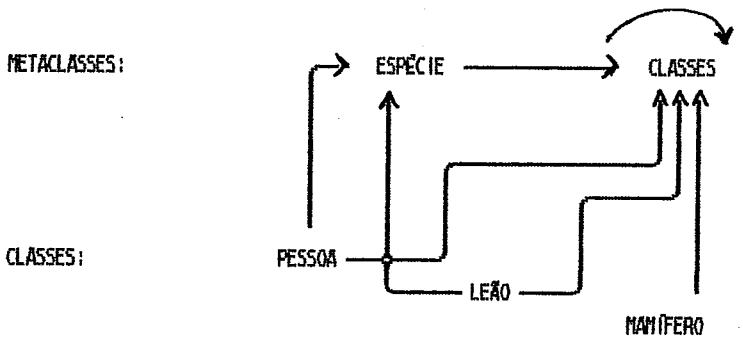


Figura III.20: Em PSN, metaclasses podem ter outras classes como instância (MYLOPOULOS-83).

Cada objeto de uma base de conhecimento deve ser uma instância de pelo menos uma classe ou relação. Diversas metaclasses embutidas tornam isto possível:

- "Objeto" tem todos os objetos, incluindo a si mesmo, como instâncias.
- "Classe", tem todas as classes, incuindo a si mesma e o objeto, como instâncias.
- "Relação" tem todas as relações como instâncias.
- "Programa" tem todos os programas como instâncias.

As metaclasses podem ser vistas como constituindo o topo da dimensão INSTANCE-OF. A profundidade da dimensão depende da aplicação e pode ir de dois sinais, classes ou metaclasses embutidas até qualquer profundidade.

A agregação é o segundo mecanismo útil na estruturação de uma base de conhecimento. Cada conceito como um agregado de conceitos mais simples, compostos de maneira particular. Por exemplo, uma pessoa pode ser conceitualizada como um agregado de suas características físicas ou, como um agregado de suas características sociais como nome, endereço e número de carteira de identidade. Este princípio organizacional é realizado na PSN através de escaninhos, que são associados a uma classe e especificam as partes do conceito representado.

Uma terceira dimensão organizacional relaciona um conceito a suas especializações ou generalizações. Desta forma, Pessoa relaciona-se a Estudante, Empregado, Macho, etc., enquanto que Estudante relaciona-se a Graduado, Pós-

graduado, Interno, etc. Mantendo uma tradição há muita estabelecida, a relação entre uma classe e suas generalizações é chamada de IS-A (assim, Estudante é-uma (IS-A) Pessoa). Como uma relação parcial entre classes, IS-A define uma hierarquia de generalização ou hierarquia IS-A. Um tema chave nas hierarquias IS-A é o tipo de herança de escaninho que elas sustentam. Assim, se Pessoa possui os escaninhos Nome e Idade, estes escaninhos são herdados por suas especializações, como Estudante. Obviamente, Estudante pode ter escaninhos adicionais, tais como "número de estudante" e "supervisor".

Cada programa é uma classe cujos escaninhos determinam seus parâmetros, pré-requisitos (condições que devem ser verdadeiras antes de cada execução, e ações que devem ser realizadas durante cada execução). Já que um programa é uma classe, ele possui suas próprias generalizações, ou programas, a partir dos quais ele pode herdar escaninhos especificando parâmetros, pré-requisitos e ações.

O projeto PSN tem a meta de desenvolver e testar instrumentos linguísticos, e outros, que facilitem a construção de grandes bases de conhecimento. Obviamente, muito ainda tem que ser feito antes de essa meta ser alcançada.

III.4.4 - SISTEMA KL-ONE (WOODS-83).

Um problema fundamental na construção de sistemas baseados no conhecimento é a análise de uma situação para

determinar o que deve ser feito. Por exemplo, muitos sistemas especialistas organizam-se em torno de um conjunto de "regras de produção", um conjunto de regras padrão-ação que caracterizam o comportamento desejado do sistema. Um sistema desse tipo determina, passo a passo, as regras que são satisfeitas pelo estado corrente do sistema e, então, age sobre este estado executando uma destas regras. Conceitualmente, esta operação obriga que cada uma das regras seja testada frente ao estado corrente. Entretanto, com o aumento do número de regras, é necessário que se busque técnicas que evitem todos estes testes, conforme figura (III.21).

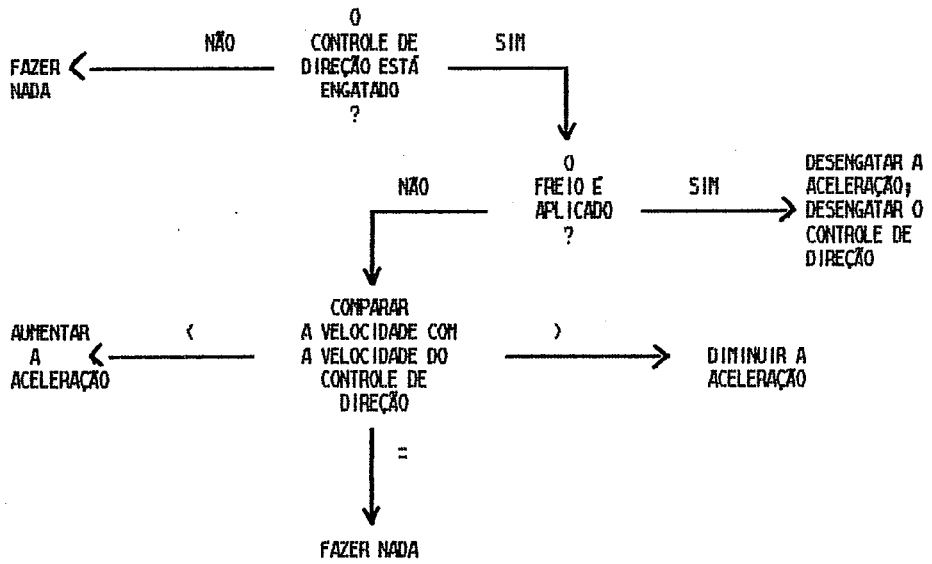


Figura III.21: Uma rede de estrutura de conhecimeto fatorada (WOODS-83).

Para determinar as regras que se aplicam, é considerado que as partes padrão de tais regras se organizem numa taxonomia estruturada de todas as situações e objetos conhecidos totalmente pelo sistema. Por "taxonomia" denomina-se uma coleção de conceitos ligados por uma relação de generalidade que permite que os conceitos mais gerais sejam acessados a partir de um dado conceito. Taxonomia "estruturada" significa que as descrições de conceitos possuem uma estrutura interna ao alcance do sistema de computação de modo que, por exemplo, a colocação dos conceitos dentro da taxonomia possa ser determinada por computação. Uma das características de tal taxonomia é que a informação pode ser armazenada em seu nível de aplicabilidade mais geral e pode ser acessada por conceitos mais específicos, ditos "herdeiros" de tal informação.

Se esta estrutura taxonômica estiver disponível, as partes de ação das regras do sistema podem ser ligadas a nós na estrutura como "conselhos" que se aplicam a situações descritas por estas regras. Deste modo, a determinação das regras aplicáveis a uma dada situação consiste em classificar a situação dentro da taxonomia e herdar o conselho. Assim, o principal papel que uma rede de conhecimento pode representar para este sistema é servir como um "cabideiro" sobre o qual possam ser herdados vários procedimentos ou métodos para o sistema executar. Uma taxonomia conceitual como esta pode organizar as partes padrão das regras de um sistema em uma estrutura eficiente que facilite o reconhecimento. O KL-ONE é um sistema deste

tipo.

Na construção de descrições internas de situações é necessário que se utilize conceitos de objetos, substâncias, tempo, lugares, eventos, condições, predicados, funções, indivíduos, etc. Cada conceito pode ser caracterizado como uma configuração de atributos ou partes, satisfazendo certas restrições e permanecendo em relações especificadas entre si. Esta caracterização é o elemento básico do sistema de representação de conhecimento KL-ONE.

Um nó de conceito no KL-ONE consiste de um conjunto de funções (generalizações das noções de atributo, parte, constituinte, característica, etc) e de um conjunto de condições estruturais expressando as relações entre eles. Os conceitos são ligados a conceitos mais gerais por uma relação chamada SUPERC. O conceito mais geral numa relação como esta é chamado de "superconceito" e inclui o subconceito, mais específico. Algumas das funções e condições estruturais de um conceito são ligados a ele diretamente, outros são herdados indiretamente de outros conceitos mais gerais.

O objetivo do KL-ONE não é produzir um sistema de computação em particular, e sim forçar a descoberta e articulação de princípios gerais de estrutura e organização de conhecimento. A adequação expressiva é uma importante força impulsora na pesquisa do KL-ONE, enfatizando a semântica da representação e sua adequação para fazer as distinções sutís que podem ser feitas pelas pessoas na

conceituação de idéias complexas.

A figura (III.22) ilustra uma estrutura taxonômica do KL-ONE, onde os conceitos estão representados por retângulos.

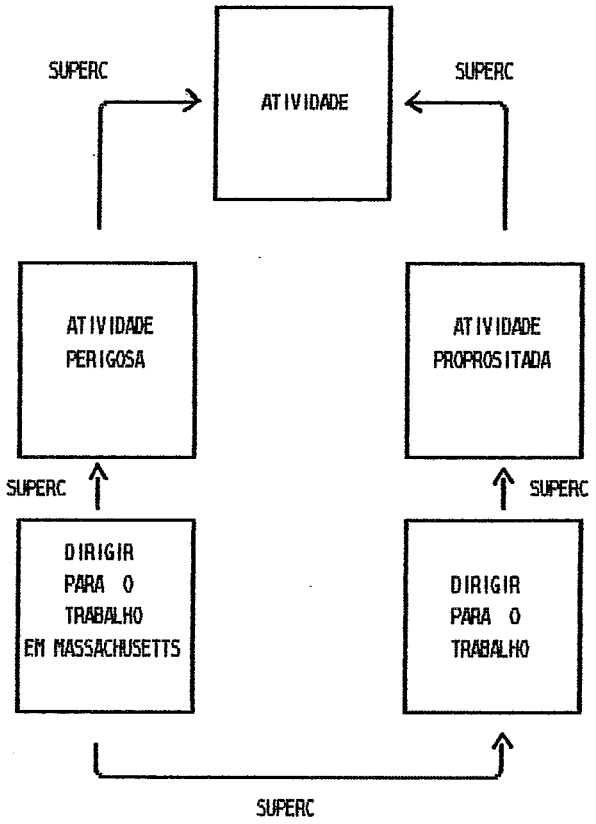


Figura III.22: Exemplo de uma rede KL-ONE (WOODS-83).

No topo da figura há um conceito de alto nível de Atividade, com papéis para Tempo, Lugar e Participantes, os quais são herdados por todos os conceitos abaixo. Abaixo de Atividade, à direita, está o conceito para Atividade Propositada, que diferencia (DIFFS) o papel geral de Participantes em um Agente (o participante que tem um propósito) e Outros Participantes. Atividade Propositada introduz um novo papel chamado Meta, para representar o

propósito da Atividade.

Abaixo de Atividade Propositada está o conceito mais específico, porém ainda geral de Dirigir para o Trabalho. Este conceito modifica a Meta de Atividade Propositada adicionando Ir ao Trabalho como uma restrição de valor (V/R), indicando que o que quer que preencha a Meta deve ser uma instância de Ir ao Trabalho. Este conceito também introduz um novo papel chamado Destino, com Local de Trabalho como sua restrição de valor. Uma condição estrutural (não mostrada) ligada ao conceito, especificaria como o Local de Trabalho relacionar-se-ia com a Meta Ir ao Trabalho (isto é, que Local de Trabalho é o Destino da meta Ir ao Trabalho). Dirigir para o Trabalho em Massachusetts é, por sua vez, uma especialização de Dirigir para o Trabalho, com seu Destino restrito (MODS) a um Local em Massachusetts. Dirigir para o Trabalho em Massachussets é também uma especialização de uma Atividade Perigosa, com um Risco de Dano Físico.

Este é o tipo de taxonomia do qual se espera uma agente de computação inteligente, incluindo abstrações de nível muito alto e conceitos bastante específicos. Além disso, há sempre espaço para a inserção de novos níveis de abstração entre os existentes. De fato, um procedimento de classificação bem definido implementado no sistema KL-ONE pode alocar uma nova descrição dentro de uma taxonomia automaticamente, ligando-a por conexões SUPERC aos conceitos que a incluem mais especificamente e àqueles que ela, por sua vez, inclui.

III.5 - CONCLUSÕES.

Apresentamos neste capítulo as características básicas e principais dos métodos de representação de conhecimento.

Não obstante aos demais métodos, a lógica de primeira ordem tem uma força expressiva bastante geral e uma semântica bem definida. Entretanto, devido sua linguagem de construção ser com variáveis atômicas e de não permitir facilidades adequadas para definição de construções mais complexas, a representação de certos domínios não é adequada quando se utiliza esse método, como ainda, há dificuldade de se entender o conhecimento expresso por ele. Também, as generalidades da lógica de primeira ordem tem tido uma barreira significante para o desenvolvimento de facilidades de dedução para utilizar o conhecimento representado.

Essas dificuldades motivaram o desenvolvimento de representações baseadas em Redes Semânticas e Quadros.

Uma representação em quadros permite um construtor da base de conhecimento desenvolver facilidades significantes descrevendo tipos de objetos do domínio que um sistema deve modelar.

Tendo em vista a dificuldade em se declarar como o conhecimento armazenado em quadros é para ser utilizado, tem-se utilizado as regras de produção para permitir a descrição de regras de inferências, regras para análise de decisão, ações que podem ser tomadas por vários agentes do

domínio, procedimentos para simulações de objeto, etc.

Uma tendência observada é o desenvolvimento de sistemas que utilizam a integração dos métodos de representação para formar facilidades híbridas de representação que combinem as vantagens dos métodos de representação de conhecimento, tais como os sistemas de representação apresentados na seção (III.4).

Para os modelos conexionistas, a década de 80 tem representado um marco na evolução e interesse pelo método, como causado enorme impacto nas "ciências da inteligência" (isto é, ciência da cognição, neurociências, etc).

Muita pesquisa precisa ser realizada para que a jovem filosofia conexionista amadureça. McClelland e Rumelhart (referenciados por (PESSOA-89)), apontam as seguintes áreas que deverão merecer futuras investigações:

- aplicação de modelos conexionistas a processos cognitivos de mais alto nível e processamento simbólico sequencial;
- prosseguir com o desenvolvimento de mecanismos de aprendizado e uma análise concomitante de arquiteturas de redes, e,
- uma ligação mais estreita entre modelos conexionistas e neurociência.

Com relação aos métodos de representação, é importante considerar uma dimensão ao longo do qual eles podem ser caracterizados, ou seja, métodos sintáticos e semânticos.

Os métodos puramente sintáticos são aqueles que não tem nenhuma preocupação com o significado do conhecimento que está sendo representado. Tais sistemas possuem regras simples e uniformes para manipular o conhecimento representado, não se importando com a informação contida pela representação. Se enquadram neste caso os seguintes métodos: lógica de primeira ordem e regras de produção.

Os métodos puramente semânticos são aqueles que não tem nenhuma forma específica. Cada aspecto da representação corresponde a uma peça de informação diferente e as regras de inferências são mais complexas. Se enquadram neste caso os seguintes métodos: redes semânticas, quadros, dependência conceitual e os roteiros.

Um outro fator a ser considerado é a adequação do domínio da aplicação a ser tratado com relação às facilidades disponíveis no método de representação.

Embora muitas sejam, ainda, as questões a serem tratadas em busca de soluções, tais como a representação de conhecimento de senso comum (ZADEH_83), representação em sistemas de aprendizagem (LANGLEY_83) e representação de linguagem natural (WEBBER-83), é importante encontrar novos caminhos em que os métodos de representação possam contribuir para a representação de domínios que permitam um conhecimento melhor de suas características.

Capítulo IV

ESTRUTURA DE FATORES PARA ANÁLISE DE SISTEMAS DE REPRESENTAÇÃO DO CONHECIMENTO

IV.1 - INTRODUÇÃO.

Na ciência da computação, uma boa solução depende, com frequência, de uma boa representação (WOODS-83). Para a maioria das aplicações da IA, a escolha da representação é ainda mais difícil, já que as possibilidades são substancialmente maiores e os critérios menos claros.

No dia-a-dia de nossas atividades, estamos sempre lidando com conhecimentos relativos a algum domínio específico , buscando através de sua manipulação, encontrar soluções e obter resultados de forma a alcançar as metas designadas. Esses conhecimentos se restringem a dois tipos: **fatos e regras**. Aos **fatos** relacionamos parte do conhecimento que se têm sobre uma parte do domínio, que significa uma **verdade** sobre ele. Quando se associa um fato com alguma relação condicional, ou a outro fato, obtém-se uma conclusão, que vêm significar uma **regra** em tal domínio. Por exemplo:

fato 1: O forno está aceso

regra 1: SE eu puser minha mão em um forno aceso, ENTÃO eu vou me queimar.

Os diversos modelos de representação estão identificados segundo características básicas que permitem conhecer o tipo de modelo e a forma de tratamento com o conhecimento e, então, se fazer um uso mais adequado. (ROLSTON-88) considera que a representação do conhecimento pode ser classificada em dois tipos: **declarativa** e **procedural**.

Os métodos **declarativos** representam a maior parte do conhecimento como uma coleção estática de fatos acompanhados por um pequeno conjunto de procedimentos gerais para manipulá-los. Por exemplo: lógica de predicados, redes semânticas, quadros e roteiros.

As vantagens consideradas por (RICH-88) para a representação declarativa, são:

- Cada fato só precisa ser armazenado uma única vez, independente do número de maneiras diferentes em que puder ser utilizado.
- Facilidade em acrescentar novos fatos aos sistemas, sem mudar os outros fatos e nem os pequenos procedimentos.

Os métodos **procedurais** representam a maior parte do conhecimento na forma de regras dinâmicas para descrever os procedimentos. Por exemplo: sistemas de produção de regras.

As vantagens consideradas por (RICH-88) para a representação procedural, são:

. Facilidade em representar o conhecimento de como fazer as coisas.

. Facilidade em representar o conhecimento que não se enquadra bem dentro de muitos modelos declarativos simples. Por exemplo: raciocínio por omissão e o probabilístico.

. Facilidade em representar o conhecimento heurístico de como fazer eficientemente as coisas.

Uma extensão para o uso da representação de conhecimento se relaciona como um auxílio para a fase de levantamento de requisitos num produto de software. Os objetos neste tipo de modelo (BORGIDA-85), são expressos para que em cada domínio represente alguma entidade ou atividade, tal como uma pessoa, prescrição de medicamento ou numa análise clínica.

Quando da construção de um modelo de requisitos, uma das preocupações com o qual o objeto do domínio será representado, se refere a quais de suas propriedades são relevantes e como se comportará no modelo que representa o domínio.

A importância de um modelo expressar claramente as características do domínio é que facilita a comunicação entre seus usuários, podendo ser um modelo subjetivo até um auxílio automatizado.

Quando um modelo permite que o conhecimento sobre o domínio seja bem construído e representado, resulta num melhor entendimento dos projetistas da aplicação sobre o

domínio, atendendo assim, às necessidades reais dos usuários.

Como apresentado no capítulo anterior, vários métodos de representação de conhecimento foram desenvolvidos para facilitar a visualização e manipulação dos fatos e regras de um domínio considerado.

Embora as características próprias de cada método estabeleçam uma tendência de utilização, outros critérios vêm sendo introduzidos a fim de permitir uma melhor adequação de métodos e domínios de aplicação.

(ROLSTON-88) considera que os métodos de representação declarativos devem ter as seguintes características:

- **Transparência**: a representação do conhecimento deve ser numa forma explícita e não-ambígua.
- **Armazenamento eficiente**: cada parte do conhecimento deve ser armazenado numa única vez e utilizado em diversos caminhos.
- **Flexibilidade**: o conhecimento pode ser armazenado no nível mais baixo, tendo seu crescimento de forma flexível.
- **Inferência direta**: permite inferências diretas e explícitas.

(RICH-88) considera que um bom modelo de representação do conhecimento estruturado complexo num

domínio particular deveria possuir as quatros propriedades seguintes:

- **Adequação Representacional**: capacidade de representar todos os tipos de conhecimento que forem necessários naquele domínio.

- **Adequação Inferencial**: capacidade de manipular as estruturas representacionais de modo a derivar novas estruturas correspondentes ao conhecimento novo inferido do antigo.

- **Eficiência Inferencial**: capacidade de incorporar, dentro da estrutura do conhecimento, informação adicional que possa ser utilizada para direcionar os mecanismos de inferência nas direções mais promissoras.

- **Eficiência Aquisicional**: capacidade de adquirir novas informações com facilidade. O caso mais simples envolve a inserção direta de novos conhecimentos, por uma pessoa, na base de dados. De modo ideal, o próprio programa, quando este for o caso, seria capaz de controlar a aquisição do conhecimento.

(FIKES-85) considera que para o conhecimento específico de um domínio, um sistema de representação de conhecimento deve satisfazer, da forma mais adequada possível, as questões abaixo:

. **Força expressiva**: pode um especialista transmitir seus conhecimentos efetivamente para o sistema ?

. **Entendimento**: pode um especialista entender o que o sistema sabe ?

. **Acessibilidade**: pode o sistema usar a informação que tem sido fornecida ?

Considerando importante a existência de um grupo de propriedades que permitam identificar as características desejadas num sistema de representação, estendemos as idéias anteriormente citadas, e organizamos uma estrutura de fatores e sub-fatores apresentada na figura (IV.1).

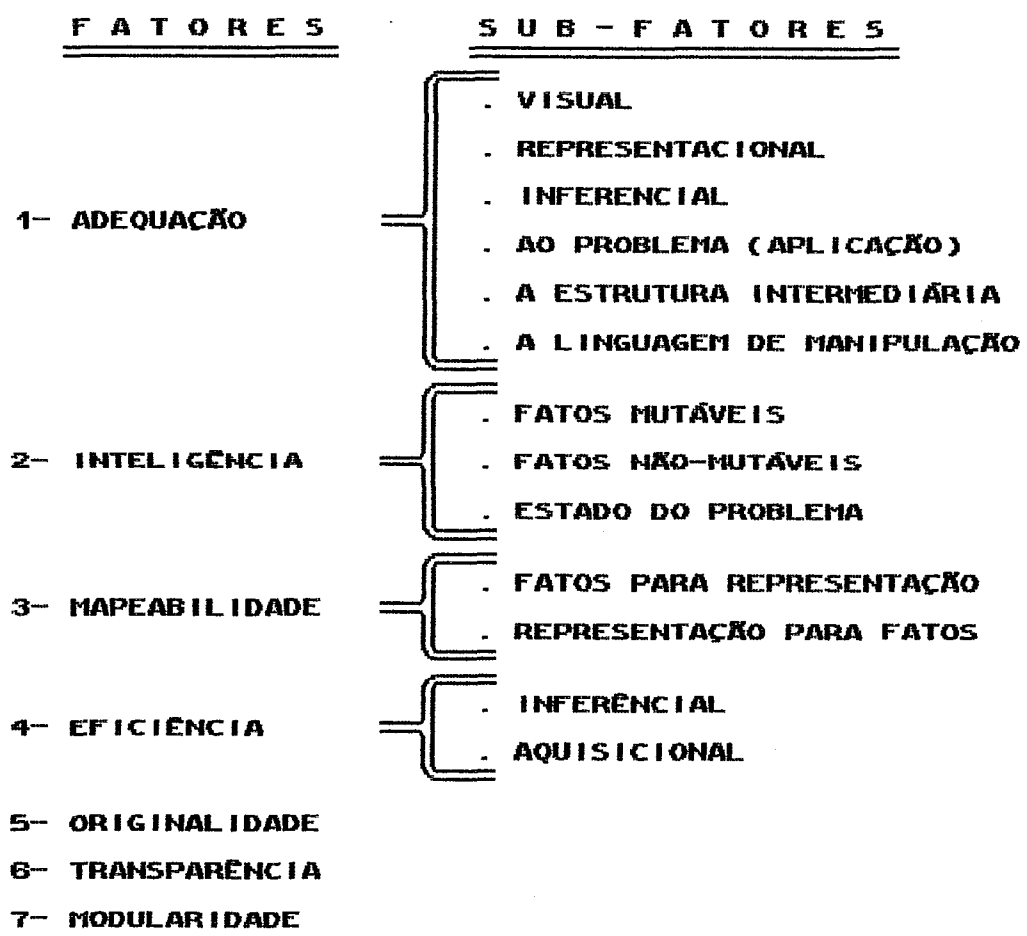


Figura IV.1: Estrutura de fatores.

IV.2 - ADEQUAÇÃO

Este fator tem o objetivo de apresentar o grau de ajustamento ou adaptação do método de representação em relação ao manuseio do conhecimento a ser representado.

Os sub-fatores tem o objetivo de especificar cada uma das formas básicas de adequação que um problema tratado e o modelo de representação devam ter atendidas entre si, ou pelo menos, ter a consciência do grau de expressividade do modelo.

. ADEQUAÇÃO VISUAL

A adequação visual expressa as carcterísticas do modelo de representação para a visualização do conhecimento representado. Algumas dessas características podem ser:

- visualização geral e de contexto;
- identificação dos tipos de conhecimentos;
- seleção de informações padronizadas;
- visualização nos meios de saidas (por exemplo: video e impressora) dos diversos tipos de descricões; e,
- visualização adequada ao meio de saída.

. ADEQUAÇÃO REPRESENTACIONAL

Este sub-fator está diretamente ligado ao modelo utilizado, buscando a adequação do conhecimento a ser tratado com o método utilizado. Neste caso se estaria a um nível abaixo da adequação visual e sendo por ela interfaceada, como ilustrado na figura (IV.2).

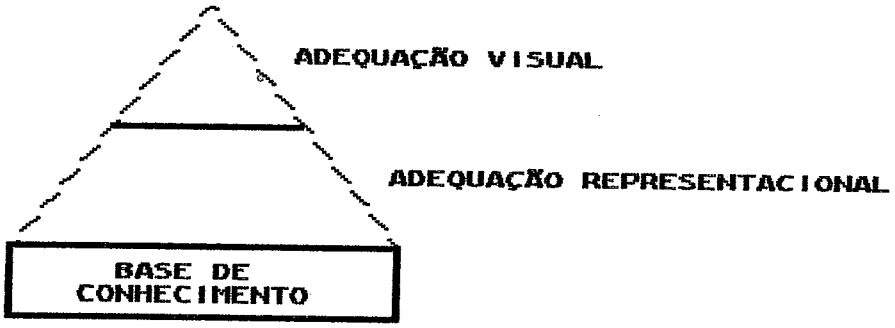


Figura IV.2: Interfaces da Adequação Representacional.

A adequação representacional significa a capacidade do modelo em representar todos os tipos de conhecimentos que forem necessários naquele domínio. O atendimento à representação desses conhecimentos, ou um grau de satisfação, contribui para o especialista e o usuário do sistema, definirem o modelo a ser utilizado (RICH-88).

. **ADEQUAÇÃO INFERENCIAL**

Cada modelo apresentado tem sua característica em relação à faciliadade e adequação em derivar novos conhecimentos inferidos do antigo.

É preciso que, neste caso, o grau de adequação representacional seja bem satisfeito, pois cada conhecimento inferido, pode se tornar um novo conhecimento a ser armazenado e atualizado quando necessário. Outra característica necessária para a adequação inferencial é a apropriação da estrutura de dados em aceitar e manipular o novo conhecimento gerado.

. ADEQUAÇÃO AO PROBLEMA (ÁREA DE APLICAÇÃO)

É fundamental se conhecer as características funcionais do problema tratado, seus requisitos e objetivos, para que a busca por uma solução seja realizada através desses conhecimentos.

É preciso que o problema seja analisado e avaliado, antes de tudo, segundo os conceitos aqui enunciados, pois assim, saber-se-á melhor escolher o método de representação e mais fácil se encontrarão as soluções para o problema.

Portanto, a adequação do problema ao método de representação de conhecimento, ou vice-versa, significa compatibilizar **problema-método-solução**, considerando-se como uma atividade preliminar e essencial para o êxito do trabalho a ser desenvolvido.

. ADEQUAÇÃO À ESTRUTURA INTERMEDIÁRIA

Com o objetivo de descrever fatos, condições e assertivas, é necessário que se defina uma estrutura que sirva de interface com uma base de conhecimento.

A denominação "estrutura intermediária" foi escolhida para diferenciar com a estrutura de dados da base de conhecimento, onde através dela o conhecimento do domínio representado será transferido para o armazenamento final.

Uma estrutura intermediária é composta de campos que objetivam descrever e especificar informações e

condições a serem tratadas pelo método para acesso à base de conhecimento.

Uma estrutura intermediária deve ter um nome para que se possa fazer referência a ela, e atributos, que permitam descrever as informações necessárias do conhecimento representado.

Uma estrutura intermediária deve ter seus campos compatíveis com as características de cada método, ou seja, por exemplo, para o método de representação por Quadros, haver campos estáticos para a descrição do conhecimento fixo e campos dinâmicos, para a descrição do conhecimento que surge a cada visão do problema.

Portanto, a adequação de uma estrutura intermediária significa organizar as informações a serem tratadas de forma compatível com o problema e com o método de representação.

. ADEQUAÇÃO À LINGUAGEM DE MANIPULAÇÃO

Para que as informações contidas numa base de conhecimento sejam acessadas, tanto para o armazenamento quanto para leitura e consulta, é preciso que haja uma linguagem colocando disponível essas funções.

Uma linguagem de manipulação de conhecimento tem diversas características próprias que devem ser consideradas para um uso adequado. No entanto, no caso da representação é preciso que o modelo utilizado seja compatível com a linguagem existente, permitindo que as

funções básicas do modelo sejam, de alguma forma, possíveis de serem especificadas na linguagem.

Assim, a adequação à linguagem de manipulação significa tornar possível que as funções básicas do modelo sejam especificadas de forma a tornar seus conceitos utilizados e definidos numa base de conhecimento.

IV.3 - INTELIGÊNCIA

Os fatos são informações essenciais em um sistema, pois significam verdades sobre o domínio, permitindo alcançar os objetivos desejados. A mente humana possui uma quantidade enorme de fatos, ou seja, conhecimentos relacionados a uma incontável lista de objetos e idéias. Nossa sobrevivência depende de nossa habilidade em aplicar esses conhecimentos em qualquer situação que apareça e aprender continuamente com as novas experiências, para que sejamos capazes de responder a situações similares no futuro.

Aquilo que, geralmente, é considerado "inteligência" pode ser dividido em uma coleção de fatos e um meio de se utilizar esses fatos para alcançar os objetivos (LEVINE-88). O tratamento com os fatos é feito através da formulação de conjuntos de regras relacionadas a todos os fatos armazenados sobre o domínio.

Este fator tem a finalidade de expressar a qualidade ou capacidade, do método de representação, de compreender e adaptar-se em relação aos conhecimentos por

ele tratado. Os seus sub-fatores revelam os tipos básicos de informações caracterizadas para o tratamento pelo método.

. **FATOS-MUTÁVEIS**

Sem entrar no mérito da questão em relação a probabilidades e critérios de certeza, o exemplo abaixo caracteriza o que seja **fatos mutáveis**, pois os fatos 1a e 1b podem ao longo do tempo ou dependente do observador deixarem de ser verdadeiros.

Por exemplo:

FATO 1a: Ruas escuras e pouco movimentadas são perigosas.

FATO 1b: Pessoas de idade geralmente não cometem crimes violentos.

REGRA 1c: SE eu estou em uma rua escura e pouco movimentada e vejo uma pessoa idosa, **ENTÃO** não devo ficar particularmente preocupado com minha segurança.

Portanto, um fato mutável é todo aquele que pode deixar de ser verdade em algum tempo dentro de um mesmo domínio. Quando existindo, deve-se ainda definir a condição de ocorrência da sua modificação, quer seja à-priori ou no momento do acontecimento.

. **FATOS NÃO-MUTÁVEIS**

No exemplo abaixo, o fato 2a será sempre verdade

no domínio.

Por exemplo:

Fato 2a: Quando fazemos a adição de dois dígitos cuja soma é maior que nove, usamos o procedimento do transporte (ou vai-um).

Fato 2b: O sistema numérico é o decimal.

Regra 2c: SE eu tiver de somar uma coluna de dígitos e a soma for maior que nove, ENTÃO eu tenho de fazer referência ao fato 2a para saber como concluir a adição.

Portanto, um fato **não-mutável** é todo aquele que se mantém inalterado ao longo do tempo num mesmo domínio.

. ESTADO DO PROBLEMA

No mundo real muitas vezes é necessário considerar a ocorrência de diversos fatos para que um resultado seja obtido. No entanto, é importante que se possa reconhecer um conjunto de fatos, com afinidades comuns ou servindo a um mesmo propósito, dando um significado para o problema.

Por exemplo:

fato 1 : Para se acordar cedo, ligue o despertador

fato 2 : Para ligar o despertador, primeiro acerte a hora.

fato 3 : Dormindo cedo se acorda mais cedo e com mais

disposição.

fato 4: Jantar uma refeição leve e adequada é bom quando se quer dormir cedo.

fato 5: É preciso acordar cedo para não apanhar o onibus cheio.

fato 6: Quem acorda cedo não chega atrasado no trabalho.

Para que o **fato 6** seja verificado, é preciso que "**acorde cedo**" tenha ocorrido, fazendo com que os **fatos 1 a 5** tenham acontecido. Portanto, numa execução é muito mais fácil se saber se "**acordar cedo**" aconteceu, do que verificar cada fato de 1 a 5, pelo menos a partir da primeira vez. Então, "**ACORDAR CEDO**" passa a representar um estado do problema que tem seu valor implícito de ocorrência ou não, a cada vez que referenciado, como se fosse um novo fato do problema tratado.

Portanto, um "**estado do problema**" vem significar a ocorrência ou não de um conjunto de fatos resultando numa condição para o novo fato resultante.

IV.4 - MAPEABILIDADE

Este fator tem a finalidade de caracterizar a capacidade do modelo em se apresentar para o meio externo e se resguardar, mantendo armazenado e íntegro seus conhecimentos, através de um meio interno, de forma própria e automática.

Os sub-fatores relacionados ao mapeamento, são as transformações necessárias de existirem, quais sejam, do meio externo para o interno e vice-versa.

. FATOS PARA REPRESENTAÇÃO

Mapear fatos para a representação significa a capacidade do modelo permitir que um conhecimento do problema seja representado visualmente e armazenado em uma base de conhecimento através de uma estrutura que permita conter informações sobre tal conhecimento e permitir, ainda, que esse conhecimento seja utilizado em alguma linguagem de manipulação.

Embora esta característica esteja inserida no contexto da representação, o mapeamento dos fatos tem importância por transformar o conhecimento de um especialista, numa forma tratável por computador e adequadamente visualizada.

Em função do método utilizado, esta tarefa deve ser adequada aos tipos de conhecimento existentes no problema: fatos mutáveis, fatos não-mutáveis e estado do problema.

. REPRESENTAÇÃO PARA FATOS

A primeira vista, este caso poderia parecer uma forma inversa do sub-fator anterior. Porém, isto não acontece devido à função do método utilizado, de transportar o conhecimento, não existir explicitamente, ficando isso a cargo do meio visual do modelo, permitindo

ao usuário tirar suas conclusões e fazer suas observações.

Uma vez um conhecimento representado, seu caminho inverso significa, a partir do armazenamento, tornar visível tais conhecimentos ao observador final, complementado ou não com novos conhecimentos.

Da mesma forma que o sub-fator anterior, devem ser mantidas as características dos fatos mutáveis, fatos não-mutáveis e estado do problema.

IV.5 - EFICIÊNCIA

Este fator tem o objetivo de determinar a capacidade do método de representação em produzir um acesso eficaz à base de conhecimento.

. INFERENCIAL

Diretamente ligada à estrutra de armazenamento, a eficiência inferencial é avaliada pela capacidade da estrutura incorporar conhecimentos adicionais que permitam ao mecanismo de inferência obter resultados de forma promissora.

. AQUISICIONAL

Segundo (RICH-88), e como já apresentado, é a capacidade de adquirir novas informações com facilidade através de inserção direta de novos conhecimentos na base de conhecimento.

IV.6 - ORIGINALIDADE

Tendo em vista as características próprias de cada método de representação, no que se refere ao tipo de conhecimento capaz de representar, nem sempre é possível a representação de um fato tal como se apresenta no mundo real.

Este fator determina a capacidade do método de representar todo o conhecimento necessário de forma original, tal como é tratado no mundo real, sem necessitar artifícios e recursos para adaptá-los ao método de representação a ser utilizado. Quanto mais capaz for o método de representar os fatos do domínio considerado, mais eficiente será para este fator.

IV.7 - TRANSPARÊNCIA

Como apresentado no capítulo anterior, os diversos métodos de representação estão voltados a algumas características próprias, fazendo com que um problema específico seja tratado por algum deles, algumas vezes até necessitando de adaptações para adequação ao modelo.

Este fator tem o objetivo de estabelecer como um método é capaz de representar o conhecimento e permitir um armazenamento explícito e transparente, de forma não-ambígua. Quanto maior esse atendimento mais eficiente é o modelo em relação ao fator.

IV.8 - MODULARIDADE

Um dos problemas enfrentados pela IA é a capacidade de se lidar com uma quantidade grande de conhecimentos. Embora os problemas do mundo real tendam para uma quantidade grande de conhecimento, é preciso que os métodos de representação tenham recursos para lidar com esta questão.

Portanto, para os modelos de representação, é importante que tenham recursos para administrar os conhecimentos de um mundo específico, permitindo que uma taxonomia do problema possa ser identificada e tratada pelo modelo de forma independente um dos outros, com partes identificadas e especificadas.

A capacidade do modelo em permitir essa partição significa a eficiência modular do modelo e, os módulos identificados e especificados serão um estado do problema, como já definido.

IV.9 - CONCLUSÕES.

Este capítulo se caracteriza pela organização que apresentamos referente às questões relacionadas com a representação do conhecimento.

Não obstante aos sistemas de software convencionais que lidam com o conhecimento de forma indireta, os métodos de representação estabelecem conceitos e recursos que permitem utilizar, diretamente, o conhecimento de forma a utilizá-lo, produtivamente, no

desenvolvimento de software.

Um aspecto fundamental de ser ressaltado é o que se refere a questão da "força expressiva" citada em (WOODS-83) e (FIKES-85).

Consideramos básico para um método ou sistema de representação do conhecimento a sua qualidade no que se refere à expressão do conhecimento, ou seja, como contribuir para o entendimento e como conhecer o que se está representando. Portanto, buscamos tratar esta questão com o fator de "adequação" onde o conjunto de seus sub-fatores buscam identificar, de forma específica, o que consideramos importante para a expressividade de um método ou sistema de representação do conhecimento.

Com relação a questão da adequação desses fatores e sub-fatores com os métodos de representação descritos no capítulo (III), destacamos os seguintes pontos.

A lógica de primeira ordem é um método com mescanismo não ambíguo e grande capacidade dedutível. No entanto, não permite inferências eficientes como ainda representar fatos do mundo real. Isto nos leva a considerá-la como um método não adequado, pois estamos lidando com um domínio específico de características reais, onde o procedimento do método de desenvolvimento, a manipulação e percepção de conhecimentos, estabelecimentos de formas de decisão e descrições, são características necessárias.

Para as redes semânticas, destacamos uma

estrutura de finalidade geral, podendo representar conhecimentos específicos e particulares de um domínio. Uma consequência que pode ocorrer com este método de representação, para certos domínios, é tornar-se uma estrutura grande, dificultando o entendimento. Para este método, o fator de "adequação" que busca a expressividade do método através dos seus sub-fatores, destacamos ser inviável para se ajustar à uma estrutura intermediária para armazenamento de conhecimentos. No caso do fator "inteligência", é importante notar que para fatos mutáveis, a atualização da rede pode resultar em grandes modificações. Uma outra dificuldade para este método, se refere aos fatores "originalidade" e "modularidade", onde a representação em redes não oferece flexibilidade para expressar o conhecimento tal como é tratado no mundo real, como ainda, organizar em partes os conhecimentos de um domínio.

O método da dependência conceitual é um método que facilita a geração de inferência e é independente da linguagem em que foi originalmente declarada. Por outro lado, requer que todos os conhecimentos sejam decompostos em primitivos de níveis relativamente baixos, o que torna difícil o entendimento e até impossível em alguns casos. É uma teoria da representação de eventos, o que não é o caso específico do domínio de desenvolvimento de software, que requer outras particularidades para expressá-los. Portanto, consideramos um método não adequado para uso nesse domínio.

O método dos quadros é uma estrutura

organizacional que permite representar conhecimentos em um dado momento, a partir de experiências já existentes ou para a manutenção dos conhecimentos, facilitando se fazer atualizações. Permite se estabelecer atributos de descrição e de valores (fixos ou por omissão) para os conhecimentos representados. Facilita a inferência de fatos e é de fácil idealização e entendimento. Não obstante ao não provimento para descrever como o conhecimento armazenado é para ser utilizado, não destacamos um fator que, dependendo dos tipos de conhecimentos que caracterizam o domínio, seja incompatível para ser expresso por ele.

A característica dos métodos dos roteiros é prever eventos não observados, ou seja, permitir a conclusão sobre um conhecimento não explícito. Este método fornece um meio de construir uma única interpretação coerente de uma coleção de observações. Esta característica que é forte neste método não é a característica geral do domínio dos métodos de desenvolvimento de software. Portanto, consideramos um método não adequado para o que esperamos expressar com os fatores e sub-fatores definidos.

O método das regras de produção (ou sistema de produção) é um método que facilita a geração de inferência e é de fácil idealização e entendimento. É inadequado para se descrever conhecimentos e as relações entre eles. Com relação ao fator "adequação", consideramos este método inviável para o entendimento visual do conhecimento representado, como ainda, para identificar inferências a partir de suas regras. Para se expressar em regras de

produção o domínio dos métodos de desenvolvimento de software será inviável no caso de descrições e de procedimentos, que são características marcantes desse domínio. Consequentemente, invibializará a adoção de uma estrutura intermediária para armazenamento. O fator referente a "modularidade" também estará invibializado uma vez que não será adequado se organizar uma forma de entendimento e relacionamento de partes dos conhecimentos do domínio tratado.

Diferentemente dos demais métodos de representação, os modelos conexionistas foram inseridos com o objetivo de apresentá-lo de forma suscinta, para que pudessemos destacar sua importância e de como lida com a questão da representação do conhecimento. Portanto, nos reservamos em não expressar, como os demais métodos, suas características em relação aos fatores e sub-fatores descritos neste capítulo.

O conjunto de fatores apresentados neste capítulo contribui não só para a análise de sistemas de representação, como ainda para permitir uma análise sobre as características desejadas quando da elaboração de um sistema de representação do conhecimento.

Consideramos este capítulo importante pela iniciativa deixada para análises futuras e para servir de apoio na elaboração do sistema de representação objeto deste trabalho.

Capítulo V

UM SISTEMA PARA REPRESENTAÇÃO DO CONHECIMENTO ADEQUADO A MÉTODOS DE DESENVOLVIMENTO DE SOFTWARE

Ao longo do desenvolvimento deste trabalho buscamos identificar e apresentar as questões mais essenciais que cercam e recaem sobre as técnicas de representar conhecimentos.

Autores como (RICH-88), (KELLER-87), (MYLOPOULOS-83) e outros, são unânimes em reconhecer a representação de conhecimento como um problema central na pesquisa de inteligência artificial, isto reforçado pela necessidade da existência de uma base de conhecimento que armazene o conhecimento e forneça facilidades para o manuseio do conhecimento armazenado.

Identificamos, ainda, durante nossa pesquisa, que o domínio tratado e o objetivo estabelecido, além de serem requisitos fundamentais para a representação do conhecimento, guiam para uma melhor solução.

Muito embora essas soluções estejam baseadas nos métodos tradicionais de representação, foi através da utilização híbrida desses métodos que pesquisadores buscaram encontrar a melhor forma de representação e manipulação do conhecimento.

Seguindo esta linha, apresentamos neste capítulo a descrição de um sistema de representação do conhecimento para métodos de desenvolvimento de software. Para tanto, a partir de uma morfologia do processo de projeto apresentado por (BACK-83), analisamos as necessidades oriundas de nossa pesquisa e identificamos os resultados desejados conforme nossos objetivos.

V.1 - ANÁLISE DE NECESSIDADES.

Conforme as características dos métodos de representação apresentadas no capítulo (III), consideramos que representar o conhecimento significa encontrar uma estrutura capaz de transferir o conhecimento de um domínio para uma forma armazenada em computador, através de um método de representação eficaz na sua forma de se apresentar ao meio externo, fazendo com que o profissional (engenheiro de conhecimento) e o usuário (para quem o produto de software é desenvolvido) ligados à solução de problemas, possam identificar "**quem**" é o domínio tratado e quais são suas características essencicais, a fim de melhor auxiliar para a determinação do "**qual (o quê)**" é o problema relacionado ao domínio e, por fim, um auxílio para "**como**" solucioná-lo.

Desta forma podemos identificar como essência para o sistema, a necessidade de um tratamento externo adequado, onde a visualização do domínio e uma interface amigável com o usuário, permitam a identificação e o entendimento das características desse domínio.

Com relação ao meio interno, ou seja, relativo à forma de armazenamento e de manipulação do conhecimento em um meio computacional, identificamos a necessidade de se estabelecer uma estrutura intermediária que armazene o conhecimento do domínio e que, através dela, se estabeleça uma interface para uma base de conhecimento e para uma linguagem de manipulação do conhecimento armazenado.

A figura (V.1) ilustra essa necessidade, onde as definimos como funções básicas para um sistema de representação do conhecimento.

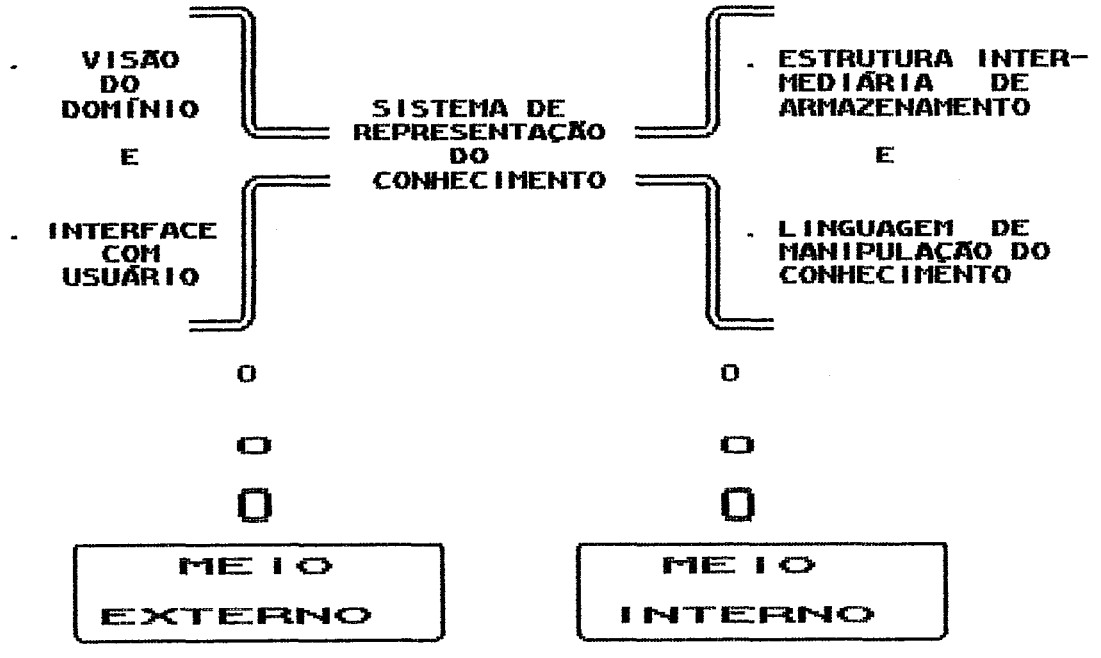


Figura V.1: Funções básicas do sistema de representação do conhecimento.

Com relação à linguagem de manipulação do conhecimento (LMC) é necessário identificar e especificar suas características fundamentais, para que se torne um recurso capaz de dinamizar todo o sistema de representação.

Uma outra questão importante que identificamos é a necessidade de que o sistema de representação seja adequado às características dos conhecimentos que deverão estar armazenados na sua base de conhecimento. Desta forma, para a definição do sistema de representação num domínio específico, que no nosso caso é o dos métodos de desenvolvimento de software, é importante identificar que tipos de conhecimentos são necessários para a representação desse domínio e de como estarão esses conhecimentos disponíveis, a partir de uma estrutura intermediária.

V.2 - RESULTADOS DESEJADOS

Nesta seção são analisadas as questões mais importantes para atender as necessidades identificadas no ítem anterior e, portanto, permitir especificar um sistema adequado conforme essas análises.

As questões mais importantes são:

- Identificar os tipos de conhecimentos adequados ao domínio tratado pelo sistema de representação;
- Estabelecer o método de representação do conhecimento a ser adotado pelo sistema. Para isto, consideramos importante uma análise de adequação dos métodos de representação;

. Analisar as características básicas da estrutura intermediária para o armazenamento intermediário do conhecimento;

. Analisar as características da linguagem de manipulação do conhecimento necessárias para a operacionalidade do sistema de representação.

Quanto à característica de permitir uma visualização adequada do conhecimento, este requisito está diretamente relacionado com o método de representação adotado pelo sistema de representação.

Quanto à eficácia de uma interface com o usuário, este requisito está diretamente relacionado com as características de implementação do sistema, o que não estamos considerando como escopo deste trabalho.

V.2.1 - TIPOS DE CONHECIMENTOS.

Como dissemos na seção (III.1), a representação do conhecimento para Inteligência Artificial acaba se tornando uma questão semelhante aos métodos de desenvolvimento de software na Engenharia de Software. Ou seja, não existe uma forma melhor, mais eficiente ou mais fácil. Existe sim, uma adequação melhor ao problema que se quer resolver.

Assim, analisaremos agora o domínio objeto deste trabalho, ou seja, o âmbito do contexto que envolve os métodos de desenvolvimento de software.

O interesse neste momento é o de poder identificar quais os conhecimentos relevantes dos métodos de desenvolvimento que necessitam ser representados num sistema de representação e, por conseguinte, os conhecimentos que seriam armazenados numa base de conhecimento.

Inicialmente, buscamos o caminho de se obter o meta-conhecimento sobre métodos de desenvolvimento de software, a fim de que, a partir desse resultado, pudessemos identificar as principais características dos conhecimentos envolvidos neste contexto. No entanto, não encontramos em nossa pesquisa realizada na literatura de ES, trabalhos voltados para esta finalidade ou que permitissem sua extensão. A partir daí, buscamos uma literatura metodológica, onde encontramos formulações de metodologias e não do método. Pesquisamos, então, a instituição IBGE, por se tratar de um orgão governamental que lida com metodologias de pesquisas estatísticas, onde o resultado foi o mesmo que o anterior. Aí, resolvemos re-analisar, agora com este enfoque, a área de IA. Constatamos que os exemplos de sistemas de representação do conhecimento, tais como, o sistema CENTAUR (BUCHANAN-84), o sistema KL-ONE (WOODS-83), o sistema PSN (MYLOPOULOS-83) e o sistema KRYPTON (BRACHMAN-83), que lidam com conhecimento mas não apresentam um meta-conhecimento de seu domínio de interesse.

Concluimos, então, dever iniciar nossa análise com a definição clássica de método: "Caminho para se chegar a um fim, através do qual se atinge um objetivo, ainda que esse caminho não tenha sido fixado à priori, de modo deliberado e refletido" (FERREIRA-75).

No entanto, pela falta de um consenso entre os diversos autores da ES em empregar uma definição única para os termos da Engenharia de Software, apresentamos as definições dos principais termos que deverão ser utilizados no sistema de representação, segundo (CRISPIM-88).

. MÉTODOS

Método é o conjunto de diretivas para a seleção e aplicação sistemática de técnicas e instrumentos, de forma a organizar o pensamento e o trabalho do usuário ao longo do processo de desenvolvimento de um produto de software. Portanto, os métodos são compostos de técnicas e instrumentos.

. TÉCNICAS

Técnica é o conjunto de princípios para a execução de uma tarefa específica do processo de desenvolvimento de software.

As técnicas existentes em métodos para o desenvolvimento de software podem ser classificadas da seguinte forma: técnicas construtivas, normativas e gerenciais.

. INSTRUMENTOS

Instrumentos são convenções notacionais e

operações automatizáveis que suportam atividades específicas do processo de desenvolvimento de software. Os instrumentos tornam possível a utilização de técnicas.

Os instrumentos existentes em métodos para o desenvolvimento de software podem ser classificados em: instrumentos construtivos, normativos e gerenciais.

. FERRAMENTAS

Ferramenta é a implementação computacional de um instrumento, de determinado método, em um dado ambiente de desenvolvimento.

As ferramentas podem ser de três tipos: ferramentas para construção, avaliação e apoio à gerência.

Agora, para facilitar nossa análise, apresentamos as definições acima numa forma estrutural. Sendo assim, podemos ilustrar os conceitos conforme a figura (V.2).

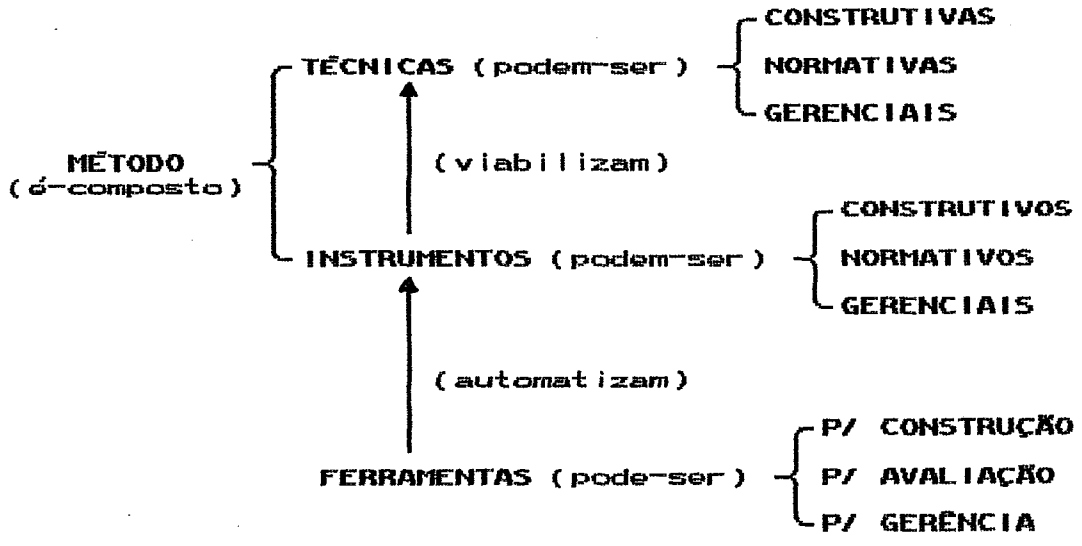


Figura V.2: Estrutura conceitual de métodos.

A partir dessa estrutura conceitual observamos a existência de técnicas, instrumentos e/ou ferramentas nos métodos de desenvolvimento de software e que esses métodos contém diretivas para sua utilização.

É em função das características particulares dos componentes dessa estrutura conceitual, que um método se difere de outro, além de seu objetivo básico da forma de lidar com o problema a ser tratado pelo sistema de desenvolvimento. Esse objetivo é conhecido na ES, como o tipo de abordagem do método, por exemplo, abordagem funcional, de dados ou de objetos.

Portanto, nosso sistema de representação do conhecimento de métodos de desenvolvimento de software, deve ser capaz de lidar com os conhecimentos relativos aos componentes da estrutura conceitual, permitindo classificar e identificar as características do método representado. É importante que o sistema de representação, além de especificar as diretivas do método, permita especificar as características particulares de cada um de seus componentes.

Para expressar os métodos e suas características particulares é necessário:

> Estabelecer a forma de proceder do método, ou seja, como se origina e se deriva. Deve-se expressar o seu comportamento através de um conjunto de atitudes. À este tipo de conhecimento denominamos "PROCEDIMENTOS";

. Identificar todo o conhecimento que pode ser apreendido, ou seja, tudo aquilo que pode ser manipulado e perceptível para o entendimento do método. A partir daí, é necessário e possível expressar o tipo de associação entre esses conhecimentos identificados. À este tipo de conhecimento denominamos "OBJETOS/RELAÇÕES";

. Expressar o que regula e dirige um objeto do método, indicando o modo correto ou condição de utilizá-lo, permitindo-se estabelecer formas de decisão sobre os seus conhecimentos. À este tipo de conhecimento denominamos "REGRAS DE DECISÃO";

. Permitir se fazer a descrição dos objetos do método, ou seja, expor de forma circunstanciada e narrativa o objetivo e/ou característica do conhecimento identificado. À este tipo de conhecimento denominamos "DESCRIÇÕES";

. Determinar a distribuição em classes e/ou grupos, segundo um sistema ou método de classificação, ou seja, os conhecimentos que representam categorias. À este tipo de conhecimento denominamos "CLASSIFICAÇÃO", e,

. Especificar verdades de formação irregular a partir de experiências de especialistas. À este tipo de conhecimento denominamos "HEURÍSTICAS".

Sendo assim, uma base de conhecimento para métodos de desenvolvimento de software, deve conter os seguintes conhecimentos, como ilustrado na figura (V.3).

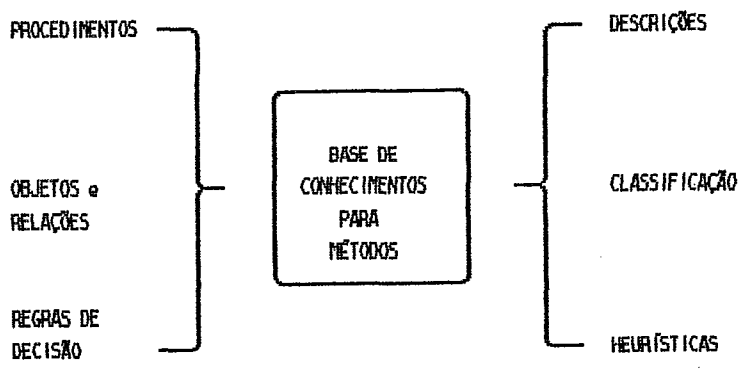


Figura V.3: Tipos de conchecimentos que podem ser representados numa base de conhecimento de métodos de desenvolvimento.

V.2.2 - ANÁLISE DE ADEQUAÇÃO DOS MÉTODOS DE REPRESENTAÇÃO.

Nos primórdios dos sistemas de IA, a representação do conhecimento não era reconhecida como um tema importante em sí mesma, embora a maioria desses sistemas incorporassem o conhecimento indiretamente, através de regras e estruturas de dados. Hoje em dia, os métodos de representação mais importantes são os sistemas de produção, quadros, lógica de primeira ordem e redes

semânticas, segundo (McCALLA-83). A partir de nossa pesquisa no capítulo (III), completamos esta lista com os métodos das dependências conceituais, roteiros e redes neuronais, os quais, ainda, não tem a mesma popularidade prática em relação aos quatro primeiros métodos.

Com relação às redes semânticas, embora possam haver tipos diferentes de redes, conforme (BRACHMAN-83), quase todas consistem de uma estrutura de dados de nós (representando conceitos) e um conjunto de procedimentos de inferência especializados que operam na estrutura de dados. Uma estrutura de dados em rede bastante popular é uma hierarquia de nós ligados por conectivos IS-A (é-um).

IS-A é um termo usado para descrever a existência de uma relação de generalização entre um subconceito como "cadeira" e seu superconceito "móveis". ou seja, "Uma cadeira IS-A (é-um) móvel". O tipo de inferência mais popular envolve a herança de informação desde os níveis superiores até a base da hierarquia desses conectivos IS-A. Tal organização permite que a informação seja dividida entre muitos nós, proporcionando uma representação do conhecimento distribuida sobre a rede. Entretanto, conforme discutido por (WOODS-83), (SCHUBERT-76) e (SCHUBERT-83), IS-A também suscita alguns temas formais complicados como, por exemplo, se deve haver distinção entre informações sobre classes e informações sobre instâncias de classes e como lidar com as exceções, que ainda são matérias de intensa pesquisa.

A utilidade da lógica de primeira ordem em um contexto de representação do conhecimento levou a uma intensa discussão a respeito dos prós e contras desse tipo de método. Tem-se demonstrado preocupação com a falta de um esquema explícito para indexar o conhecimento relevante, com a ineficácia para se lidar com conhecimento incompleto ou em mutação e com as limitações da inferência dedutiva. No entanto, os defensores da lógica apresentam contra-argumentos para muitas destas inquietações e, sem dúvida, a precisão formal e a interpretabilidade da lógica são úteis e oferecem a expressividade que outros métodos de representação de conhecimento não possuem.

(MINSKY-75) coloca que uma maneira útil de se organizar uma base de conhecimento seria dividi-la em partes modulares denominadas quadros. Os quadros tornaram-se a base de uma outra escola de representação do conhecimento. A divisão da base de conhecimento em quadros tornou-se comum em várias aplicações. Frequentemente faz-se uma distinção entre roteiros com pequena capacidade para inferência e quadros melhor orientado a nível de procedimento.

As arquiteturas de sistemas de produção são uma outra maneira de se representar o conhecimento. Esses sistemas foram apresentados originalmente como modelos de argumentação humana. Um conjunto de regras de produção (cada uma delas sendo essencialmente um par "situação-ação") operam em um "buffer" de conceitos, embora as versões recentes tendam a possuir uma memória de conceitos

ilimitada. A representação do conhecimento em pares situação-ação mostrou ser um modo natural de se extrair e codificar o conhecimento baseado em regra e, atualmente, os sistemas de produção são largamente utilizados para a construção de sistemas baseados no conhecimento.

Tendo em vista que, diversos sistemas de representação se baseiam em métodos distintos de representação, ou na combinação deles, podemos considerar que alguns domínios do conhecimento, ou tipos de conhecimentos de um mesmo domínio, se enquadram melhor em métodos específicos de representação.

Portanto, a partir das características dos métodos de representação do conhecimento, elaboramos o quadro ilustrado pela figura (V.4), onde apresentamos a sua relação com os fatores de qualidade descritos no capítulo (IV), buscando expressar o que consideramos ser adequado ou não.

Um caminho viável a ser seguido para contribuir na escolha de um método de representação do conhecimento, poderia ser através da adequação das características essenciais de cada método de representação com as características do domínio envolvido.

		LÓGICA DE PRIMEIRA ORDEM	REDES SEMÂNTICAS	QUADROS	SISTEMA DE PRODUÇÃO
ADEQUAÇÃO	VISUAL	NÃO	SIM (+)	NÃO	NÃO
	REPRESENTACIONAL	NÃO	SIM (+)	SIM (-)	SIM (-)
	INFERENCIAL	NÃO	SIM (+)	NÃO	NÃO
	AO PROBLEMA	NÃO	SIM (-)	SIM (+)	NÃO
	ESTRUTURA INTERMEDIÁRIA	NÃO	NÃO	SIM (+)	NÃO
	LINGUAGEM DE MANIPULAÇÃO	NÃO	SIM (+)	SIM (+)	SIM (+)
INTELIGÊNCIA	FATOS MUTÁVEIS	NÃO	NÃO	SIM (+)	SIM (+)
	FATOS NÃO-MUTÁVEIS	NÃO	SIM (+)	SIM (+)	SIM (+)
	ESTADO PROBLEMA	NÃO	NÃO	NÃO	SIM (-)
MAPEABILIDADE	FATOS P/ REPRESENTAÇÃO	NÃO	SIM (+)	SIM (+)	SIM (+)
	REPRESENTAÇÃO P/ FATOS	NÃO	SIM (+)	SIM (+)	SIM (+)
EFICIÊNCIA	INFERENCIAL	NÃO	SIM (-)	NÃO	SIM (+)
	AQUISICIONAL	NÃO	SIM (-)	SIM (+)	SIM (-)
ORIGINALIDADE		NÃO	NÃO	SIM (-)	SIM (-)
TRANSPARÊNCIA		NÃO	SIM (-)	SIM (+)	SIM (-)
MODULARIDADE		NÃO	NÃO	SIM (+)	SIM (-)

Figura V.4: Adequação dos métodos de representação do conhecimento.

Uma vez que já temos concluído os tipos de conhecimentos envolvidos no domínio de nosso interesse, elaboramos os quadros ilustrados pelas figuras (V.5), (V.6), (V.7) e (V.8), onde buscamos expressar a adequação de cada método de representação com os fatores de qualidade e com os tipos de conhecimentos, já referenciados anteriormente.

LÓGICA DE PRIMEIRA ORDEM		PROCEDIMENTOS	OBJETOS/ RELAÇÕES	REGRAS DE DECISÃO	DESCRIÇÕES	CLASSIFICAÇÃO	HEURÍSTICAS
ADEQUAÇÃO	VISUAL	NÃO	SIM (-)	NÃO	NÃO	SIM (-)	NÃO
	REPRESENTACIONAL	NÃO	SIM (-)	NÃO	NÃO	SIM (-)	NÃO
	INFERENCIAL	NÃO	NÃO	NÃO	NÃO	NÃO	NÃO
	AO PROBLEMA	NÃO	NÃO	NÃO	NÃO	NÃO	NÃO
	ESTRUTURA INTERMEDIÁRIA	NÃO	SIM (-)	NÃO	NÃO	SIM (-)	NÃO
	LINGUAGEM DE MANIPULAÇÃO	NÃO	SIM (-)	NÃO	NÃO	SIM (-)	NÃO
INTELIGÊNCIA	FATOS MUTÁVEIS	NÃO	NÃO	NÃO	NÃO	NÃO	NÃO
	FATOS NÃO-MUTÁVEIS	NÃO	NÃO	NÃO	NÃO	NÃO	NÃO
	ESTADO PROBLEMA	NÃO	NÃO	NÃO	NÃO	NÃO	NÃO
MAPEABILIDADE	FATOS P/ REPRESENTAÇÃO	NÃO	NÃO	NÃO	NÃO	NÃO	NÃO
	REPRESENTAÇÃO P/ FATOS	NÃO	NÃO	NÃO	NÃO	NÃO	NÃO
EFICIÊNCIA	INFERENCIAL	NÃO	NÃO	NÃO	NÃO	SIM (+)	NÃO
	AQUISICIONAL	NÃO	NÃO	NÃO	NÃO	NÃO	NÃO
ORIGINALIDADE		NÃO	NÃO	NÃO	NÃO	NÃO	NÃO
TRANSPARÊNCIA		NÃO	NÃO	NÃO	NÃO	NÃO	NÃO
MODULARIDADE		NÃO	NÃO	NÃO	NÃO	NÃO	NÃO

Figura V.5: Adequação dos conhecimentos dos métodos de desenvolvimento na lógica de primeira ordem.

REDES SEMÂNTICAS		PROCEDIMENTOS	OBJETOS/ RELAÇÕES	REGRAS DE DECISÃO	DESCRIÇÕES	CLASSIFICAÇÃO	HEURÍSTICAS
ADEQUAÇÃO	VISUAL	NÃO	SIM (+)	NÃO	NÃO	SIM (+)	NÃO
	REPRESENTACIONAL	NÃO	SIM (+)	NÃO	NÃO	SIM (+)	NÃO
	INFERENCIAL	NÃO	SIM (+)	SIM (+)	NÃO	SIM (+)	NÃO
	AO PROBLEMA	NÃO	SIM (+)	SIM (+)	NÃO	SIM (+)	NÃO
	ESTRUTURA INTERMEDIÁRIA	NÃO	NÃO	NÃO	NÃO	NÃO	NÃO
	LINGUAGEM DE MANIPULAÇÃO	NÃO	SIM (+)	SIM (-)	NÃO	SIM (+)	NÃO
INTELIGÊNCIA	FATOS MUTÁVEIS	NÃO	NÃO	NÃO	NÃO	NÃO	NÃO
	FATOS NÃO-MUTÁVEIS	NÃO	SIM (+)	NÃO	NÃO	SIM (+)	NÃO
	ESTADO PROBLEMA	NÃO	SIM (-)	NÃO	NÃO	SIM (-)	NÃO
MAPEABILIDADE	FATOS P/ REPRESENTAÇÃO	NÃO	SIM (+)	NÃO	NÃO	SIM (+)	NÃO
	REPRESENTAÇÃO P/ FATOS	NÃO	SIM (+)	NÃO	NÃO	SIM (+)	NÃO
EFICIÊNCIA	INFERENCIAL	NÃO	SIM (+)	NÃO	NÃO	SIM (+)	NÃO
	AQUISICIONAL	NÃO	SIM (+)	NÃO	NÃO	SIM (+)	NÃO
ORIGINALIDADE		NÃO	NÃO	NÃO	NÃO	NÃO	NÃO
TRANSPARÊNCIA		NÃO	SIM (-)	NÃO	NÃO	SIM (-)	NÃO
MODULARIDADE		NÃO	NÃO	NÃO	NÃO	NÃO	NÃO

Figura V.6: Adequação dos conhecimentos dos métodos de desenvolvimento nas redes semânticas.

QUADROS		PROCEDIMENTOS	OBJETOS/ RELAÇÕES	REGRAS DE DECISÃO	DESCRIÇÕES	CLASSIFICAÇÃO	HEURÍSTICAS
ADEQUAÇÃO	VISUAL	SIM (+)	SIM (-)	NÃO	SIM (-)	SIM (+)	NÃO
	REPRESENTACIONAL	SIM (+)	SIM (+)	NÃO	SIM (+)	SIM (+)	NÃO
	INFERENCIAL	NÃO	SIM (-)	NÃO	NÃO	SIM (-)	NÃO
	AO PROBLEMA	SIM (+)	SIM (-)	NÃO	SIM (+)	SIM (-)	NÃO
	ESTRUTURA INTERMEDIÁRIA	SIM (+)	SIM (+)	SIM (+)	SIM (+)	SIM (+)	SIM (+)
	LINGUAGEM DE MANIPULAÇÃO	SIM (+)	SIM (+)	SIM (+)	SIM (+)	SIM (+)	SIM (+)
INTELIGÊNCIA	FATOS MUTÁVEIS	SIM (+)	SIM (-)	SIM (+)	SIM (+)	SIM (+)	SIM (+)
	FATOS NÃO-MUTÁVEIS	SIM (+)	SIM (-)	SIM (+)	SIM (+)	SIM (+)	SIM (+)
	ESTADO PROBLEMA	SIM (+)	SIM (-)	SIM (+)	SIM (+)	SIM (+)	SIM (+)
MAPEABILIDADE	FATOS P/ REPRESENTAÇÃO	SIM (+)	SIM (+)	SIM (+)	SIM (+)	SIM (+)	SIM (+)
	REPRESENTAÇÃO P/ FATOS	SIM (+)	SIM (+)	SIM (+)	SIM (+)	SIM (+)	SIM (+)
EFICIÊNCIA	INFERENCIAL	NÃO	SIM (+)	NÃO	NÃO	SIM (+)	NÃO
	AQUISICIONAL	SIM (+)	SIM (+)	NÃO	SIM (+)	SIM (+)	NÃO
ORIGINALIDADE		SIM (-)	SIM (-)	NÃO	SIM (-)	SIM (-)	NÃO
TRANSPARÊNCIA		SIM (-)	SIM (+)	NÃO	SIM (-)	SIM (-)	NÃO
MODULARIDADE		SIM (+)	SIM (+)	SIM (+)	SIM (+)	SIM (+)	SIM (+)

Figura V.7: Adequação dos conhecimentos dos métodos de desenvolvimentos nos quadros.

SISTEMAS DE PRODUÇÃO		PROCEDIMENTOS	OBJETOS/ RELAÇÕES	REGRAS DE DECISÃO	DESCRIÇÕES	CLASSIFICAÇÃO	HEURÍSTICAS
ADEQUAÇÃO	VISUAL	NÃO	NÃO	SIM (-)	NÃO	NÃO	NÃO
	REPRESENTACIONAL	NÃO	SIM (-)	SIM (+)	NÃO	SIM (-)	SIM (+)
	INFERENCIAL	NÃO	NÃO	SIM (-)	NÃO	NÃO	SIM (-)
	AO PROBLEMA	NÃO	NÃO	SIM (+)	NÃO	NÃO	SIM (+)
	ESTRUTURA INTERMEDIÁRIA	NÃO	NÃO	SIM (+)	NÃO	NÃO	SIM (+)
	LINGUAGEM DE MANIPULAÇÃO	NÃO	SIM (+)	SIM (+)	NÃO	NÃO	SIM (+)
INTELIGÊNCIA	FATOS MUTÁVEIS	NÃO	SIM (-)	SIM (+)	NÃO	NÃO	SIM (+)
	FATOS NÃO-MUTÁVEIS	NÃO	SIM (-)	SIM (+)	NÃO	NÃO	SIM (+)
	ESTADO PROBLEMA	NÃO	SIM (-)	SIM (+)	NÃO	NÃO	SIM (+)
MAPEABILIDADE	FATOS P/ REPRESENTAÇÃO	NÃO	SIM (+)	SIM (+)	NÃO	NÃO	SIM (+)
	REPRESENTAÇÃO P/ FATOS	NÃO	SIM (+)	SIM (+)	NÃO	NÃO	SIM (+)
EFICIÊNCIA	INFERENCIAL	NÃO	SIM (+)	SIM (+)	NÃO	NÃO	SIM (+)
	AQUISICIONAL	NÃO	SIM (-)	SIM (-)	NÃO	NÃO	SIM (-)
ORIGINALIDADE		NÃO	SIM (-)	SIM (-)	NÃO	NÃO	SIM (-)
TRANSPARÊNCIA		NÃO	SIM (-)	SIM (-)	NÃO	NÃO	SIM (-)
MODULARIDADE		NÃO	NÃO	SIM (-)	NÃO	NÃO	SIM (-)

Figura V.8: Adequação dos conhecimentos dos métodos de desenvolvimento nos sistemas de produção.

V.2.3 - CARACTERÍSTICAS BÁSICAS DA ESTRUTURA INTERMEDIÁRIA.

Uma vêz identificados e analisados os tipos de conhecimentos envolvidos nos métodos de desenvolvimento, vamos analisar quais as características essenciais de uma estrutura intermediária para o armazenamento desses conhecimentos.

O ponto de partida para uma solução é o entendimento das relações existentes entre os dados que são relevantes para o problema (VELOSO-85).

Dentre os conhecimentos identificados para o domínio dos métodos de desenvolvimento destacamos o tipo referente à OBJETOS/RELAÇÕES como sendo o elemento central, por representar a base da percepção e do entendimento do método representado. Sobre este conhecimento, torna-se possível quando necessário, utilizar os tipos de conhecimento DESCRIÇÕES, PROCEDIMENTOS e REGRAS para expressarem as características essenciais do OBJETO identificado.

Embora não representando uma essência do método, o conhecimento relativo a HEURÍSTICAS permite estabelecer características particulares de orientações para o uso do método, através da referência aos OBJETOS. A partir deste é possível, ainda, identificar àqueles que representam alguma categoria em relação a um sistema de classificação considerado, através do tipo CLASSIFICAÇÃO.

Uma relação hierárquica que podemos apresentar para os conhecimentos identificados para o domínio dos

métodos de desenvolvimento, é ilustrado na figura (V.9).

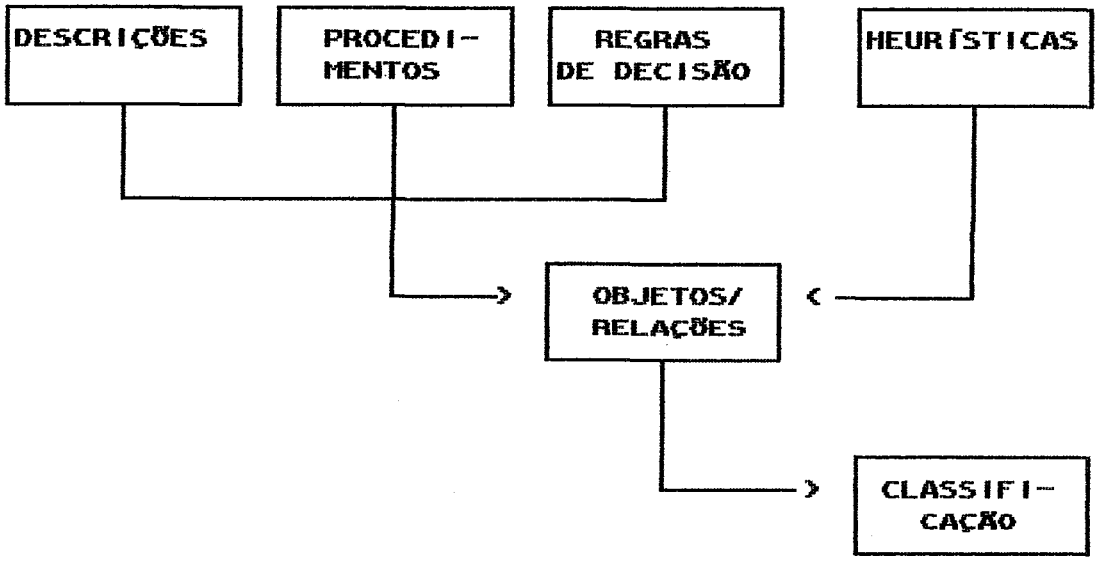


Figura V.9: Relacionamento entre os conhecimentos do domínio dos métodos de desenvolvimento.

A estrutura intermediária significa onde estarão armazenados os conhecimentos especificados para o método de desenvolvimento, que por sua vez podem ser acessados pelo sistema de representação.

É importante ressaltar, no entanto, que a estrutura intermediária, conforme descrita no capítulo (IV), se relaciona ao fator de qualidade de adequação representacional, resultando na capacidade de interfacear o conhecimento armazenado com a base de conhecimento e com a capacidade de apresentá-los ao meio externo, agora através do fator de qualidade de adequação visual.

V.2.4 - CARACTERÍSTICAS DA LINGUAGEM DE MANIPULAÇÃO DO CONHECIMENTO.

Com relação à linguagem de manipulação do conhecimento (LMC), esta deve ser construída de forma a permitir as operações necessárias para o interfaceamento objetivado pela estrutura intermediária e permitir o dinamismo operacional nas demais funções estabelecidas para o sistema de representação.

Tendo em vista que, não entraremos em considerações a respeito das características de construção da base de conhecimento, a LMC deve se restringir, no escopo deste trabalho, a atender às necessidades funcionais que o sistema de representação tornará disponível para permitir a identificação, o acesso e a manipulação dos conhecimentos envolvidos no domínio dos métodos de desenvolvimento.

É necessário ressaltar que não desejamos a construção de um sistema de representação que seja dependente da LMC apresentada neste trabalho, o que buscamos é caracterizar uma LMC que realize as funções e operações estabelecidas pelo sistema de representação.

Portanto, o sistema de representação não deve ter uma dependência da LMC e sim, a dependência da existência de uma LMC, desde que satisfaça às características definidas pelo sistema de representação.

As funções principais a serem tratadas pela LMC, são:

. permitir que o usuário tenha uma visão global do domínio, reconhecendo e entendendo suas características essenciais;

. permitir que o usuário realize consultas através dos componentes do sistema;

. permitir a montagem de uma estrutura intermediária que contenha a concepção e especificação dos conhecimentos do domínio, e,

. permitir o acesso aos conhecimentos definidos para o domínio, atualizando e possibilitando obter consultas e, por conseguinte, conclusões.

As operações principais a serem tratadas pela LMC, são:

. Operação de **CRIAÇÃO**, que permite a formação da base de conhecimento, através do armazenamento do conhecimento na estrutura intermediária;

. Operação de **ATUALIZAÇÃO**, que permite manter a base de conhecimento atualizada, através da estrutura intermediária;

. Operação de **RETIRADA**, que permite a retirada de conhecimento já existente na base de conhecimento, através da estrutura intermediária;

. Operação de **SELEÇÃO**, que permite estabelecer as características desejadas para a escolha de um objeto existente na base de conhecimento;

. Operação de **CONSULTA**, que permite estabelecer as características desejadas para o acesso ao conhecimento existente na base de conhecimento. O resultado da consulta estará na estrutura intermediária.

V.3 - CONCEPÇÃO DO SISTEMA.

O pensamento em termos de sistemas desempenha um papel fundamental em uma ampla área de atuação, que vão das empresas indústriais e de armamentos até tópicos diversos da ciência pura.

À algum tempo no passado surgiram profissões e empregos desconhecidos até então, tendo os nomes de projeto de sistemas, análise de sistemas, engenharia de sistemas e outros.

A tecnologia, no entanto, foi levada a pensar não em termos de máquinas isoladas mas em termos de "sistemas". Uma máquina a vapor, um automóvel ou um receptor de rádio, achavam-se dentro da competência do engenheiro treinado na respectiva especialidade. Mas quando se chega aos mísseis balísticos ou aos veículos espaciais, estes engenhos têm de ser constituídos pela reunião de componentes originados em tecnologias.

Um sistema pode ser definido como um complexo de elementos em interação. A interação significa que os

elementos p estão em relações R, de modo que o comportamento de um elemento p em R é diferente de seu comportamento em outra relação R'. Se os comportamentos em R e R' não são diferentes não há interação, e os elementos se comportam independentemente com respeito às relações R e R' (BERTALANFFY-73).

Com relação a sistemas de representação do conhecimento, o que se tem encontrado com maior sucesso, (FIKES-85) se refere a isto, é o desenvolvimento de sistemas baseados em mais de um método tradicional, formando uma utilização híbrida das facilidades de representação que combina as vantagens dos métodos de representação utilizados.

Podemos observar esta característica particular, nos sistemas de representação do conhecimento descritos no capítulo (III).

No sistema CENTAUR o conhecimento é representado utilizando os métodos de regras de produção e de quadros.

O sistema KRYPTON se desenvolve na essência dos métodos de quadros e da lógica de primeira ordem.

O sistema PSN se baseia no método de redes semânticas e nas funções propostas pelo método de quadros.

O sistema KL-ONE, baseado numa estrutura de conhecimento fatorada, tem a essência de sua técnica no método de redes semânticas e de regras de produção. Os conceitos do KL-ONE são similares, em estrutura, às noções

gerais do método de quadros.

Com o objetivo de direcionar para uma conclusão sobre o/os método/os a serem utilizados no nosso sistema, baseamos esta análise nos quadros elaborados nas figuras (V.5) à (V.8), onde relacionamos os métodos de representação considerados como os mais importantes com os fatores de qualidade descritos no capítulo (IV).

Inicialmente, observamos uma incidência acentuada de "negações" no quadro da figura (V.5), que expressa a adequação dos conhecimentos em lógica de primeira ordem.

No caso da figura (V.6), referente às redes semânticas, observamos que quatro, dos seis tipos de conhecimentos, não são adequados para representação neste método. Os conhecimentos adequados neste caso, são: objetos/relações e classificação.

Na figura (V.7), referente aos quadros, observa-se que este método não seria adequado para dois dos seis tipos de conhecimentos, para o domínio dos métodos de desenvolvimento de software. Esses conhecimentos não adequados, em parte, seriam os referentes à regras e heurísticas. Embora, a não adequação seja para uma quantidade de fatores de qualidade equivalente aos que são adequados para o mesmo método.

Deve-se observar, ainda, que os conhecimentos adequados aos quadros, englobam os que são apropriados às redes semânticas mais os referentes a "procedimentos" e "descrições".

Com referência a figura (V.8), do método dos sistemas de produção, observa-se uma alternância nos conhecimentos adequados aos métodos anteriores e, principalmente, uma adequação a "regras" e "heurísticas", descartadas pelos demais.

No entanto, conforme (ARTHUR-85), a qualidade é o caminho para a produtividade e a qualidade de um produto de software é mensurável em diversas formas. Consideraremos neste trabalho, que o atendimento aos tipos de conhecimentos envolvidos no domínio dos métodos de desenvolvimento é mais importante do que satisfazer a todos os fatores de qualidade (é óbvio que o ideal seria satisfazer completamente às duas dimensões). Portanto, resumiremos num só quadro os percentuais relativos à cada tipo de conhecimento considerado adequado, obtidos com a divisão da quantidade de fatores com adequação sim(+) ou sim(-) por 16 (que é o total de fatores definidos), e, a partir daí, buscaremos concluir com quais métodos se pode atender aos tipos de conhecimento. A figura (V.10) ilustra esses percentuais.

Nesta análise, o método que mais se destaca é o QUADRO, pois encontramos uma adequação para todos os tipos de conhecimento, enquanto para os demais isto não ocorre. Por outro lado, é importante notar que, o método dos SISTEMAS DE PRODUÇÃO se destaca com um percentual alto de adequação exatamente para dois dos tipos de conhecimentos que tem os mais baixos percentuais no método dos QUADROS (esses conhecimentos são: regras e heurísticas).

	PROCEDI-MENTOS	OBJETOS/ RELAÇÕES	REGRAS DE DECISÃO	DESCRIÇÕES	CLASSIFI-CAÇÃO	HEURÍSTI-CAS
LÓGICA DE PRIMEIRA ORDEM	0	25,00	0	0	31,25	0
REDES SEMÂNTICAS	0	75,00	18,75	0	75,00	0
QUADROS	87,50	100,00	50,00	87,50	100,00	50,00
SISTEMA DE PRODUÇÃO	0	68,75	100,00	0	6,25	93,75

Figura V.10: Percentuais de adequação dos métodos de representação.

Sendo assim, com a tendência até o momento, a utilização dos métodos dos quadros e sistemas de produção satisfariam às nossas necessidades.

Contudo, como temos colocado ao longo do trabalho, é imprescindível para o nosso sistema ter uma boa adequação visual e ser de fácil construção e entendimento. Estes requisitos encontramos melhor no método das redes semânticas.

Agora, se voltarmos à figura (V.9), onde é apresentado o relacionamento entre os conhecimentos do

domínio de métodos de desenvolvimento, verificamos que o elemento central se refere aos "objetos/relações" referentes aos métodos de desenvolvimento, os quais podem ser bem representados pelas redes semânticas, inclusive para representar uma "classificação" desse conhecimento. Cada um desses "objetos/relações" podem receber uma "descrição" e ter "procedimentos" específicos a alguma tarefa, os quais podem ser bem representados pelos quadros. Para as "regras" e "heurísticas" referentes a esses "objetos/relações", podem ser bem representados pelo método dos sistemas de produção.

Desta forma, a concepção do nosso sistema de representação se baseia na seguinte estratégia, conforme figura (V.11).

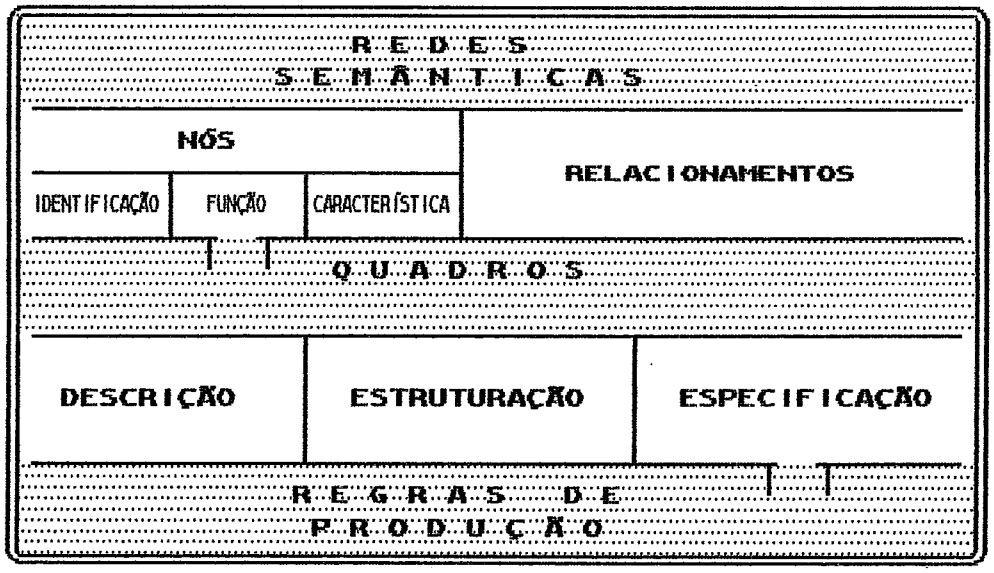


Figura V.11: Concepção estratégica do sistema de representação.

Baseado nesta estratégia, um método de desenvolvimento de software seria representado através do desenvolvimento de uma rede de conhecimento baseada nos conceitos das redes semânticas, com as conexões adequadas aos conhecimentos representados pelos nós. Para determinados nós, onde se requer uma caracterização maior do conhecimento rotulado pelo nó, deve-se considerá-lo como as características do método de quadro. A partir daí, utilizar regras de produção para a especificação de procedimentos, regras e/ou heurísticas, quando necessárias.

V.4 - MODELAGEM DO SISTEMA DE REPRESENTAÇÃO.

O sistema de representação proposto se baseia numa utilização híbrida das características dos métodos de Redes Semânticas e Quadros, com especificações baseadas em Regras de Produção.

Com as redes semânticas se obtém uma visão macroscópica do conhecimento de forma adequada e de fácil entendimento. Os nós identificam e rotulam o conhecimento e os relacionamentos, expressam as conexões apropriadas ao domínio tratado.

Para a descrição de cada nó da rede de conhecimento e para expressar as propriedades comuns em um conjunto de nós, utilizaremos os conceitos dos quadros para tal tarefa.

Os nós, na rede de conhecimento, representam os conhecimentos perceptíveis do método de desenvolvimento e que determinam uma característica essencial ou necessária

para o entendimento visual. Essas informações são relativas ao contexto do conhecimento do domínio, não devendo se relacionar com o contexto de solução de um problema para o mesmo domínio. Por exemplo, um conhecimento relativo ao contexto do domínio seria que a Análise Estruturada é um método para auxílio na análise e especificação de requisitos funcionais de um sistema de software, enquanto um contexto relacionado à solução de um problema seria que a entrada de dados para um programa será através de transferência de arquivos.

Essas informações (ou seja, o conhecimento do método de desenvolvimento) são obtidas a partir da observação de uma descrição ou de um relato ou da especificação de um especialista sobre o mesmo.

A identificação e descrição dessas informações é o ponto mais relevante para a representação do conhecimento, onde o nível de aceitação dos conhecimentos representados, está relacionado à satisfação dessa representação às necessidades atendidas por ela.

A organização de uma rede de conhecimento no sistema proposto, busca expressar conhecimentos individuais (utilização do conceito de nó da rede semântica) e conhecimentos genéricos (utilização do conceito de quadros). Isto é, não obstante a possibilidade de se representar ambos os conhecimentos em ambos os métodos de representação, ressaltamos que, a representação de conhecimento genérico numa rede semântica torna extenso o entendimento visual. Enquanto que, se considerarmos o

conhecimento genérico, que têm implícita a propriedade de herança para os conhecimentos de sua hierarquia descendente, se obtém uma rede de conhecimento mais simplificada, facilitando o entendimento visual do domínio. A figura (V.12) ilustra a apresentação visual do conhecimento no nosso sistema, onde os nós A,B,D,E,F,G e H representam conhecimentos individuais, enquanto o nó C representa um conhecimento genérico, que tem características de herança para os nós D,E,F e G.

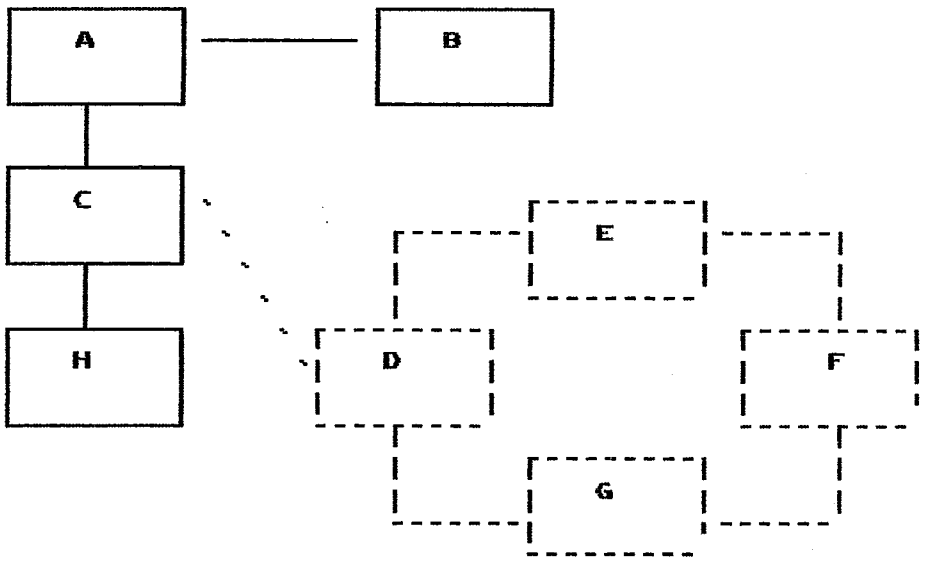


Figura V.12: Apresentação visual do conhecimento.

Tendo em vista que se pode pensar em um quadro como se fosse uma rede de nós e relações (MINSKY-75), entendemos que a rede de conhecimento expressa pelo nosso sistema, resulta no que denominamos **REDE DE CONEXÕES DE QUADROS (RCQ)**.

As construções desenvolvidas através de quadros, permitem uma organização que representa classes dentro de taxonomias. Essas classes, na RCQ, são representadas pelos nós. Com esse tipo de construção é possível projetar uma base de conhecimento descrevendo cada uma dessas classes como sub-classe ou super-classe de especialização de outras classes. Esse tipo de construção permite, ainda, uma otimização da rede de conhecimento, uma vez que se observando as classes (nós) mais gerais, já é possível o entendimento do conhecimento representado, reforçando a característica da RCQ ilustrada anteriormente.

Para o armazenamento intermediário da identificação e descrição do conhecimento representado pelo nó, define-se uma **ESTRUTURA DO CONHECIMENTO (EC)** que é agregada a cada nó.

A figura (V.13) ilustra a apresentação de um par de nós com suas respectivas estruturas de conhecimento.

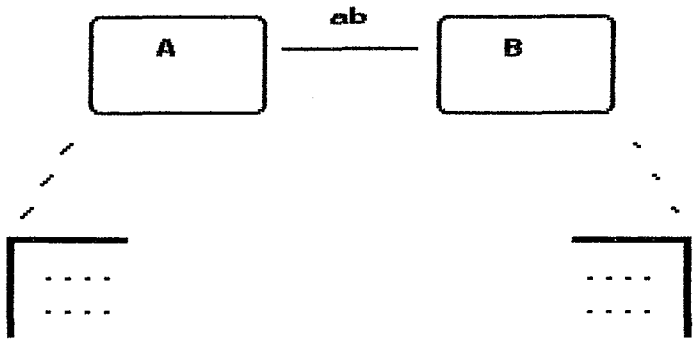


Figura V.13: Ilustração de uma rede de conhecimento.

A pesquisa sobre a representação do conhecimento pode envolver formas de representações de noções semânticas particulares como tempo, causalidade, crenças e intenções. Por outro lado, ela pode tomar a forma de um projeto de elaboração de linguagem, onde esta linguagem serve para a representação de conhecimento e os programas escritos na linguagem, ou seu uso, resultam numa base de conhecimento armazenando o conhecimento a respeito de algum domínio. Uma outra alternativa é a de que a pesquisa sobre a representação de conhecimento possa envolver o desenvolvimento de um auxílio de programação, para construir e utilizar bases de conhecimento (MYLOPOULOS-83).

Portanto, seguindo a noção de que os métodos de representação de conhecimento devem ser um recurso automatizado, ressaltamos que, embora não estejamos lidando com a tarefa de implementação, este trabalho descreve as características básicas de uma linguagem de manipulação do conhecimento, de forma a permitir o acesso e manipulação do conhecimento representado neste sistema.

Sendo assim, o sistema de representação que propomos é composto de três componentes básicos:

- **Rede de conhecimento**, composta de nós e relacionamentos;
- **Estrutura de conhecimento**;
- **Linguagem de manipulação do conhecimento (LMC)**.

A figura (V.14) ilustra a apresentação visual desses componentes.

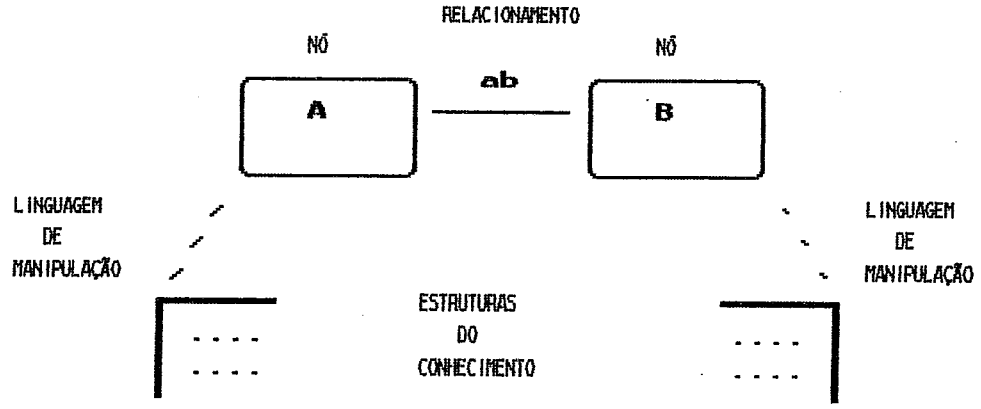


Figura V.14: Componentes do sistema de representação.

Como o nosso objetivo é a representação de métodos de desenvolvimento de software, esses componentes são definidos segundo características que buscam descrever seus conhecimentos, de forma que as representações resultem num recurso que permita identificá-los, conhecê-los e manipulá-los.

V.5 - DESCRIÇÃO DOS COMPONENTES DO SISTEMA DE REPRESENTAÇÃO.

Com o desenvolvimento de uma **rede de conhecimento**, formada de nós e relacionamentos, obtém-se uma facilidade para se expressar os conhecimentos básicos, essenciais e de fácil entendimento para o projetista e usuário.

De forma a tornar concisa a rede de conhecimento e, por outro lado, permitir uma descrição e especificação das características dos conhecimentos representados pela rede, o componente de **estrutura do conhecimento** permite esta tarefa através de um conjunto de estruturas definidas para tratar cada tipo especial, necessário para a representação do método, sem detrimento da adequação visual e através de uma adequada capacidade representacional.

Da mesma forma que um programa lento pode ser inútil, e da mesma forma que os sermões de uma igreja podem assustar os membros de outra, não se pode esperar pelo sucesso na representação do conhecimento de um domínio se não for possível dizer o que as representações significam (DOYLE-83).

Cabe ao componente da **linguagem de manipulação do conhecimento** (LMC) tratar com a questão semântica do sistema, como ainda, permitir o acesso e manipulação do conhecimento representado.

Para a descrição de cada componente utilizaremos como exemplo o método de desenvolvimento da Análise Estruturada (GANE-83), cuja descrição suscinta se encontra no apêndice A.

V.5.1 - REDE DE CONHECIMENTO.

Uma rede de conhecimento é expressa pelos conhecimentos especificados nos nós e do relacionamento associado entre eles.

Para cada nó da rede de conhecimento, deve ser estabelecido um conjunto de informações (conceitos do sistema de representação) que permitem organizar a rede e caracterizar o conhecimento representado pelo nó.

A partir daí, determina-se o relacionamento que melhor representa a conexão de um nó em referência com o seu consequente.

V.5.1.1 - NÓS.

Os nós da rede de conhecimento expressam os conhecimentos do tipo OBJETO/RELAÇÃO. Somente a partir deles é que se poderá manipular e desenvolver as funções desejadas para a manipulação, conforme seus relacionamentos e do conhecimento contido na sua estrutura de conhecimento.

Os conceitos referentes à especificação dos nós da rede de conhecmento, são:

. Representação gráfica;
. Identificação;
. Nós posterior e anterior;
. Sentido (direção) do relacionamento;
. Tipo;
. Função;
. Conexões E, OU e EOU;
. Representação de relacionamentos iguais;
. Segmentação;
. Imagem modular;
. Continuidade na rede de conhecimento.

REPRESENTAÇÃO GRÁFICA

Cada nó é representado por um retângulo arredondado, como mostra a figura (V.15).

Figura V.15: Representação gráfica de um nó.

IDENTIFICAÇÃO

Cada nó é identificado na rede por um **nome** e um **número**. O nome representa a identificação do conhecimento representado pelo nó. A numeração de um nó representa uma forma de acesso ao conhecimento alí representado. Essa numeração deve ser crescente, porém, não significando ser uma evolução lógica do nó na rede e, sim, a evolução de sua construção e atualização. A numeração do nó deve aparecer na parte superior esquerda do retângulo, como ilustrado na figura (V.16).

Figura V.16: Identificação de um nó.

Tanto o nome como o número de um nó, são definidos para permitir, além de sua identificação, se fazer referências utilizando-os na LMC.

Portanto, com relação à identificação numérica, ela deve ser gerada de forma automática, quando da criação ou atualização de um nó. Quando da retirada de um nó na rede, a identificação disponível não deverá ser reaproveitada até que se utilize um recurso do sistema que reorganize toda a identificação numérica da rede.

Sendo assim, reservamos para o primeiro nó da rede, ou seja, o de número 1 (um), para ser o que identifica a rede de conhecimento. Isto é, o nó de número 1 representa o nó raiz da rede de conhecimento e, portanto, sua descrição se referirá a toda rede, como ainda, sua descrição poderá ter estruturas do conhecimento diferentes dos demais nós.

Neste caso, o nome que identifica o nó de número 1 (um) é o nome do método de desenvolvimento representado pela rede de conhecimento.

NÓS POSTERIOR E ANTERIOR

De forma similar ao conceito de heranças nas conexões IS-A (BRACHMAN-83) e ao conceito de nível de atividade nas estruturas taxonômicas do KL-ONE (WOODS-83), onde a posição dos nós nas redes tem um significado organizacional, consideramos como nó-posterior aquele que sucede um nó na rede a partir de uma sequência das relações até eles em função do nó raiz. No caso inverso, ou seja, do

nó anterior, se refere àqueles que vêem imediatamente antes, em função da organização da rede partir do nó raiz. Como exemplo, considere a figura (V.17).

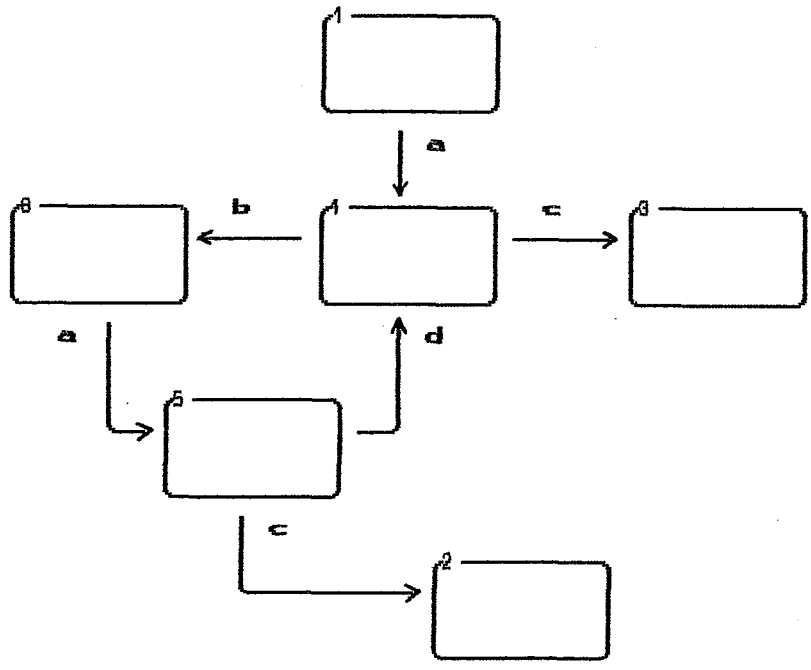


Figura V.17: Ilustração de nós anterior e posterior.

Na figura anterior, enumeramos cada nó arbitrariamente para demonstrar que a identificação numérica não interfere na caracterização de nós anterior e posterior.

Sendo assim, a figura (V.18) ilustra a caracterização de nós posterior e anterior em função da rede da figura (V.17).

	nós-anterior	nós-posterior
nó 1	-	nó 4
nó 4	nó 1	nós 6, 5, 3
nó 6	nó 4	nó 5
nó 5	nós 6, 4	nó 2
nó 2	nó 5	-
nó 3	nó 4	-

Figura V.18: Nós anterior e posterior.

SENTIDO (DIREÇÃO) DO RELACIONAMENTO

É importante observar que a identificação de nós anterior e posterior independe do sentido da relação entre os dois nós. Todavia, para o nosso sistema, o sentido da relação entre dois nós é uma característica fundamental.

Sendo assim, o nó 1 é origem em relação ao nó 4 com o tipo de relacionamento "a", em contra partida, o nó 4 é destino em relação ao nó 1 com o tipo de relacionamento "a".

Como ilustração, vamos supor dois nós denominados "João" e "Maria", e que a relação entre eles é "gostar". Se considerarmos o sentido da relação indo de "João" para "Maria", vai significar que "João gosta de Maria", o que não é necessariamente o mesmo significado no caso do relacionamento indo de "Maria" para "joão".

TIPO

Com o objetivo de identificar os conhecimentos essenciais e de permitir expressar os conhecimentos que venham tornar mais claro o entendimento da rede de conhecimento, é definido para cada nó um "tipo", que diferencia esses casos.

Então, nas redes de conhecimento, podem existir nós significando o que denominamos de **nós fracos**. Os nós fracos significam que são nós dispensáveis, ou seja, existem para compor um raciocínio melhor na rede, onde a sua ausência não deve alterar o conhecimento essencial do domínio. A figura (V.19) ilustra este caso.

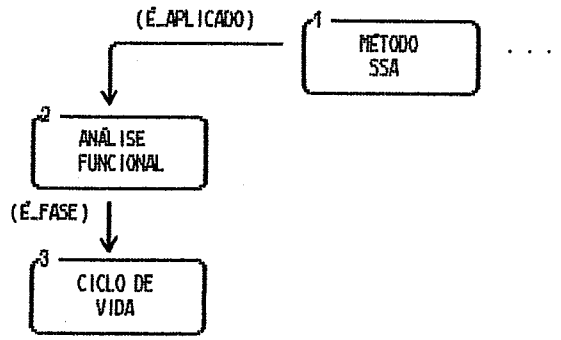


Figura V.19: Ilustração do tipo de nó.

O nó de número 3, representado no exemplo da figura (V.19), é considerado fraco devido sua ausência não alterar o conhecimento da rede. No entanto, sua inclusão é

providencial para que, além de tornar a organização mais clara, permitir uma estruturação da rede que facilitará o manuseio do conhecimento representado, através de outros recursos existentes neste sistema.

Os nós que representam um conhecimento essencial, são denominados nós ativos. Esses nós não podem ser retirados da rede de conhecimento, pois sua ausência influi no entendimento do método representado.

FUNÇÃO

Com o objetivo de expressar o conhecimento através dos nós e de descrevê-los de forma simplificada ou através de estruturas do conhecimento que identificam e descrevem características particulares dos nós, é definida para cada nó uma "função", que pode ser "R" para nó de função "rede" e "Q" para nó de função "quadro".

A função "R" determina um nó de características dos métodos de redes semânticas, ou seja, onde o conhecimento é expresso pelo próprio nó e por uma estrutura de conhecimento simplificada, o suficiente para identificação e descrição.

A função "Q" determina um nó de características dos métodos dos quadros, ou seja, o conhecimento é expresso pelas informações contidas em suas estruturas de conhecimento, além da estrutura simplificada que permite a sua identificação e descrição.

A figura (V.20) ilustra uma rede de conhecimento

contendo nós com ambas as funções.

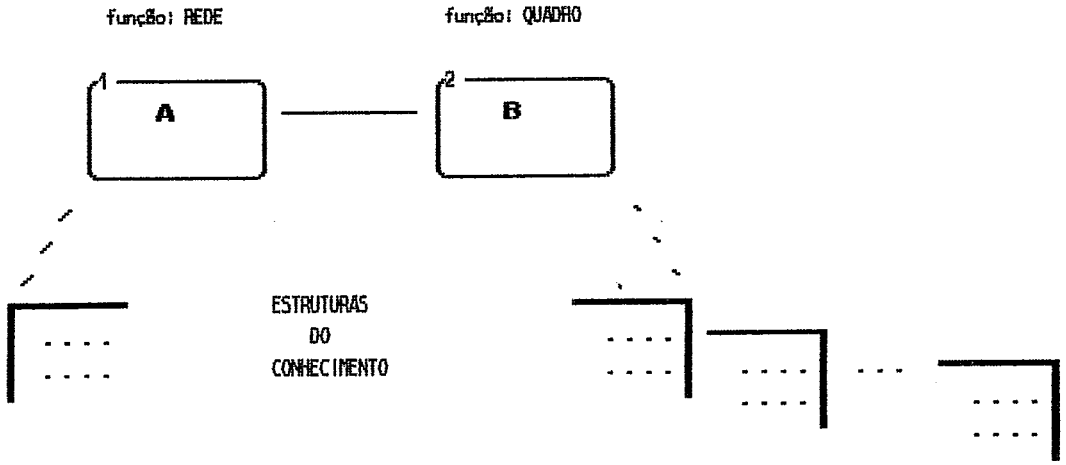


Figura V.20: Representação das funções do nó.

CONEXÕES E, OU e EOU

A fim de permitir uma melhor organização e entendimento das características de conexão dos nós na rede de conhecimento, são definidos três elementos de conexão: conexões E, OU e EOU.

Uma conexão E determina que todos os nós posteriores a ele são obrigatórios em função dos tipos de relacionamentos que os ligam. A figura (V.21) ilustra este tipo de elemento.

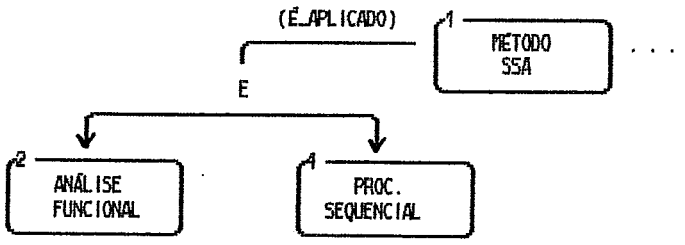


Figura V.21: Representação do elemento de conexão "E".

Uma conexão do tipo OU determina que somente um (ou alguns, dependendo da condição de seleção) dos nós posteriores a ele é para ser considerado. A determinação da escolha do nó a ser utilizado estará definida na estrutura do conhecimento dos nós envolvidos. A figura (V.22) ilustra este caso.

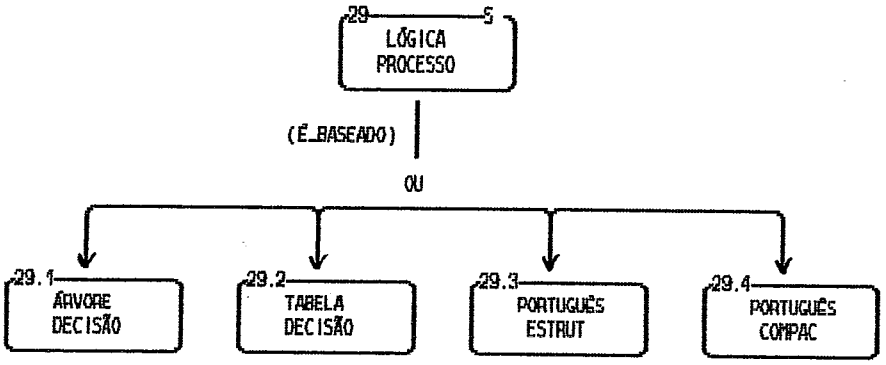


Figura V.22: Representação do elemento de conexão "OU".

Uma conexão do tipo EOU determina ambas as possibilidades, ou seja, da conexão E e da conexão OU. A figura (V.23) ilustra este caso.

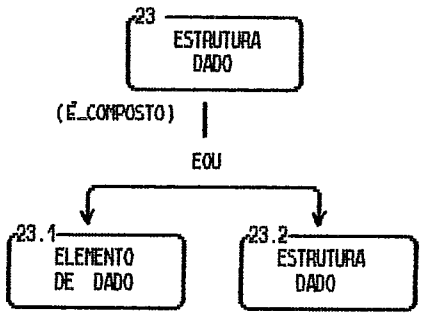


Figura V.23: Representação do elemento de conexão EOU.

REPRESENTAÇÃO DE RELACIONAMENTOS IGUAIS

A partir da disponibilidade de uso dos elementos de conexão E, OU e EOU na rede de conhecimento, sempre que um nó contiver mais de um relacionamento posterior do mesmo tipo, esses relacionamentos devem ser precedidos por um dos elementos de conexão, a fim de representar e especificar a condição de sequenciamento das relações. Neste caso, utilizar o próprio relacionamento para ligar o nó em referência com o elemento E, OU e EOU e, a partir desse, omitir o tipo de relacionamento para os seus nós posteriores. Esses elementos de conexão aparecem na rede simplesmente como "E", "OU" e "EOU", como se fossem um nó. A figura (V.24) ilustra esta situação.

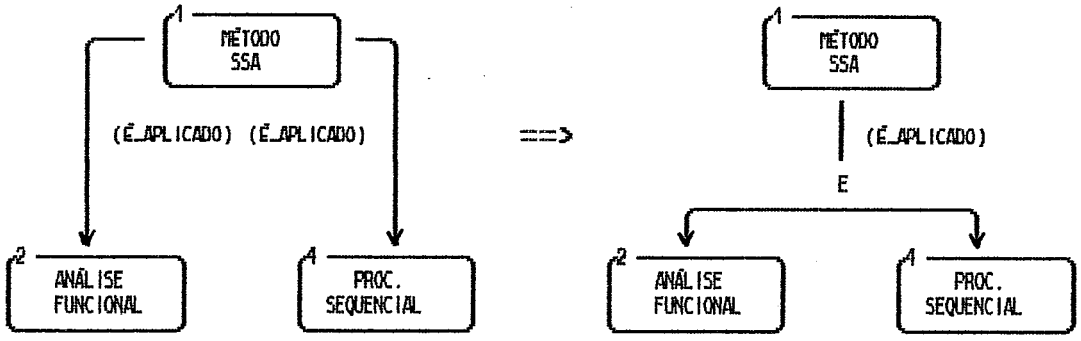


Figura V.24: Representação com mais de um relacionamento do mesmo tipo.

SEGMENTAÇÃO

Com o objetivo de permitir uma otimização da rede, facilitando o entendimento através de "partes" do conhecimento representado, é definida uma característica particular para os nós, denominada de "SEGMENTAÇÃO".

Um nó de segmento determina a existência de uma sub-rede, contida de nós de função "R" ou "Q", que omitida, apresenta uma rede somente com os nós considerados essenciais para o entendimento das características básicas do método representado. Este nó define, ainda, que todos os nós de sua sub-rede herdem suas características até o limite de sua segmentação.

O limite de uma sub-rede é até se encontrar um próximo nó de segmentação ou um nó terminal (isto é, quando não existe nó posterior). Quando esses limites não forem, suficientementes, adequados em alguma parte da rede de conhecimento para expressar o limite de uma sub-rede, a identificação numérica dos nós passa a tomar parte com uma característica a mais. Ou seja, a sub-rede de um nó de segmento deve ser numericamente identificada, utilizando-se como prefixo a identificação desse nó e, a partir dai, uma nova formação numérica.

Um nó de segmentação é representado por um "S" contido na parte superior direita do retângulo. A figura (V.25) ilustra este caso.

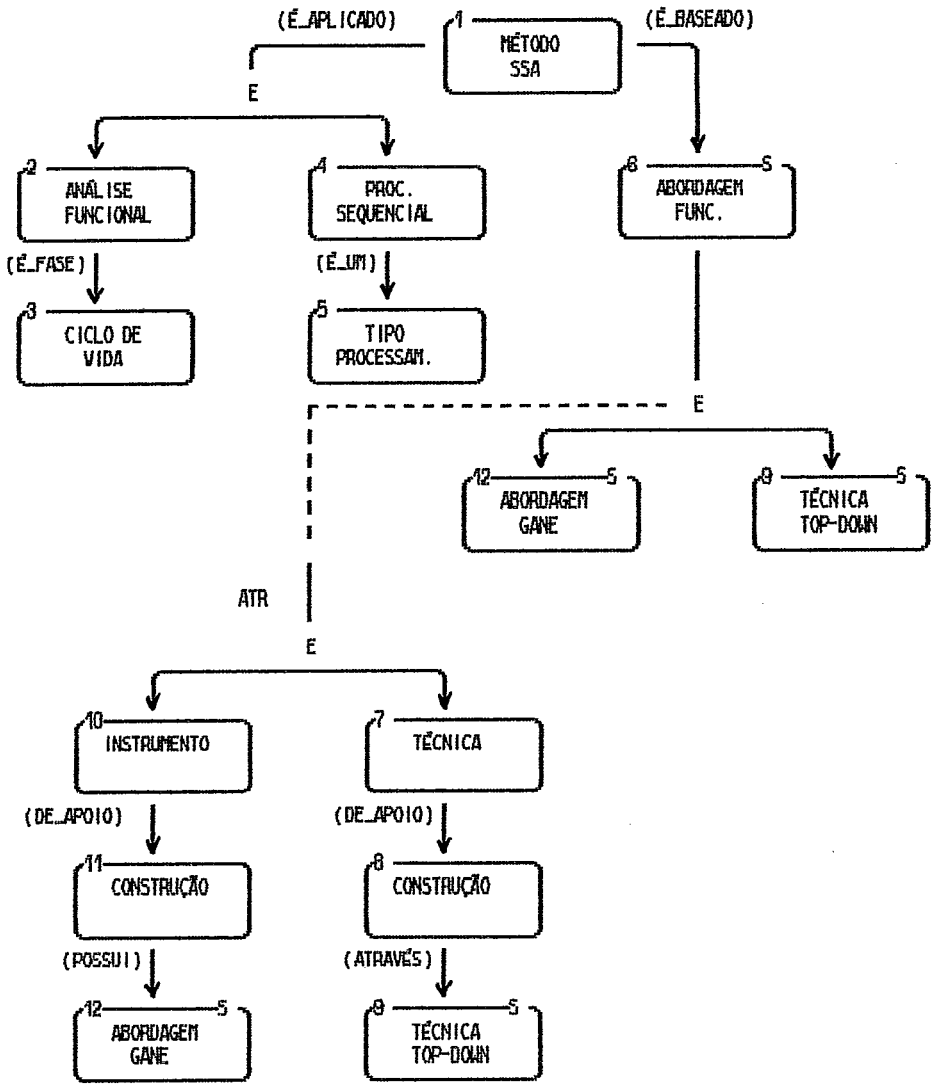


Figura V.25: Segmentação da rede de conhecimento.

É importante notar que estamos considerando, somente, um nível de abstração na rede de conhecimento.

Uma outra questão a observar é que na representação de uma rede com omissão de sub-redes de nós de segmentação, o relacionamento entre os nós envolvendo nós de segmento deve ser omitido.

IMAGEM MODULAR

Com a finalidade de permitir uma visualização completa da rede, no menor espaço físico possível, definimos um recurso que denominamos de IMAGEM MODULAR. Com este recurso tem-se uma visão organizacional, completa, do conhecimento representado na rede de conhecimento.

Para a construção de uma imagem modular, deve-se substituir a representação dos nós, de retângulos arredondados por S para os nós segmentos e, por N, os demais nós. Em seguida, a cada S e N que representam os nós da rede, identificá-los com a sua respectiva identificação numérica. Com relação aos relacionamentos, eles devem ser omitidos neste caso, tendo em vista o objetivo modular. Assim, a rede representada na figura (V.25), tem sua imagem modular conforme mostrado na figura (V.26).

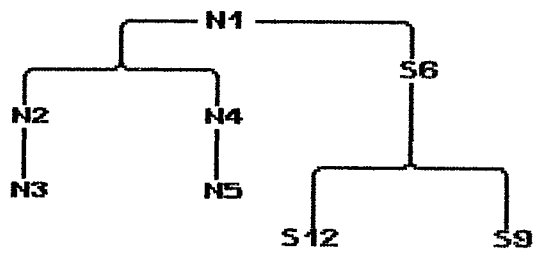


Figura V.26: Representação de uma imagem modular.

CONTINUIDADE DA REDE DE CONHECIMENTO

Por fim, um outro recurso que objetiva permitir uma continuidade na representação visual de uma rede, identificando na ponta de um relacionamento com a identificação numérica do nó que será continuado em outra parte da apresentação. A figura (V.27) ilustra este recurso, simulando em (a) uma parte da rede e em (b) sua continuação.

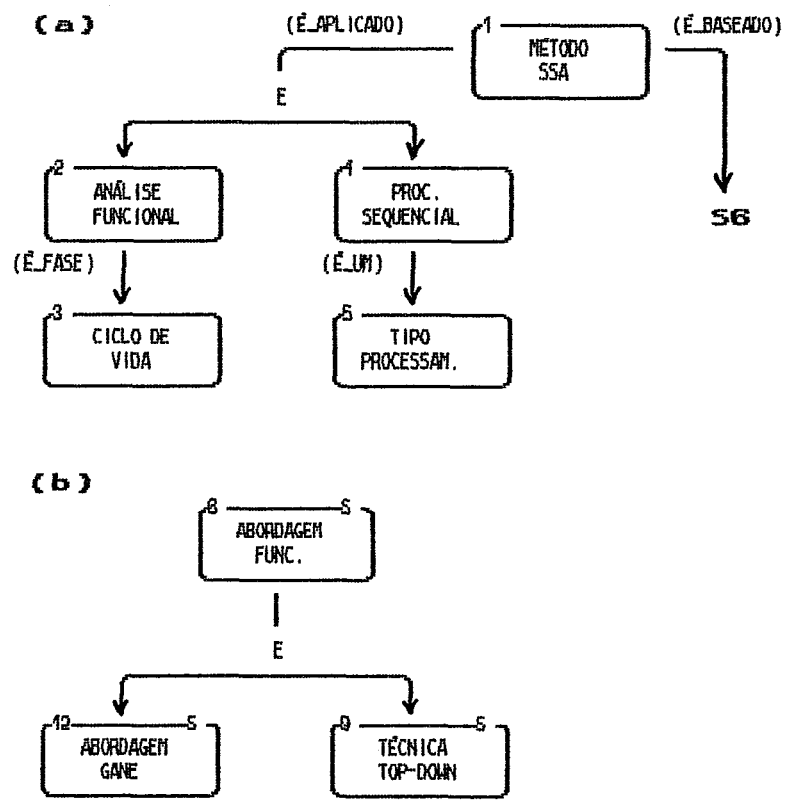


Figura V.27: Representação de continuidade na rede.

V.5.1.2 - RELACIONAMENTOS.

Uma vez estabelecida uma forma de apresentação visual do conhecimento (ou seja, através de nós de uma rede de conhecimento) é necessário um elemento de conexão (relacionamento) para associar os nós da rede de conhecimento com um significado adequado aos conhecimentos envolvidos nos nós, tendo em vista que o sistema que propomos utiliza os conceitos das redes semânticas.

Para se poder expressar adequadamente o conhecimento de um domínio, é necessário que o sistema de representação atenda, através de seus relacionamentos, a princípios organizacionais e de características essenciais relativos ao domínio tratado.

É importante observar que cada domínio reserva uma característica própria de seus conhecimentos, tais como os exemplos apresentados na seção (III.4.2), onde os relacionamentos necessários à representação são em função de tal característica.

Alguns dos princípios organizacionais que determinam um padrão para os conhecimentos, já fazem parte dos modelos semânticos para projetos de bases de dados. São eles:

- **Generalização**: organização de entidades dentro de uma hierarquia, com herança de atributos. Por exemplo, a entidade PESSOA é uma generalização de PROFESSOR e, portanto, herda todos os atributos daquela;

. **Classificação**: organização de entidades, relacionamentos ou atributos em classes, que enfatizam suas propriedades comuns e relevam as particularidades. Por exemplo, a classe PROFESSOR engloba os membros das classes PROFESSOR ADJUNTO, PROFESSOR HORISTA, etc;

. **Agregação**: Organização de entidades, relacionamentos ou atributos como partes de uma nova classe ou grupamento. Por exemplo, CARTEIRAS DE ALUNOS, MESA DE PROFESSOR e QUADRO NEGRO são componentes de uma sala de aula.

No entanto, novos princípios organizacionais são necessários para a representação, embora levando-se em consideração um domínio apropriado. São eles, segundo (MYLOPOULOS-88):

. **Projeção**: é o relacionamento semântico entre um objeto físico e sua projeção em algum plano, ou entre um evento (som) e sua representação através de algum meio (gráfico de frequência), ou ainda entre estruturas de dados e o conjunto de operações associado e a respectiva abstração que representa.

. **Exceções**: é a capacidade de expressar exceções a regras, exceções a exceções e assim por diante. Este princípio é muito importante já que as exceções são partes do conhecimento.

. **Reflexão**: um sistema de representação do conhecimento deve permitir a expressão de

conhecimento sobre a base de conhecimento e sobre o próprio processo de raciocínio. A partir dele, o sistema pode refletir sobre seu próprio estado, seus objetivos, seus sucessos, etc. É a partir do conhecimento reflexivo que se pode medir o nível de conhecimento de uma base.

. **Relatividade**: em um sistema de representação do conhecimento deve-se poder expressar diversos pontos de vista sobre um mesmo conhecimento, ou seja, deve-se ter um mecanismo que permita o armazenamento de conhecimento subjetivo, e não só o objetivo. Por exemplo, uma regra tal como "Se estiver chovendo e for dia útil, então eu não vou de carro ao centro da cidade." é uma regra subjetiva e não vale para todos os indivíduos.

Tendo em vista que o nosso sistema também contempla as características dos quadros, deve-se observar que as linguagens baseadas neles provêm uma representação estruturada de objetos ou classes de objetos e já incorporam alguns princípios organizacionais necessários, como os de generalização, classificação, agregação, projeção, além de possibilitar a associação de comportamento a objetos do domínio.

Como estamos tratando com a representação do conhecimento de métodos de desenvolvimento de software, é fundamental que hajam as relações necessárias e suficientes para representar os conhecimentos envolvidos nesse domínio.

Os relacionamentos precisam ser escolhidos e definidos de forma que permita um entendimento único, que os objetos por eles relacionados sejam identificados e, por fim, possam auxiliar na semântica do sistema de representação.

Sendo assim, concluímos que para encontrarmos as relações adequadas para expressar os métodos de desenvolvimento em redes de conhecimento, o melhor caminho seria através de uma análise num conjunto de métodos que revelasse, pelo menos, um padrão necessário de relacionamentos.

Para essa análise, escolhemos os seguintes métodos de desenvolvimento de software, nos quais identificamos as relações necessárias para atingir nosso objetivo. São eles:

- SSA - "Structured Systems Analysis" (GANE-83) e (DeMARCO-78);
- SADT - "Structured Analysis and Design Technique" (DICKOVER-78) e (ROSS-85);
- SREM - "Software Requirements Engineering Methodology" (ALFORD-80) e (BELL-77);
- DARTS - "Design Approach for Real-Time Systems" (GOMAA-84) e (GOMAA-89);
- VDM - "Vienna Development Method" (COHEN-86) e (MENDES-89).

Assim, os relacionamentos que identificamos como necessários para compor a rede de conhecimento, estão relacionados abaixo. Para facilitar o seu manuseio e dispô-los na rede de conhecimento, estabelecemos uma abreviação para cada um deles com, no máximo, três letras. São elas:

- **APO** ..... (APOIO)
- **ATR** ..... (ATRAVÉS)
- **DES** ..... (DESCREVE)
- **APL** ..... (APLICAÇÃO)
- **ATI** ..... (ATIVIDADE)
- **BAS** ..... (BASEADO)
- **COM** ..... (COMPOSTO)
- **FAS** ..... (FASE)
- **EUM** ..... (É_UM)
- **REP** ..... (REPRESENTA)
- **POS** ..... (POSSUI)

A seguir descrevemos cada um destes relacionamentos:

- **APO** (relacionamento "apoio")

Significa tudo o que serve de suporte ou auxílio. Expressa um apoio do nó-origem para o nó-destino. Por exemplo, os instrumentos disponíveis no método SSA são de

apoio à construção de software.

. ATR (relacionamento "através")

Significa uma realização de um para outro nó envolvido. Expressa que um nó-origem atinge sua meta através do nó-destino. Por exemplo: as etapas de construção de software, na abordagem (GANE-83), são através do estudo inicial, estudo detalhado, definição de alternativas, obtenção dos compromissos do usuário, aperfeiçoamento do projeto físico e da especificação das fases posteriores do projeto.

. **DES** (relacionamento "descreve")

Significa fazer uma exposição. Expressa que um nó-origem é narrado pelo nó-destino. Por exemplo, um processo é descrito no dicionário de dados pela identificação das entradas, saídas e da lógica dos processos.

. **APL** (relacionamento "aplicação")

Significa uma execução, cumprimento ou prática da utilização de um conhecimento. Expressa o emprego do nó-origem pelo nó-destino. Por exemplo, o método SSA é utilizado (ou empregado) para análise de software e para processamento do tipo sequencial.

. ATI (relacionamento "atividade")

Significa qualquer ação ou trabalho específico. Expressa que um nó-origem é atividade do nó-destino. Por

exemplo, a análise de software, no método SSA, é uma atividade da análise funcional.

. BAS (relacionamento "baseado")

Significa fundamentar ou abalizar um conhecimento. Expressa que um nó-origem é fundamentado pelo nó-destino. Por exemplo, o método SSA é baseado, fundamentado, por abordagem funcional.

. COM (relacionamento "composto")

Significa a formação ou construção de diferentes partes. Expressa que o nó-origem é constituido por dois ou mais conhecimentos indicados pelos nós-destino. Por exemplo, os instrumentos disponíveis no método SSA são compostos pelo diagrama de fluxo de dados, dicionários de dados, diagrama de acesso imediato e pela lógica de processos.

. FAS (relacionamento "fase")

Significa qualquer estágio (ou etapa) de uma evolução. Expressa que o nó-origem é uma etapa do conhecimento indicado no nó-destino. Por exemplo, a atividade de análise, no método SSA, é uma das fases de um ciclo de vida.

. EUM (relacionamento "é_um")

Significa uma identificação de características. Expressa que o nó-origem é identificado pela mesma característica do conhecimento representado no nó-destino.

Por exemplo, o processamento sequencial, ou seja uma característica do método SSA, é um tipo de processamento de dados.

. **REP** (relacionamento "representa")

Significa a imagem ou reprodução de um conhecimento. Expressa como um conhecimento é representado. Por exemplo, um retângulo arredondado, expresso pelo DFD do método SSA (GANE-83), representa um processo funcional.

. **POS** (relacionamento "possui")

Significa ter ou reter em seu poder um determinado conhecimento. Expressa que o nó-origem têm como propriedade o conhecimento representado pelo nó-destino. Por exemplo, a abordagem GANE do método SSA tem etapas de construção definidas para guiar a construção de software.

V.5.2 - ESTRUTURA DO CONHECIMENTO

A Estrutura de Conhecimento (EC) tem o objetivo de permitir a identificação e descrição de um conhecimento representado por um nó na rede de conhecimento. Sua função organizacional permite uma otimização da rede, onde características particulares do conhecimento são representadas por ela.

A estrutura do conhecimento está baseada nas características do método dos quadros. Portanto, as propriedades desse método de representação devem estar

disponíveis nas EC's, desde que sejam adequadas e necessárias para o método tratado.

De acordo com os tipos de conhecimentos definidos e ilustrado na figura (V.4), as EC's servem de armazenamento intermediário dos conhecimentos envolvidos nos métodos de desenvolvimento de software.

A estrutura do conhecimento é formada por seis estruturas com características distintas. Cada estrutura é composta por "campos" (ou seja, os atributos das estruturas dos quadros), que por sua vez representam as informações, num determinado momento, das características do conhecimento representado pelo nó.

As seis estruturas que definimos são:

(1) IDENTIFICAÇÃO

(2) DESCRIÇÃO

(3) CONSTRUÇÃO

(4) ESTATÍSTICA

(5) CLASSIFICAÇÃO

(6) ESPECIFICAÇÃO

Os nós da rede de conhecimento de função "R" (rede) devem ser descritos, obrigatoriamente, pelas estruturas EC(1), EC(2) e EC(3).

Os nós da rede de conhecimento de função "Q" (quadro) devem ser descritos, obrigatoriamente, pelas estruturas EC(1), EC(2), EC(3) e EC(6).

Quanto à estrutura EC(3), esta pode existir duas vezes para um mesmo nó, ou seja, uma referente aos nós anteriores e outra aos nós posteriores.

Quanto à estrutura EC(4), esta foi definida neste trabalho, para reunir dados quantitativos sobre as características da rede de conhecimento. Portanto, esta só poderá existir uma única vez e sendo para o nó raiz da rede (nó que identifica a rede), independente de ser de função "R" ou "Q".

Com o mesmo objetivo organizacional da estrutura EC(4), a estrutura EC(5) foi definida para reunir as informações dos nós da rede que classificam um método, instrumento ou técnica. Portanto, também sendo utilizada somente no nó raiz, ela pode existir até três vezes, conforme o tipo de classificação que está representando.

Quanto a estrutura EC(6), esta pode ser definida tantas vezes quantas necessárias forem para expressar as características desejadas num nó tipo "Q".

. EC(1): IDENTIFICAÇÃO

Esta estrutura de conhecimento tem o objetivo de conter os conhecimentos de identificação da rede e do domínio, para cada nó da rede de conhecimento.

A figura (V.28) apresenta esta estrutura e a descrição de seus campos.

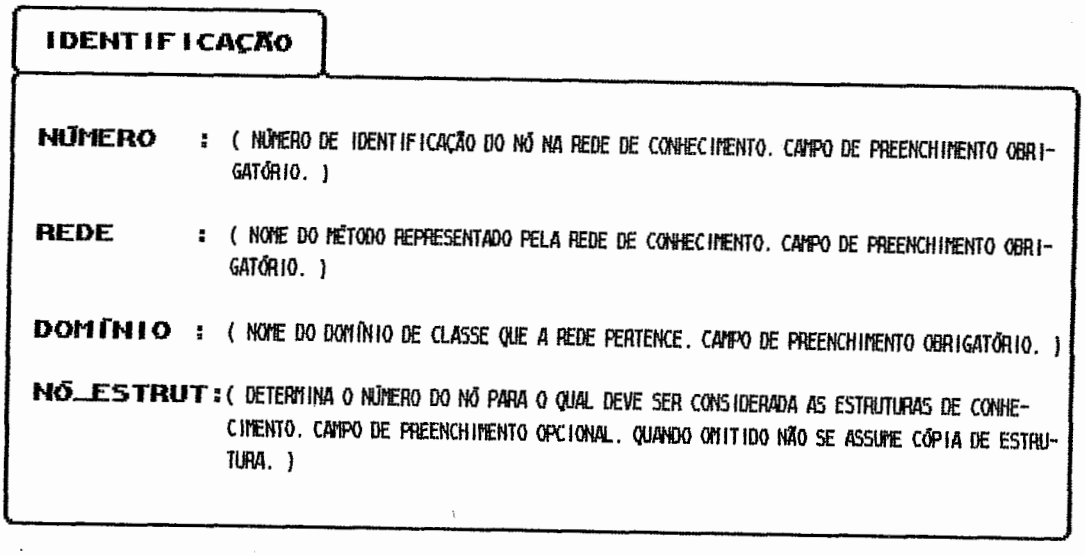


Figura V.28: Estrutura de conhecimento "identificação".

. EC(2): DESCRIÇÃO

Esta estrutura de conhecimento tem o objetivo de conter os conhecimentos necessários para o entendimento do nó e de suas características básicas.

A figura (V.29) apresenta esta estrutura e a descrição de seus campos.

DESCRIÇÃO

NOME	: (NOME DE IDENTIFICAÇÃO DO NÓ NA REDE DE CONHECIMENTO. CAMPO DE PREENCHIMENTO OBRIGATÓRIO.)
DESCRIÇÃO	: (DESCRIÇÃO GERAL E/OU OBJETIVO E/OU CARACTERÍSTICAS DO NÓ. CAMPO DE PREENCHIMENTO OBRIGATÓRIO.)
FUNÇÃO	: (DETERMINA A FUNÇÃO DO NÓ NA REDE, SE R=REDE OU Q=QUADRO. CAMPO DE PREENCHIMENTO OPCIONAL. QUANDO OMITIDO É ASSUMIDO O VALOR R=REDE.)
TIPO_NÓ	: (DETERMINA A CARACTERÍSTICA DO TIPO DE NÓ NA REDE, SE A=ATIVO OU F=FRACO. CAMPO DE PREENCHIMENTO OPCIONAL. QUANDO OMITIDO É ASSUMIDO O VALOR A=ATIVO)
SEGMENTO	: (DETERMINA A CARACTERÍSTICA DO NÓ NA REDE, SE É OU NÃO UM NÓ DE SEGMENTAÇÃO, S=SIM E N=NAO. CAMPO DE PREENCHIMENTO OPCIONAL. QUANDO OMITIDO É ASSUMIDO O VALOR N=NÃO.)
CLASF	: (DETERMINA A CARACTERÍSTICA DO NÓ NA REDE, SE REPRESENTA UM CONHECIMENTO CLASSIFICATÓRIO M=MÉTODO, I=INSTRUMENTO, T=TÉCNICA OU N=NENHUM, REFERENTE AO DOMÍNIO DE CLASSE. CAMPO DE PREENCHIMENTO OPCIONAL. QUANDO OMITIDO É ASSUMIDO N=NENHUM.)

Figura V.29: Estrutura de conhecimento "descrição".

. EC(3): CONSTRUÇÃO

Esta estrutura do conhecimento tem o objetivo de conter as informações relativas a construção da rede de conhecimento. O princípio de construção é de, para cada nó, se ter o conhecimento relativo às construções imediatamentes anterior e posterior ao nó. Portanto, poderão existir duas estruturas desse tipo: uma se referindo a construção anterior e outra para a posterior.

A figura (V.30) apresenta esta estrutura e a descrição de seus campos.

CONSTRUÇÃO

TIPO_CONST	:	(DETERMINA SE A CONSTRUÇÃO DO NÓ É RELATIVA AS CONEXÕES A=ANTERIOR OU P=POSTERIOR. CAMPO DE PREENCHIMENTO OBRIGATÓRIO.)
QUANT_CONST	:	(DETERMINA A QUANTIDADE DE CONEXÕES PARA O NÓ EM REFERÊNCIA. CAMPO DE PREENCHIMENTO OBRIGATÓRIO COM VALOR)0.)
NÓS_CONST	:	(IDENTIFICA OS NÓS ANTERIOR OU POSTERIOR AO NÓ EM REFERÊNCIA. CAMPO DE PREENCHIMENTO OBRIGATÓRIO.)
RELAC_CONST	:	(IDENTIFICA OS RELACIONAMENTOS ANTERIOR OU POSTERIOR AO NÓ EM REFERÊNCIA. CAMPO DE PREENCHIMENTO OBRIGATÓRIO.)
SENT_CONST	:	(IDENTIFICA OS SENTIDOS DE DIREÇÃO DOS RELACIONAMENTOS EM RELAÇÃO AO NÓ EM REFERÊNCIA. IDENTIFICANDO COM O=PARA OS NÓS ORIGEM E D=PARA OS NÓS DESTINO. CAMPO DE PREENCHIMENTO OBRIGATÓRIO.)

Figura V.30: Estrutura de conhecimento "construção"

. EC(4): ESTATÍSTICA

Esta estrutura do conhecimento tem o objetivo de conter conhecimentos quantitativos resultantes de toda a rede.

A figura (V.31) apresenta esta estrutura e a descrição de seus campos.

ESTATÍSTICA

COMEN_ESTAT	: (DESCREVE UM OBJETIVO E/OU CARACTERÍSTICA PARA A ESTATÍSTICA. CAMPO DE PREENCHIMENTO OBRIGATÓRIO.)
NÓS_TOTAL	: (DETERMINA A QUANTIDADE TOTAL DE NÓS EXISTENTES NA REDE DE CONHECIMENTO. CAMPO DE PREENCHIMENTO OBRIGATÓRIO, COM VALOR >0.)
NÓS_ATIVO	: (DETERMINA A QUANTIDADE DE NÓS DO TIPO ATIVO NA REDE. CAMPO DE PREENCHIMENTO OBRIGATÓRIO, COM VALOR >0.)
NÓS_FRACO	: (DETERMINA A QUANTIDADE DE NÓS DO TIPO FRACO NA REDE. CAMPO DE PREENCHIMENTO OPCIONAL. SE OMITIDO É ASSUMIDO O VALOR 0.)
NÓS_REDE	: (DETERMINA A QUANTIDADE DE NÓS DE FUNÇÃO REDE. CAMPO DE PREENCHIMENTO OPCIONAL. SE OMITIDO É ASSUMIDO O VALOR 0.)
NÓS_QUADRO	: (DETERMINA A QUANTIDADE DE NÓS DE FUNÇÃO QUADRO. CAMPO DE PREENCHIMENTO OPCIONAL. SE OMITIDO É ASSUMIDO O VALOR 0.)
NÓS-SEGMENTO	: (DETERMINA A QUANTIDADE DE NÓS DE SEGMENTAÇÃO EXISTENTE NA REDE. CAMPO DE PREENCHIMENTO OPCIONAL. QUANDO OMITIDO É ASSUMIDO O VALOR 0.)
NÓS_CLASF	: (DETERMINA A QUANTIDADE DE NÓS DE CLASSIFICAÇÃO EXISTENTE NA REDE. CAMPO DE PREENCHIMENTO OPCIONAL. QUANDO OMITIDO É ASSUMIDO O VALOR 0.)

Figura V.31: Estrutura de conhecimento "estatística"

. EC(5): CLASSIFICAÇÃO

A estrutura de conhecimento de classificação identifica os nós da rede que tem conhecimento que se referem à classes de métodos, instrumentos e técnicas.

A figura (V.32) apresenta esta estrutura e a descrição de seus campos.

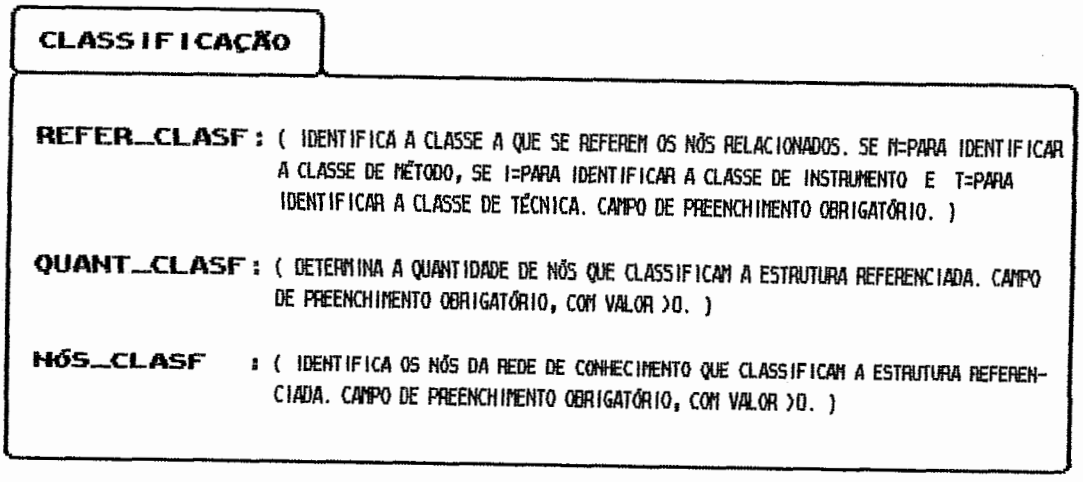


Figura V.32: Estrutura de conhecimento "classificação"

.EC(6): ESPECIFICAÇÃO

Esta estrutura de conhecimento tem o objetivo de expressar os conhecimentos relativos à: descrição de procedimentos, regras e heurísticas.

Para a descrição de procedimentos, identificam-se os conhecimentos para, por exemplo, estabelecer as etapas de construção de software quando se utiliza a abordagem GANE.

Para a descrição de regras, identificam-se os conhecimentos para se estabelecer em critérios para a tomada de decisões a partir de fatos relativos ao método representado pela rede de conhecimento.

Para a descrição de heurísticas, identificam-se os conhecimentos para se estabelecer em heurísticas que traduzem o conhecimento a partir de um especialista do método representado e que, ainda, traduz uma necessidade desejada pelo especialista e que não significa uma essência do método.

A figura (V.33) apresenta esta estrutura e a descrição de seus campos.

ESPECIFICAÇÃO

IDENT_ESPEC	: (DETERMINA O NOME DE IDENTIFICAÇÃO DA ESTRUTURA DE PROCEDIMENTOS. CAMPO DE PREENCHIMENTO OBRIGATÓRIO.)
COMEN_ESPEC	: (DESCREVE O OBJETIVO E/OU CARACTERÍSTICA DA ESTRUTURA DE PROCEDIMENTOS. CAMPO DE PREENCHIMENTO OBRIGATÓRIO.)
TIPO_ESPEC	: (DETERMINA O TIPO DE PROCEDIMENTO, SE: P=PARA UM PROCEDIMENTO GENÉRICO, R=PARA DETERMINAR UM PROCEDIMENTO DE REGRAS PERTINENTE AO MÉTODO OU H=PARA DETERMINAR A ESPECIFICAÇÃO DE REGRAS TRADUZIDAS POR UM ESPECIALISTA.)
QUANT_FATOS	: (DETERMINA A QUANTIDADE DE FATOS ESPECIFICADOS PARA A ESTRUTURA. CAMPO DE PREENCHIMENTO OPCIONAL. QUANDO OMITIDO É ASSUMIDO O VALOR 0.)
FATO<SEQUÊNCIA>	: (ESPECIFICA OS FATOS DE INTERESSE PARA A ESTRUTURA. CAMPO DE PREENCHIMENTO OPCIONAL.)
QUANT_AÇÕES	: (DETERMINA A QUANTIDADE DE AÇÕES DISPONÍVEIS NA ESTRUTURA. CAMPO DE PREENCHIMENTO OPCIONAL. QUANDO OMITIDO É ASSUMIDO O VALOR 0.)
AÇÃO<SEQUÊNCIA>	: (ESPECIFICA AS AÇÕES DE INTERESSE PARA A ESTRUTURA. CAMPO DE PREENCHIMENTO OPCIONAL.)
QUANT_REGRAS	: (DETERMINA A QUANTIDADE DE REGRAS PARA ATUAREM SOBRE OS FATOS E AÇÕES DISPONÍVEIS NA ESTRUTURA. CAMPO DE PREENCHIMENTO OBRIGATÓRIO.)
REGRA<SEQUÊNCIA>	: (ESPECIFICA AS REGRAS, EM PARES CONDIÇÃO-AÇÃO, RELATIVOS AOS FATOS E AÇÕES DISPONÍVEIS NA ESTRUTURA.)
PROX_ESPEC	: (ESPECIFICA A IDENTIFICAÇÃO DE UMA PRÓXIMA ESTRUTURA DE ESPECIFICAÇÃO. CAMPO DE PREENCHIMENTO OPCIONAL.)

Figura V.33: Estrutura de conhecimento "especificação".

V.5.3 - MANIPULAÇÃO DO CONHECIMENTO.

Conforme análises realizadas nas seções (V.1) e (V.2), uma linguagem de manipulação tem a função básica de permitir toda a operacionalidade sobre os demais componentes definidos para o sistema.

Porém, tendo em vista que a implementação deste sistema não está considerada como escopo deste trabalho, nos reteremos à especificação das funções que consideramos fundamentais para o contexto do sistema, de forma que possamos expressar suas características como base para uma implementação futura.

É importante, no entanto, que sejam apresentadas e discutidas as situações que podem ocorrer para a representação de um conhecimento como, ainda, o raciocínio com o conhecimento representado. A partir daí, analisar situações com relação ao nível de conhecimento.

V.5.3.1 - ANÁLISE DE SITUAÇÕES

Um sistema de representação de conhecimento tem vantagens consideráveis para a definição e reconhecimento de informações, os quais se constituem como forma de conhecimento, num domínio em particular.

Este processo consiste da descoberta de que essas informações podem ser, ou são projetadas para serem, manipuladas e interpretadas de forma que as características de seu domínio expressem sobre "quem" se está tratando.

Há dois meios básicos através dos quais o conhecimento é identificado e representado: através dos tipos de conhecimentos que caracterizam o método de desenvolvimento e, a partir daí, através dos componentes de representação que compõem o sistema (ou sejam, a rede de conhecimento e as estruturas de conhecimento).

A partir dessas informações envolvidas na representação, uma questão inicial a ser abordada é o estabelecimento do nível de conhecimento a ser expresso pelos nós da rede e pelas suas estruturas de conhecimento.

O conhecimento pode ser expresso de duas formas:

1) Quando um nó da rede expressa um conhecimento por sí só, ou seja, através do significado de seu nome.

Um nó deste tipo é definido no sistema como um nó de função "R" (rede) e, portanto, sua definição se dá com a seleção das estruturas de conhecimento EC(1)-IDENTIFICAÇÃO, EC(2)-DESCRIÇÃO e EC(3)-CONSTRUÇÃO.

2) Quando um nó da rede expressa seu conhecimento através de estruturas de conhecimento EC(6)-ESPECIFICAÇÃO.

Um nó deste tipo é definido no sistema como um nó de função "Q" (quadro) e, portanto, sua definição se dá com a seleção das estruturas de conhecimento básicas, ou sejam, EC(1)-IDENTIFICAÇÃO, EC(2)-DESCRIÇÃO, EC(3)-CONSTRUÇÃO e de tantas EC(6)-ESPECIFICAÇÃO quantas forem necessárias.

Vamos supor, como exemplo, que desejamos representar o conhecimento do seguinte texto:

"Na abordagem GANE, as etapas de construção para a análise estruturada de software são através de um estudo inicial, de um estudo detalhado, de uma definição de alternativas, de uma obtenção dos compromissos dos usuários, do aperfeiçoamento do projeto físico e das fases posteriores do projeto."

Este tipo de conhecimento pode ser representado, então, de duas formas, conforme a necessidade ou desejo do nível de conhecimento a ser expresso pelo sistema.

A primeira forma de se representar o conhecimento do texto sugerido, é de se expressar as etapas de construção na própria rede de conhecimento através de nós específicos para elas. Este tipo de representação busca tornar expressivo o entendimento de quais são as etapas já na rede de conhecimento. Uma rede de conhecimento para este caso, poderá ser conforme a figura (V.34).

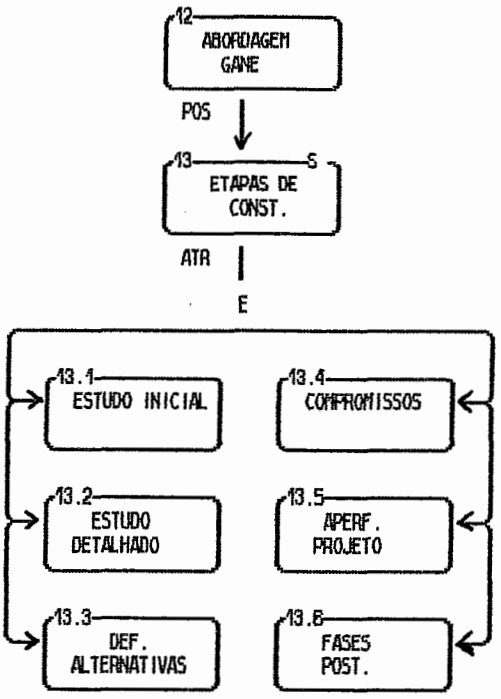


Figura V.34: Uma rede de conhecimento.

Neste caso, as estruturas de conhecimento para os nós N13-Etapas de const., N13.1-Estudo inicial, N13.2-Estudo detalhado, N13.3-Def. alternativas, N13.4-compromissos, N13.5-Aperf. projeto e N13.6-Fases post., são de função "R" (rede). Essas estruturas são definidas como mostra a figura (V.35).

IDENTIFICAÇÃO

NÚMERO	:	N13
REDE	:	MÉTODO SSA
DOMÍNIO	:	MÉTODOS DE DESENVOLVIMENTO DE SOFTWARE

DESCRIÇÃO

NOME	:	ETAPAS DE CONST.
DESCRIÇÃO	:	IDENTIFICA A EXISTÊNCIA DE ETAPAS PARA GUIAREM A CONSTRUÇÃO DE SOFTWARE.
FUNÇÃO	:	R
TIPO_NÓ	:	A
SEGMENTO	:	S
CLASF	:	N

CONSTRUÇÃO

TIPO_CONST	:	A
QUANT_CONST	:	1
NÓS_CONST	:	N12
RELAC_CONST	:	POS
SENT_CONST	:	D

CONSTRUÇÃO

TIPO_CONST	:	P
QUANT_CONST	:	6
NÓS_CONST	:	E (N13.1, N13.2, N13.3, N13.4, N13.5, N13.6)
RELAC_CONST	:	ATR
SENT_CONST	:	0

IDENTIFICAÇÃO

NÚMERO	:	N13.1
REDE	:	MÉTODO SSA
DOMÍNIO	:	MÉTODOS DE DESENVOLVIMENTO DE SOFTWARE

DESCRIÇÃO

NOME	:	ESTUDO INICIAL
DESCRIÇÃO	:	O ESTUDO INICIAL DEVE SER RÁPIDO E BARATO, E AVALIAR A SOLICITAÇÃO.
FUNÇÃO	:	R
TIPO_NÓ	:	A
SEGMENTO	:	N
CLASF	:	N

CONSTRUÇÃO

TIPO_CONST	:	A
QUANT_CONST	:	1
NÓS_CONST	:	N13
RELAC_CONST	:	ATR
SENT_CONST	:	D

IDENTIFICAÇÃO

NÚMERO	:	N13.2
REDE	:	MÉTODO SSA
DOMÍNIO	:	MÉTODOS DE DESENVOLVIMENTO DE SOFTWARE

DESCRIÇÃO

NOME	:	ESTUDO DETALHADO
DESCRIÇÃO	:	O ESTUDO DETALHADO BASEIA-SE NO ESTUDO INICIAL, PARA UMA DOCUMENTAÇÃO MAIS REFINADA.
FUNÇÃO	:	R
TIPO_NÓ	:	A
SEGMENTO	:	N
CLASF	:	N

CONSTRUÇÃO

TIPO_CONST	:	A
QUANT_CONST	:	1
NÓS_CONST	:	N13
RELAC_CONST	:	ATR
SENT_CONST	:	D

IDENTIFICAÇÃO

NÚMERO	:	N13.3
REDE	:	MÉTODO SSA
DOMÍNIO	:	MÉTODOS DE DESENVOLVIMENTO DE SOFTWARE

DESCRIÇÃO

NOME	:	DEF. ALTERNATIVAS
DESCRIÇÃO	:	A DEFINIÇÃO DE ALTERNATIVAS PERMITE TOMAR DECISÕES SOBRE O NOVO SISTEMA.
FUNÇÃO	:	R
TIPO_NÓ	:	A
SEGMENTO	:	N
CLASF	:	N

CONSTRUÇÃO

TIPO_CONST	:	A
QUANT_CONST	:	1
NÓS_CONST	:	N13
RELAC_CONST	:	ATR
SENT_CONST	:	D

IDENTIFICAÇÃO

NÚMERO	:	N13.4
REDE	:	MÉTODO SSA
DOMÍNIO	:	MÉTODOS DE DESENVOLVIMENTO DE SOFTWARE

DESCRIÇÃO

NOME	:	COMPROMISSOS
DESCRIÇÃO	:	ESTA ETAPA BUSCA OBTER O COMPROMISSO DOS USUÁRIOS TOMADORES DE DECISÃO.
FUNÇÃO	:	R
TIPO_NÓ	:	A
SEGMENTO	:	N
CLASF	:	N

CONSTRUÇÃO

TIPO_CONST	:	A
QUANT_CONST	:	1
NÓS_CONST	:	N13
RELAC_CONST	:	ATR
SENT_CONST	:	D

IDENTIFICAÇÃO

NÚMERO	:	N13.5
REDE	:	MÉTODO SSA
DOMÍNIO	:	MÉTODOS DE DESENVOLVIMENTO DE SOFTWARE

DESCRIÇÃO

NOME	:	APERF. PROJETO
DESCRIÇÃO	:	O APERFEIÇOAMENTO DO PROJETO BUSCA A OBTENÇÃO DE UM PROJETO FÍSICO MAIS SEGURO.
FUNÇÃO	:	R
TIPO_NÓ	:	A
SEGMENTO	:	N
CLASF	:	N

CONSTRUÇÃO

TIPO_CONST	:	A
QUANT_CONST	:	1
NÓS_CONST	:	N13
RELAC_CONST	:	ATR
SENT_CONST	:	D

IDENTIFICAÇÃO

NÚMERO	:	N13.6
REDE	:	MÉTODO SSA
DOMÍNIO	:	MÉTODOS DE DESENVOLVIMENTO DE SOFTWARE

DESCRIÇÃO

NOME	:	FASES POST.
DESCRIÇÃO	:	ESTABELECE AS FASES POSTERIORES DO DESENVOLVIMENTO DO PROJETO.
FUNÇÃO	:	R
TIPO_NÓ	:	A
SEGMENTO	:	N
CLASF	:	N

CONSTRUÇÃO

TIPO_CONST	:	A
QUANT_CONST	:	1
NÓS_CONST	:	N13
RELAC_CONST	:	ATR
SENT_CONST	:	D

Figura V.35: Exemplo de estruturas de conhecimento.

A segunda forma de se representar o conhecimento do texto sugerido é de se expressar na rede de conhecimento somente a existência das etapas de construção e, através de estruturas de conhecimento EC(6)-ESPECIFICAÇÃO, identificar quais são essas etapas. Uma rede de conhecimento para este caso pode ser, como ilustrada pela figura (V.36).

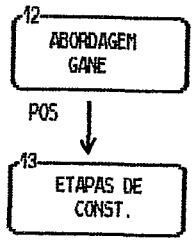


Figura V.36: Uma rede de conhecimento.

Neste caso, os nós N12 e N13 podem ser estruturados conforme ilustrado pela figura (V.37).

IDENTIFICAÇÃO

NÚMERO	:	N12
REDE	:	MÉTODO SSA
DOMÍNIO	:	MÉTODOS DE DESENVOLVIMENTO DE SOFTWARE

DESCRIÇÃO

NOME	:	ABORDAGEM GANE
DESCRIÇÃO	:	IDENTIFICA UM TIPO DE ABORDAGEM DE CARACTERÍSTICA PARTICULAR.
FUNÇÃO	:	R
TIPO_NÓ	:	A
SEGMENTO	:	N
CLASF	:	N

CONSTRUÇÃO

TIPO_CONST	:	P
QUANT_CONST	:	1
NÓS_CONST	:	N13
RELAC_CONST	:	POS
SENT_CONST	:	0

IDENTIFICAÇÃO

NÚMERO	:	N13
REDE	:	MÉTODO SSA
DOMÍNIO	:	MÉTODOS DE DESENVOLVIMENTO DE SOFTWARE

DESCRIÇÃO

NOME	:	ETAPAS DE CONST.
DESCRIÇÃO	:	IDENTIFICA A EXISTÊNCIA DE ETAPAS PARA GUIAREM A CONSTRUÇÃO DE SOFTWARE.
FUNÇÃO	:	Q
TIPO_NÓ	:	A
SEGMENTO	:	N
CLASF	:	N

CONSTRUÇÃO

TIPO_CONST	:	A
QUANT_CONST	:	1
NÓS_CONST	:	N12
RELAC_CONST	:	POS
SENT_CONST	:	D

ESPECIFICAÇÃO

IDENT_ESPEC	:	CONST_SOFT
COMEN_ESPEC	:	DETERMINA AS ETAPAS PARA A COSNTRUÇÃO DE SOFTWARE.
TIPO_ESPEC	:	P
QUANT_FATOS	:	6
FATO1	:	ESTUDO INICIAL
FATO2	:	ESTUDO DETALHADO
FATO3	:	DEFINIÇÃO DE ALTERNATIVAS
FATO4	:	COMPROMISSOS DOS USUÁRIOS
FATO5	:	APERFEIÇOAMENTO DO PROJETO FÍSICO
FATO6	:	FASES POSTERIORES DO PROJETO
QUANT_AÇÕES	:	0
QUANT_REGRAS	:	0

Figura V.37: Exemplo de estruturas de conhecimento.

A decisão da escolha de uma ou outra forma, ou seja, escolher o nível da representação, não é uma tarefa trivial e nem deve ser aleatória. Deve-se buscar entender quais conhecimentos são os essenciais para que uma rede expresse o entendimento desejado. Deve-se observar, ainda, que a decisão de se colocar um conhecimento em rede ou através de estruturas de conhecimento, além do entendimento visual, interfere na capacidade inferencial em se acessar esse conhecimento, quer seja por consultas diretas ou através de pesquisas interativas.

V.5.3.2 - LINGUAGEM DE MANIPULAÇÃO DO CONHECIMENTO

A linguagem de manipulação do conhecimento (LMC) busca apresentar como os componentes de representação do sistema podem ser utilizados e como seus conhecimentos podem ser acessados.

Conforme nossa estratégia para concepção do sistema de representação, figura (V.14), o objeto central é representado pelos nós da rede de conhecimento. A partir deste cabe à estrutura de conhecimento o armazenamento intermediário de todo o conhecimento, através da qual a LMC deve permitir sua criação, atualização e manipulação do conhecimento alí contidos.

Uma linguagem pode ser representada através de metalinguagens tais como a notação de BACKUS-NAUR FORM (BNF), a notação de WIRTH ou alguma outra forma de expressão de gramática (LEE-74) (McKEEMAN-70).

Para expressar a LMC, adotaremos a notação BNF.

A BNF é uma metalinguagem bastante popular de expressão da sintaxe de linguagens de programação. Trata-se de uma notação recursiva de formalização da sintaxe de linguagens através de produções gramaticais, permitindo assim a criação de dispositivos de geração de sentenças. Para tanto, cada produção corresponde a uma regra de substituição, em que a um símbolo da metalinguagem são associadas uma ou mais cadeias de símbolos, indicando as diversas possibilidades de substituição. Os símbolos em questão correspondem a não-terminais da gramática que está sendo especificada. As cadeias podem ser formadas de terminais e/ou não-terminais, e do símbolo &, que representa a cadeia vazia. A simbologia adotada é a seguinte:

⟨x⟩ - representa um não-terminal, cujo nome é dado por uma cadeia x de caracteres quaisquer. Os caracteres ⟨ e ⟩ são usados para delimitar o nome do não-terminal.

::= - é o símbolo da metalinguagem que associa a um não-terminal um conjunto de cadeias de terminais e/ou não-terminais, incluindo o símbolo da cadeia vazia. O não-terminal é escrito à esquerda deste símbolo, e as diversas cadeias, à sua direita. Lê-se "define-se como".

| - é o símbolo da metalinguagem que separa as diversas cadeias que constam à direita do

símbolo ::=. Lê-se "ou".

X - representa um terminal da linguagem que está sendo definida, e pertence ao conjunto de todos os átomos que compõem as sentenças da linguagem. Deve ser denotado tal como figura nas sentenças da linguagem, e não entre os caracteres ⟨ e ⟩, como ocorre no caso da denotação escolhida para os não-terminais.

& - representa a cadeia vazia.

yz - representa uma cadeia construída pela concatenação dos elementos y e z nesta ordem. Estes dois elementos podem, por sua vez, ser símbolos de terminais, de não-terminais, de cadeia vazia, ou mesmo outras cadeias.

A partir dessa notação, a gramática da LMC é:

==========

⟨representação⟩ ::= ⟨conhecimento⟩ ⟨acesso⟩

⟨conhecimento⟩ ::= ⟨rede⟩ ⟨quadro⟩

⟨rede⟩ ::= ⟨nó⟩ ⟨relacionamento⟩

⟨nó⟩ ::= ⟨ident_nó⟩

⟨ident_nó⟩ ::= N ⟨números⟩ |

N ⟨números⟩ . ⟨números⟩

⟨relacionamento⟩ ::= ⟨ident_relac⟩

```
ident_relac    ::= APO | ATR | DES | APL | ATI |
                   BAS | COM | FAS | EUM | REP | POS
quadro         ::= ⟨ident_ec⟩
ident_ec       ::= IDENT | DESCR | CONST |
                   ESTAT | CLASF | ESPEC
acesso         ::= Def ⟨tipo_acesso⟩
                       ⟨param_acesso⟩
                   End ⟨tipo_acesso⟩
tipo_acesso    ::= Cria | Atua | Reti | Cons
param_acesso   ::= ⟨selec_nó-acesso⟩ ; |
                   ⟨selec_nó_acesso⟩ ;
                   ⟨ident_ec⟩ :
                       ⟨expressão⟩ ;
                   End ⟨ident_ec⟩ : |
                   ⟨ident_ec⟩ :
                       ⟨expressão⟩ ;
                   End ⟨ident_ec⟩ : |
                   ⟨param_acesso⟩
selec_nó_acesso ::= Sel (⟨ident_nó⟩ | &);
expressao      ::= ⟨campos_ec⟩ = ⟨texto⟩ |
                   ⟨campos_ec⟩ = ? |
                   &
texto          ::= ⟨letras⟩ |
                   ⟨números⟩ |
                   ⟨símb_esp⟩ |
                   ⟨letras⟩ ⟨números⟩ |
                   ⟨números⟩ ⟨símb_esp⟩ |
                   ⟨letras⟩ ⟨números⟩ ⟨símb_esp⟩
```

```
letras          ::= a...z
números         ::= 0 ... 9999
símb_esp        ::= !|@|#|$|%|^|&|
                    *|(|)|_|+|=|-|{|}|[|]|:
                    "|'|^|;|?|/|>|.|<|,|~|'|`
=====================================================
```

Com relação a gramática apresentada da LMC, temos o não-terminal <campos_ec> que se refere aos campos das estruturas de conhecimento.

Agora, com o objetivo de demonstrar o uso da LMC especificada, vamos supor que desejamos criar uma base de conhecimento da rede representada pela figura (V.36), que expressa uma das formas de se representar o texto sugerido no ítem anterior.

Para este caso, uma LMC pode ser especificada da seguinte forma:

```
DefCria
  IDENT:
    número = n12;
    rede = Método SSA;
    domínio = Métodos de desenvolvimento de software;
  EndIDENT:
  DESCR:
    nome = Abordagem GANE;
```

descrição = Identifica um tipo de abordagem de caractersítica particular de análise estruturada de

```
software;
          função = r;
          tipo_nó = a;
          segmento = n;
          clasf = n;
        EndDESCR:
        CONST:
          tipo_const = p;
          quant_const = 1;
          nós_const = n13;
          relac_const = pos;
          sent_const = o;
        EndCONST:
      EndCria
      DefCria
        IDENT:
          número = n13;
          rede = Método SSA;
          domínio = Métodos de desenvolvimento de software;
        EndIDENT:
        DESCR:
          nome = Etapas de const;
          descrição = Identifica a existência de etapas para
guiarem a construção de software;
          função = q;
          tipo_nó = a;
          segmento = n;
          clasf = n;
        EndDESCR:
```

```
        CONST:
          tipo_const = a;
          quant_const = 1;
          nós_const = n12;
          relac_const = pos;
          sent_const = d;
        EndCONST:
        ESPEC:
          ident_espec = const_soft;
          comen_espec   =   Apresenta   as   etapas   para   a
construção de software;
          tipo_espec = p;
          quant_fatos = 6;
          fato1 = Estudo inicial;
          fato2 = Estudo detalhado;
          fato3 = Definição de alternativas;
          fato4 = Obtenção de compromissos;
          fato5 = Projeto físico;
          fato6 = Fases do projeto;
        EndESPEC:
      EndCria
```

Com esta base de conhecimento instalada torna-se possível o sistema acessá-la e responder a perguntas através de consultas.

Como ilustração, vamos supor que desejamos acessar a base de conhecimento para saber quais são as etapas de construção de software para a análise estruturada, segundo abordagem GANE. Uma consulta deste

tipo é especificada pela LMC da seguinte forma:

```
DefCons
  Sel (n13);
  ESPEC:
    quant_fatos= ?;
    fato = ?;
  EndESPEC:
EndCons
```

Neste caso, o sistema fornece o seguinte resultado:

```
. 6
. Estudo inicial
. Estudo detalhado
. Definição de alternativas
. Obtenção de compromissos
. Projeto físico
. Fases do projeto
```

Embora não tratando da implementação do sistema neste trabalho, conforme já temos colocado, é importante que comentemos sobre a seguinte questão. Na ilustração anterior, exemplificamos um acesso à base de conhecimento onde já sabiamos exatamente onde se encontrava a informação de interesse. Neste caso, poderemos chamar de uma consulta direta na base de conhecimento.

No entanto, isto pode não ocorrer sempre e, então, a base de conhecimento deve estar preparada e a

implementação do sistema deve prever o que chamamos de, consultas interativas através de um usuário do sistema.

Uma ilustração para este caso, para se obter as etapas de construção como mostrado anteriormente, primeiramente o usuário precisa saber se existe este conhecimento na base, onde se localiza e, por fim o que deseja, as etapas de construção.

Uma simulação para esta interação mostramos a seguir, onde S representa a interação do sistema e U a do usuário.

S) Sobre o quê deseja pesquisar ?

U) Abordagem Gane

S) Eu tenho essa informação como um nó da rede.
Voce quer que eu prossiga?

U) Sim

S) O que voce deseja saber ?

U) Quais os relacionamentos do nó

S) n12 - Abordagem Gane
Construção posterior:
nós: n13
relac.: pos - possui
sentido: o - origem

U) Qual o nome do nó n13 ?

S) n13 - Etapas de const.

U) Qual a descrição do nó ?

S) Identifica a existência de etapas para guiarem a construção de software.

U) Qual a função do nó ?

S) quadro

U) Quais as identificações de especificação do nó ?

S) const_soft

U) Qual o comentário da especificação do nó ?

S) Apresenta as etapas para a construção de software.

U) Qual o tipo de especificação do nó ?

S) procedimento

U) Mostre os fatos da especificação do nó ?

S) . Estudo inicial

. Estudo detalhado

. Definição de alternativas

. Obtenção de compromissos

. Projeto físico

. Fases do projeto

U) Encerrar a sessão

V.6 - CONCLUSÕES

Neste capítulo apresentamos as características básicas de um sistema para a representação do conhecimento, em particular, de métodos de desenvolvimento de software.

Toda uma organização foi definida para a elaboração de uma rede de conhecimento, como ainda, os tipos de relacionamentos e dos conhecimentos relativos ao domínio, foram definidos e descritos.

Situações foram simuladas para que o raciocínio com o conhecimento, o estabelecimento do nível de

conhecimento a ser utilizado como as características relevantes para a representação de um método, foram exercitadas e demonstradas.

Foi estruturada uma linguagem de manipulação para que as operações desejadas e necessárias fossem atendidas.

As análises e considerações iniciais do capítulo, foram de vital importância para a elaboração do sistema, tendo em vista a coerência e adequação do sistema resultante com os objetivos e características desejados.

Capítulo VI

CONCLUSÕES

O desenvolvimento deste trabalho proporcionou se abordar questões da representação do conhecimento e de métodos de desenvolvimento de software. Isto nos levou a considerar importante a identificação das características básicas das áreas da Inteligência Artificial e Engenharia de Software e, a partir daí, apresentar as características entre programas convencionais e de inteligência artificial.

Foram apresentadas definições e objetivos da representação do conhecimento, descritos os métodos mais tradicionais de representação do conhecimento, como ainda, sistemas de representação utilizando esses métodos.

Como resultado dessa experiência e da necessidade de se organizar as questões mais importantes que norteiam a tarefa de representar conhecimento, elaboramos um conjunto de fatores e sub-fatores para permitir guiar nossas análises sobre os métodos e sistemas de representação do conhecimento.

Para a elaboração do sistema de representação do conhecimento adequado para métodos de desenvolvimento de software, buscamos identificar as necessidades mais importantes e os tipos de conhecimentos que caracterizam o domínio dos métodos de desenvolvimento de software.

Análises foram realizadas para se adequar os tipos de conhecimentos identificados com os métodos de representação do conhecimento considerados mais convenientes, para identificar as características básicas de uma estrutura intermediária e de uma linguagem para manipulação do conhecimento. A partir daí, foi possível conceber o sistema e descrever seu modelo e componentes.

O ponto mais importante que consideramos, com a elaboração deste trabalho, é a contribuição deixada, como um primeiro passo, para tratar a representação do conhecimento de métodos de desenvolvimento de software de forma específica.

Com relação a contribuição deste trabalho para o projeto TABA, identificamos o resultado das características para a formação de uma base de conhecimentos sobre métodos de desenvolvimento de software. Portanto, se relaciona ao contexto do "meta ambiente" e do "ambiente gerador de ferramentas".

No contexto do "meta ambiente", o sistema de representação pode contribuir no processo de especificação do ADS, para que o usuário possa ter o método mais adequado conforme sua aplicação. No contexto do ambiente gerador de ferramentas, sua contribuição se dá por conter as características das ferramentas adequadas aos métodos de desenvolvimento.

Em termos de expectativas sobre a abrangência dos métodos de desenvolvimento serem representados pelo sistema

de representação, consideramos que a possibilidade de se lidar com conhecimentos que não sejam somente aqueles considerados essenciais do método de desenvolvimento é um recurso importante e disponível no sistema. No entanto, entendemos que somente com a utilização do sistema em diversos métodos de desenvolvimento, da representação de casos particulares e daqueles específicos de especialistas, que se poderá conhecer a extensão da capacidade do sistema.

A implementação deste sistema é o resultado mais imediato deixado por este trabalho. A partir daí, com a exercitação de diversas situações, certamente resultados e conclusões importantes serão obtidas.

Por fim, ressaltamos uma conclusão de William A. Woods:

"Enfatizar a força expressiva das representações através de estruturas de classificação taxonômica, é um passo no caminho de um novo estilo de programação."

REFERÊNCIAS BIBLIOGRÁFICAS

ALFORD, Mack W. (1977), " A Requirements Engineering Methodology for Real-Time Processing Requirements", IEEE Transactions on Software Engineering, Vol.SE-3, No.1, pp.60-69.

ALFORD, Mack W. (1980), "Software Requirements Engineering Methodoloy (SREM) at the Age of Four", Computer Software and Applications Conference, October, pp.866-874.

ALTY, J. L. e COOMBS, M. J., (1986), "Sistemas Expertos: Conceptos y Ejemplos", Diaz de Santos S.A..

ARTHUR, Lowel Jay (1985), "Measuring Programmer Productivity and Software Quality", Joan Wiley & Sons Inc..

BACK, Nelson (1983), "Metodologia de projeto de produtos indústriais", Ed. Guanabara Dois.

BELL, Thomas E., BIXLER, David C. e DYER, Margaret E., (1977), "An Extendable Approach to Computer-Aided Software Requirements Engineering", IEEE-Transactions on Software Engineering, Vol.SE-3, No.1, pp.49-59.

BERTALANFFY, L. Von (1973), "Teoria geral dos sistemas", Ed. Vozes.

BILLER, Horst e NEUHOLD, Erich J., (1988), "Semantics of Databases: The Semantics of Data Models", in Publication Data Readings in IA and Databases, pp.273-292.

BOEHM, B. (1976), "Software Engineering", Transactions on Computers.

BORGIDA, Alexander, GREENSPAN, Sol e MYLOPOULOS, John, (1985), "Knowledge Representation as the Basis for Requirements Specifications", IEEE-Computer, April, pp.82-90.

BRACHMAN, Ronald J. (1983), "What IS-A is and is not: An analysis of Taxonomic Links in Semantic Networks", IEEE-Computer, Vol.16, No.10, pp.30-36.

BRACHMAN, Ronald J., FIKES, Richard J. e LEVESQUE, Hector J., (1983a), "KRYPTON: A funtional approach to Knowledge Representation", IEEE- Computer, Vol.16, No.10, pp.67-73.

BRACHMAN, Ronald J. e SCHMOLZE, James G., (1988), "An Overview of the KL-One Knowledge Representation System", in Publication Data Readings in IA and Databases, pp.207-222.

BRACHMAN, Ronald J., GILBERT, Victoria P. e LEVESQUE, Hector J., (1988a), "An essential Hybrid reasoning System: Knowledge and Symbol level accounts of KRYPTON", in Publication Data Readings in IA and Databases, pp.293-300.

BRODIE, Michael L. e JARKE, Matthias, (1984), "On integrating Logic Programming and Databases", On Expert Database Systems.

BRODIE, Michael L. e MANOLA, Frank, (1988), "Database management: A survey", in Publication Data Readings in IA and Databases, pp.10-34.

BRODIE, Michael L. e RIDJANOVIC, Dzenan, (1988a), "On the Design and Specification of Database transactions", in Publication Data Readings in IA and Databases.

BRODIE, Michael L. (1988b), "Future Intelligent Information Systems: IA and Database Technologies Working To Gether", in Publication Data Readings in IA and Databases, pp.623-639.

BRYAN, William e SIEGEL, Stanley, (1984), "Making Software Visible Operational, and Maintainable in a Small Project Environment", IEEE Transaction on Software Engineering.

BUCHANAN e SHORTLIFE (edts), (1984), "Rule-based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project", Stanford University.

CAVALCANTE, Sueli M. A. (1988), "Uma ferramenta automatizada de apoio ao método DARTS", Tese de Mestrado, COPPE/UFRJ, Rio de Janeiro.

CHAN, Aryola (1988), "Storage and access structures to support a Semantic Data Model", in Publication Data Readings in IA and Databases.

CHEN, Peter Pin-Shan (1988), "The Entity-Relationship Model - Towards a Unified View of Data", in Publication Data Readings in IA and Databases.

CHANTLER, Alan (1984), "Técnicas e Práticas de Programação", Editora Campus.

CHORAFAS, Dimitris N. (1988), "Sistemas Especialistas", McGraw-Hill.

CLEMM, Geoffrey M. (1988), "The Workshop System - A practical Knowledge - Based Software Environment", ACM-Press, Sigsoft, Vol.13, No.5, November, pp.55-64.

COHEN, B., HARWOOD, W.T. e JACKSON, M.I., (1986), "The specification of complex Systems", Addison-Wesley Publishing Company.

CRISPIM, Emília M.H. (1988), "Definição de termos em Ambientes para o desenvolvimento de Software", Relatório técnico COPPE/UFRJ - ES175/88.

DAHL, Veronica (1983), "Logic Programming as a Representation of Knowledge", IEEE-Computer, Vol.16, No.10, pp.106-111.

DAVIS, Carl G. e VICK, Charles R., (1977), "The Software Development System", IEEE Transaction on Software Engineering, January.

DAVIS, Randall e SHROBE, Howard, (1983), "Representing Structure and behavior of Digital Hardware", IEEE-Computer, Vol.16, No.10, pp.75-82.

DEMARCO, Tom (1978), "Structured Analysis and System Specification", Yourdon Inc..

DEMARCO, Tom (1985), "Princípios e Conceitos em Engenharia de Software", Compucenter Sistemas.

DICKOVER, M. E., McGOWAN, C. L. e ROSS, Douglas T., (1978), "Software Design using SADT", Structured Analysis and Design, Vol.II, pp.101-114.

DOYLE, Jon (1983), "Admissible state Semantics for Representational Systems", IEEE-Computer, Vol.16, No.10, pp.119-123.

ELCOCK, E. W. (1983), "How complete are Knowledge Representation Systems? ", IEEE-Computer, Vol.16, No.10, pp.114-118.

FAIRLEY, Richard E. (1985), "Software Engineering Concepts", McGraw-Hill.

FERRARI, Domenico (1982), "Theory and Practice on Software Technology", International Seminars on Software Engineering, Capri/Italy.

FERREIRA, Aurélio B.H. (1975), "Novo Dicionário da Língua Portuguesa", Editora Nova Fronteira.

FIKES, Richard e KEHLER, Tom, (1985), "The role of Frame-based representation in reasoning", Communications of the ACM, Vol.28, No.9, pp.904-920.

FRANÇA, F.M.G. et al. (1989), "Redes neuronais artificiais: A volta do cérebro eletrônico?", ES-194/89, UFRJ.

FRENZEL, Louise Jr. (1987), "Crash Course in Artificial Inteliggence and Expert Systems", Albright Com..

FRISCH, Alan M. e ALLEN, James F., (1988), "Knowledge retrieval as limited inference", in Publication Data Readings in IA and Databases, pp.444-451.

FUNT, Brian V. (1983), "Analogical Modes of Reasoning and Process Modeling", IEEE-Computer, Vol.16, No.10, pp.99-104.

GALLAIRE, Herve, MINKER, Jack e NICOLAS, Jean-Marie, (1988), "Logic e Databases: A deductive approach", in Publication Data Readings in IA and Databases, pp.231-247.

GANE, Chris e SARSON, Trish, (1983), "Análise Estruturada de Sistemas", LTC-Livros Técnicos e Científicos.

GENARO, Sergio (1986), "Sistemas Especialistas: o conhecimento artificial", LTC-Livros Técnicos e Científicos.

GOMAA, H. (1984), "DARTS-A Software Design Method for Real-Time Systems", Communications of the ACM, Vol.27, No.9, pp.938-949.

GOMAA, H. (1989), "A Software Design Method for Distributed Real Time Applications", The Journal of Systems and Software, No.9, pp.81-94.

GUIMARÃES, Angelo de Moura (1985), "Introdução à Ciência da Computação", LTC-Livros Técnicos e Científicos.

HAUSEN, Hans-Ludwig e MULLERBURG, Monika, (1984), "Conspectus of Software Engineering Environments", Institute for Software Technologie.

HAVENS, William e MACKWORTH, (1983), "Representing Knowledge of the Visual World", IEEE-Computer, Vol.16, No.10, pp.90-96.

HU, S. David (1987), "Expert Systems for Software Engineers and Managers", Chapman and Hall.

ISRAEL, David J., BERANER, Bolt e NEWMAN, (1983), "The Role of Logic in Knowledge Representation", IEEE-Computer, Vol.16, No.10, pp.37-41.

KENT, William (1988), "Limitations of record-based information Models", in Publication Data Readings in IA and Databases, pp.85-97.

KELLER, Robert (1987), "Expert System Technology - Development & Application", Prentice Hall.

KELLER, Robert (1990), "Análise estruturada na prática", McGraw-Hill.

KISHIDA, Konichi (1988), "SDA: A Novel Approach to software environment design and construction", IEEE-Computer, pp. 69-78.

LAFUE, Gilles M. E. e SMITH, Reid G., (1988), "A Modular Tool Kit for Knowledge Management", in Publication Data Readings in IA and Databases.

LANGLEY, Pat (1983), "Representational Issues in Learning Systems", IEEE-Computer, Vol.16, No.10, pp.47-51.

LAUBER, Rudolf J. (1982), "Development Support Systems", IEEE-Transactions on Software Engineering.

LEE, John A. N. (1974), "The anatomy of a compiler", Computer science series.

LEITE, Júlio C.S. (1983), "SADT Datagrams, A Powerful Tool for Requirements Analysis", Anais do XVI Congresso Nacional de Informática, SUCESU, São Paulo.

LEVESQUE, Hector J. (1988), "Knowledge Representation and Reasoning", in Publication Data Readings in IA and Databases.

LEVESQUE, Hector J. (1988a), "The Logic of incomplete Knowledge Bases", in Publication Data Readings in IA and Databases, pp.328-341.

LEVINE, Robert I. (1988), "Inteligência Artificial e Sistemas Especialistas", McGraw-Hill.

LORIE, Raymond A. e HASKIN, Roger L., (1981), "On Extending the Functions of a Relational Database System", IBM Research Laboratory-Computer Science.

LUCAS, R. (1984), "An Expert System to Detect Burglars Using A Logic Language and A Relational Database", IEEE-Transactions on Software Engineering.

LUCENA, Carlos (1987), "Inteligência Artifial e Engenharia de Software", Jorge Zahar.

MANNUCCI, Stefano (1989), "GRASPIN: A structured development environment for analysis and design", IEEE - Software, November, pp.35-43.

MASURKAR, Vijay (1982), "Requirements for a Practical Software Engineering Environments", Wang Laboratories Inc..

McCALLA, Gordon e CERCONE, Nick, (1983), " Approaches to Knowledge Representation", IEEE-Computer, Vol.16, No.10, pp.12-16.

MCKEEMAN, William M., HORNING, James J. e WORTMAN, David B., (1970), "A compiler generator", Prentice-Hall Series in automatic computation.

MENDES, Sueli e AGUIAR, Teresa C., (1989), "Métodos para especificação de sistemas", Ed. Edgard Blucher Ltda.

MINSKY, M. (1975), "A Framework for representing Knowledge", in the Psychology of computer vision, New York-McGraW Hill.

MULDER, Jan A., MACKWORTH, Alan K. e HAVENS, William S., (1988), "Knowledge Structuring and Constraint Satisfaction: The Mapsee Approach", IEEE-Transactions on Pattern Analysis and Machine Intelligence.

MYLOPOULOS, John, SHIBAHARA, Tetsutaro e TSOTSOS, John K., (1983), "Building Knowledge-Based Systems: The PSN experience", IEEE-Computer, Vol.16, No.10, pp.83-88.

MYLOPOULOS, John, BERNSTEIN, Philip A. e WONG, Harry K. T., (1988), "A language facility for Designing Database-intensive applications", in Publication Data Readings in IA and Databases.

NEWELL, Allen e SIMON, H. A., (1972), "Human Problem Solving", Prentice-Hall, Englewood Clifs.

NEWELL, Allen (1982), "The Knowledge Level", Artificial Intelligence, No.18, North-Holland, pp.87-127.

PESSOA, Luiz A. F. C. (1989), "Modelos conexionistas e representação do conhecimento", Dissertação de Bacharelado COPPE-UFRJ.

PARTRIDGE, D. A. (1986), "Artificial Intelligence: Applications in the Future of Software Engineering", Ellis Horwood.

PETKOVIC, Dragutin e HINKLE, Eric B., (1987), "A Rule-Based System for Verifying Engineering Specifications in Industrial Visual Inspection Applications", IEEE-Transactions on Pattern Analysis and Machine Intelligence, Vol.9, No.2, pp.306-311.

PRESSMAN, Roger S. (1982), "Software Engineering: A Practitioner's Approach", Editora McGraw-Hill.

PRESSMAN, Roger S. (1988), "Software Engineering: A Practitioner's Aproach", Editora McGraw-Hill, segunda edição.

QUILLIAM, M.R. (1968), "Semantic memory", in Semantic Information Processing, M. Minsky (Ed.), MIT Press, Cambridge, MIT.

REITER, Raymond (1988), "On closed World Databases", in Publication Data Readings in IA and Databases.

REITER, Raymond (1988a), "Towards a logical reconstruction of Relational Database Theory", in Publication Data Readings in IA and Databases, pp.301-326.

RICH, Charles (1986), "A Formal Representation for Plans in the Programmerś Apprentice", Massachusetts Institute of Technology.

RICH, E. (1988), "Inteligência Artificial", McGraw-Hill.

RIEGER, Chuck (1976), "An Organization of Knowledge for Problem Solving and Language Comprehension", Artificial Intelligence 7, pp. 89-127.

ROCHA, Ana R. C. (1987), "Ambientes para Desenvolvimento de Software: Definição de Formas", Relatório Técnico ES-137/87, COPPE-UFRJ.

ROCHA, Ana R. C. (1989), "O Meta-Ambiente da Estação TABA", Relatório Técnico - Coppe/UFRJ.

ROCHA, Ana R. C. et al. (1990), "Uma visão funcional da Estação TABA", Relatório Técnico, COPPE/UFRJ.

ROLSTON, David W. (1988), "Principles of Artificial Intelligence and Expert Systems Development", McGraw-Hill.

ROSS, Douglas T. (1977), "Structured Analysis (SA): A language for Communicating Ideas", IEEE Transactions on Software Engineering, Vol.SE-3, No.1, pp.16-34.

ROSS, Douglas T. (1985), "Applications and Extensions of SADT", IEEE-Computer, April, pp.25-33.

ROUSSOPOULOS, Nicholas e MYLOPOULOS, John, (1988), "Using Semantic Networks for Database management", in Publication Data Readings in IA and Databases, pp.112-137.

SCHANK, R.C. (1975), "The structure of episodes in memory", D. Bobrow e A. Collins (eds.), Academic Press Inc..

SCHANK, R.C. e Abelsonn R.P., (1977), "Scripts, Plans, Goals and Understanding: An inquiry into human knowledge structures", Lawrence Erlbaum Associates Inc..

SCHUBERT, L.K. (1976), "Extending the Expressive Power of Semantic Networks", Artificial Intelligence 7, pp.163-198.

SCHUBERT, L.K., PAPALASKARIS, Mary A. e TAUGHER, Jay, (1983), "Determining Type, Part, Color and Time Relationships", IEEE-Computer, Vol.16, No.10, pp.53-60.

SCORE, Edward (1982), "Towards an Integrated Database-Prolog System", On Expert Database Systems.

SIEVERT, Gene E. e Mizell, Terrence A., (1985), "Specification Based Software Engineering with Tags", IEEE Transactions on Software Engineering.

SMITH, John Miles e SMITH, Diane C. P., (1988), "Database abstractions: Aggregation and Generalization", in Publication Data Readings in IA and Databases.

STAA, Arndt Von (1983), "Engenharia de Programas", Editora LTC.

STONEBRAKER, Michael (1986), "Adding Semantic Knowledge to a Relational Database System", University of California.

TRAVASSOS, Guilherme H. (1990), "FEGRES: ferramentas gráficas para engenharia de software", Tese COPPE-UFRJ.

TURNER, Ray (1984), "Software Engineering Methodology", A Prentice-Hall Company.

WEBBER, Bonnie Lynn (1983), "Logic and Natural Language", IEEE-Computer, Vol.16, No.10, pp.43-46.

WEBSTER, Dallas E. (1988), "Mapping the Design Information Representation Terrain", IEEE-Computer, December, pp.8-23.

WETHERBE, James C. (1984), "Análise de Sistemas para Sistemas de Informação por Computador", Editora Campus.

WOODS, William A. (1983), "Whats important about Knowledge Representation ?", IEEE-Computer, Vol.16, No.10, pp.22-27.

YEO, C., THORPE, J. e LONGSTAFF, J., (1985), "Knowledge Base Enhancements to Relational Databases", IEEE-Transactions on Software Engineering.

YOURDON, Edward (1982), "Managing the System Life Cycle", Yourdon Inc..

ZADEH, Lotfi A. (1983), "Commonsense Knowledge Representation Based on Fuzzy Logic", IEEE-Computer, Vol.16, No.10, pp.61-65.

ZANIOLO, Carlo (1984), "Object Oriented Database Systems and Knowledge Systems", Experts Database Systems.

Apêndice A

DESCRIÇÃO E REPRESENTAÇÃO DO CONHECIMENTO DO MÉTODO ANÁLISE ESTRUTURADA

Este apêndice apresenta uma descrição resumida das principais características do método análise estruturada, segundo abordagem (GANE-83) e a representação do conhecimento segundo a proposta deste trabalho.

A.1 - DESCRIÇÃO DO MÉTODO ANÁLISE ESTRUTURADA

A Análise Estruturada (SSA) é um método de desenvolvimento de sistemas formado de técnicas e instrumentos que auxiliam na análise e definição de requisitos de um produto de software a ser desenvolvido. Este método permite definir com detalhes o que o sistema faz e como é sua lógica funcional.

A Análise Estruturada utiliza a técnica "top-down" para a especificação do modelo lógico do sistema. O método dispõe dos seguintes instrumentos:

- **Linguagem para representar Diagramas de fluxo de dados (DFD)**, que especificam as funções do sistema;
- Dicionário de dados, que contém informações detalhadas sobre os elementos do DFD;

. **Linguagens para descrição da lógica dos processos**, tais como: português estruturado, tabelas de decisão, árvores de decisão e português compacto;

. **Linguagem para representar Diagrama de acesso imediato**, que descreve a organização lógica dos dados.

O diagrama de fluxo de dados representa um papel fundamental para a construção do sistema. Seu objetivo básico é representar o fluxo lógico dos dados e as suas transformações. Este diagrama tem os seguintes objetivos:

. Mostrar as origens e destinos dos dados;

. Identificar e denominar as funções lógicas;

. Identificar e denominar os grupos de elementos de dados que ligam uma função a outra;

. Identificar os depósitos de dados acessados pelas funções.

A construção de um DFD é feita utilizando-se uma linguagem gráfica cuja notação é composta por quatro elementos básicos:

1) Entidade externa ou fonte dos dados

Indica as coisas ou pessoas que representam uma fonte ou um destino para as transações, significando que ela está fora dos limites do sistema. Este elemento é representado por um quadrado duplo.

2) Eluxo de dados

Significa a informação que está sendo transportada de um lado para o outro de suas extremidades, onde a direção da seta indica a direção do próprio fluxo. O conteúdo de cada fluxo de dados deve ser indicado em cada seta através de uma descrição breve e significativa. Este elemento é representado por uma seta.

3) Processo

Significa uma função a ser desenvolvida para transformar o fluxo de entrada e produzir o de saída. Na representação de um processo deve aparecer sua identificação, uma descrição da função que realiza e o local físico onde é desempenhado. Este elemento é representado por um retângulo arredondado.

4) Depósito de dados ou arquivo

Significa o lugar onde os dados são armazenados. Este elemento é representado por um retângulo aberto.

A Análise Estruturada envolve a construção de uma especificação funcional por refinamentos sucessivos. Cada processo é expandido de forma a se tornar um novo diagrama de fluxo de dados.

Cada processo de nível inferior deve se relacionar com um processo de nível superior. A expansão de um processo deve ser efetuada quantas vezes forem

necessárias, até que se tenha o diagrama no nível de detalhe desejado.

A forma mais clara de representar a expansão de um processo é desenhar os diagramas de fluxo de dados de menor nível dentro da divisa que representa o processo de nível superior. Todos os fluxos de dados que entram e saem de um processo expandido devem também entrar e sair da divisa. Já os depósitos de dados só devem ser desenhados dentro da divisa se forem criados e processados apenas por este processo.

O processo de construção dos diagramas de fluxos de dados pode ser resumido nas etapas abaixo:

1) Identificar as entidades externas envolvidas;

2) Identificar as entradas e saídas esperadas;

3) Identificar as consultas e os pedidos de informação que possam surgir;

4) Desenhar o primeiro esboço do diagrama;

5) Verificar se todas as entradas e saídas listadas foram incluídas, exceto aquelas que tratam de erros e exceções;

6) Produzir um segundo esboço mais claro;

7) Rever o segundo esboço com um representante do usuário, anotando qualquer mudança resultante da revisão;

8) Produzir uma expansão de nível inferior para cada processo definido no segundo esboço;

9) Rever novamente o diagrama produzido.

A construção de diagramas de fluxo de dados por meio de uma abordagem top-down permite que todas as funções sejam subdivididas gradativamente. Este detalhamento e a consequente revisão do diagrama devem ser efetuados repetidamente, até que seja atingido o nível de detalhe desejado.

Uma vez definido um processo e especificadas suas entradas e saídas, um resumo de sua lógica deve ser feito de forma clara e sem ambiguidades. Para esta tarefa existem os seguintes instrumentos: Tabelas de decisão, árvores de decisão, português estruturado e português compacto.

Tendo em vista esses instrumentos terem o objetivo de expressar a lógica de processos, cada um trás características particulares que os tornam apropriados conforme o problema tratado.

Com relação às tabelas e árvores de decisão, as normas abaixo contribuem para direcionar o uso mais adequado desses instrumentos.

- **Norma 1**: utilizar uma árvore de decisão quando o número de decisões for pequeno e nem toda combinação de condições for possível; usar uma tabela de decisão quando o número de ações for grande e ocorram muitas combinações de condições.

- **Norma 2**: Utilizar uma tabela de decisão se existirem dúvidas de que a árvore de decisão mostra toda a complexidade do problema.

- **Norma 3**: Mesmo que seja necessário uma tabela de decisão para descobrir a lógica, procurar representá-la como uma árvore desde que a Norma 1 não seja violada.

Árvores de decisão e tabelas de decisão são os instrumentos que lidam com os processos de ramificação complexa, encontradas normalmente em cálculos de uma forma geral. Entretanto muitos dos processos que é preciso documentar não são tão complexos, possuindo operações "faça isto, depois aquilo ...", algumas decisões e alguns ciclos. Os processos em análise estruturada são compostos de combinações apropriadas de instruções do tipo passo-a-passo (como Mover e Somar), de decisões binárias (se-então-senão-logo) e de ciclos, podendo ser executados manualmente ou por computador. Essas especificações podem ser escritas utilizando o português estruturado ou compactado.

Quando a lógica é escrita com sentenças em português, utilizando as convenções de letra maiúsculas e deslocamentos vertical, esta lógica é conhecida como português estruturado. As convenções para o português estruturado são:

1. A lógica de todos os processos num sistema é expressa é expressa como uma combinação de estruturas de sequência, decisão, caso e repetição;

2. Manter em observação as regras de ambiguidade do português;
3. As palavras-chave "SE, ENTÃO, SENÃO, LOGO, REPETIR e ATÉ" devem ser escritas com letras maiúsculas e as estruturas devem ser deslocadas verticalmente para mostrar sua hierarquia lógica;
4. Blocos de instruções podem ser agrupados recebendo um nome significativo que descreva a função deles e que seja escrito também com letras maiúsculas;
5. Quando usarmos uma palavra ou frase que esteja definida em um dicionário de dados, a palavra ou frase deve ser sublinhada.

O português estruturado apresenta muito da precisão de um programa de computador, mas não é um programa de computador. Não há especificação de leitura e gravação de arquivos físicos, nenhuma preparação de contadores ou chaves ou qualquer projeto físico.

Para um estilo equivalente ao do português, o português compacto refleti uma análise completa e precisa. É possível incluir àrvores de decisão ao se escrever um português compacto, se se tiver certeza de que elas serão compreendidas.

O português compacto é familiar em aparência e estruturalmente equivalente ao português estruturado. As convenções para o português compacto podem ser resumidas da seguinte maneira.

1. As operações sequenciais são apresentadas como

instruções imperativas para serem desempenhadas de forma rotineira, simples e direta.

2. As estruturas "SE, ENTÃO, SENÃO e LOGO" são apresentadas com notação decimal e deslocamento vertical para demonstrar aninhamento.
3. As condições SENÃO são apresentadas como "Para (explicação de condição)".
4. As estruturas de caso são apresentadas como tabelas.
5. Quando houver condições de exceção genuínas, a estrutura "Ação-1 a menos que condição onde neste caso Ação-2" pode ser usada para maior clareza.

A figura (A.1) resume os pontos fortes e fracos relativos aos quatro instrumentos para a especificação da lógica de processo.

U S O	ÁRVORES DE DECISÃO	TABELAS DE DECISÃO	PORTUGUÊS ESTRUTURADO	PORTUGUÊS COMPACTO
VERIFICAÇÃO LÓGICA	MODERADO	MUITO BOM	BOM	MODERADO
EXIBIR ESTRUTURA LÓGICA	MUITO BOM (MAS SOMENTE DECISÃO)	MODERADO (SOMENTE DECISÃO)	BOM (TODAS)	MODERADO (TODAS, DEPEND. AUTOR)
SIMPLICIDADE (FACILIDADE DE USO)	MUITO BOM	DE MUITO POBRE A POBRE	MODERADO	BOM
VERIFICAÇÃO PELO USUÁRIO	BOM	POBRE (A MENOS TREINA.USUÁRIO)	DE POBRE A MODERADO	BOM
ESPECIFICAÇÃO DE PROGRAMA	MODERADO	MUITO BOM	MUITO BOM	MODERADO
LEITURA-PELO-COMPUTADOR	POBRE	MUITO BOM	MUITO BOM	MUITO BOM
AVERIGUAÇÃO-PELO-COMPUTADOR	POBRE	MUITO BOM	MODERADO (NECESSITA DE SINTAXE)	POBRE
ALTERABILIDADE	MODERADO	POBRE (A MENOS DE REGRA SIMPLES	BOM	BOM

Figura A.1: Comparação dos instrumentos da lógica de processos (GANE-83))

Pode-se, à partir das indicações da figura anterior, resumir a situação mais conveniente para o uso de cada um dos instrumentos de especificação de lógica de processos, como a seguir:

O uso de **árvores de decisão** é melhor quando se trata de verificação lógica ou de decisões moderadamente complexas que resultam em até 10-15 ações. As árvores de decisão também são úteis para mostrar a lógica de uma tabela de decisão aos usuários.

O uso de **tabelas de decisão** é melhor em problemas que envolvem combinações complexas de até 5-6 condições. As tabelas de decisão podem lidar com qualquer número de ações; numerosas combinações de condições podem dificultar o manuseio de tabelas de decisão.

O emprego de **português estruturado** é melhor toda vez que o problema envolver a combinação de sequência de ações com decisões ou ciclos.

O emprego de **português compacto** é melhor para apresentar lógica moderadamente complexa desde que o analista tiver certeza que não possam surgir ambiguidades.

A.2 - REPRESENTAÇÃO DO CONHECIMENTO DO MÉTODO ANÁLISE ESTRUTURADA.

Nesta seção, apresentamos uma representação do método análise estruturada de forma a demonstrar a utilização do sistema de representação descrito neste trabalho.

É importante ressaltar que, conforme apresentado ao longo do trabalho, podem existir várias versões de uma representação para uma mesma aplicação (no nosso caso o método SSA), tendo em vista a identificação dos objetos, a expressão de suas descrições, como ainda, da especificação do nível de conhecimento desejado.

Buscamos nesta representação destacar as caracterísiticas básicas do método e de como pode ser sua especificação no sistema de representação. Em certos momentos, buscamos expressar o conhecimento do método de forma que didaticamente pudessemos mostrar a flexibilidade do sistema, como ainda, inserimos a orientação de (KELLER-90) que formaliza algumas regras básicas para o desenho de DFD's.

A representação do conhecimento do método análise estruturada está composta de:

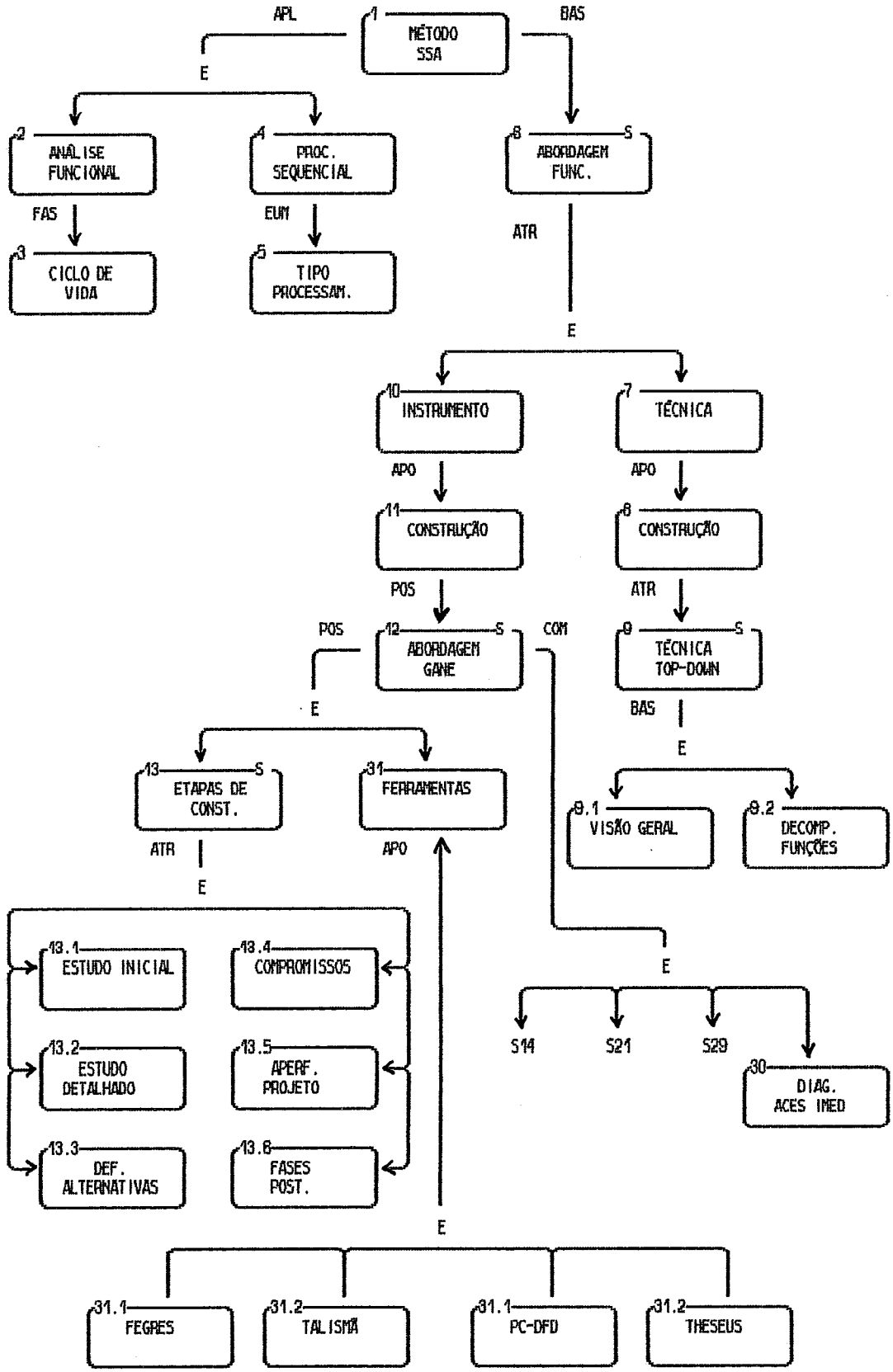
. Rede de conhecimento;

. Imagem modular;

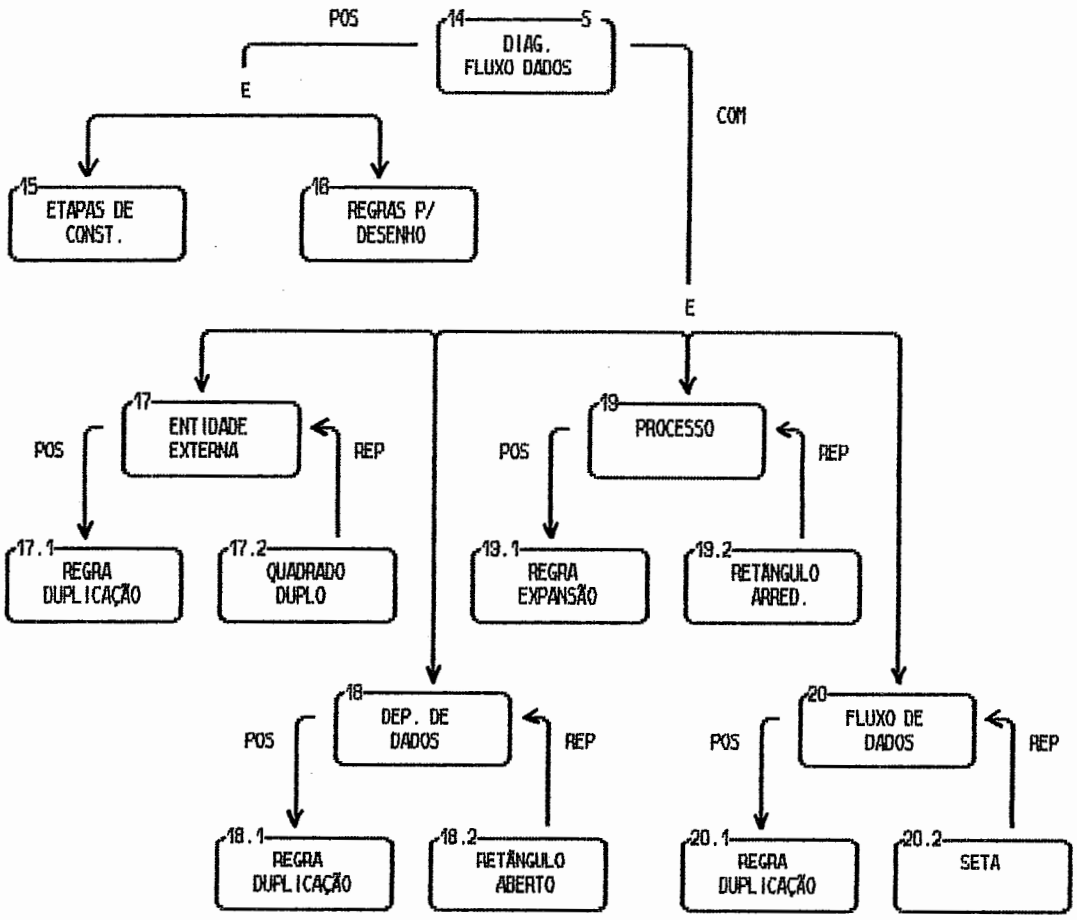
. Segmentação; e,

. Relatório da base de conhecimento.

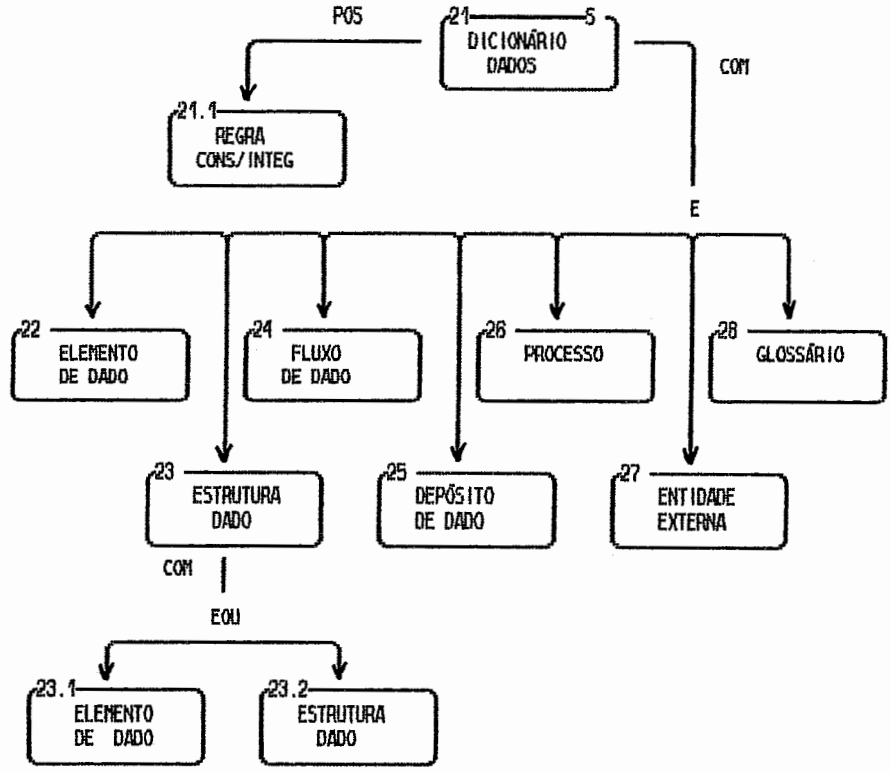
REDE DE CONHECIMENTO: Método SSA

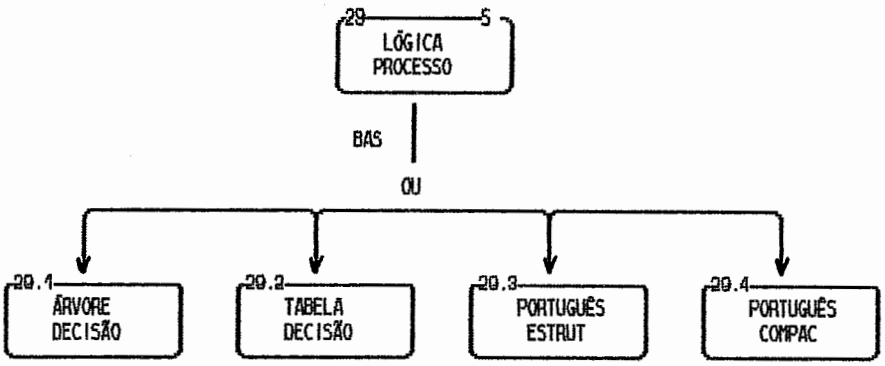


REDE DE CONHECIMENTO: Método SSA (continuação)

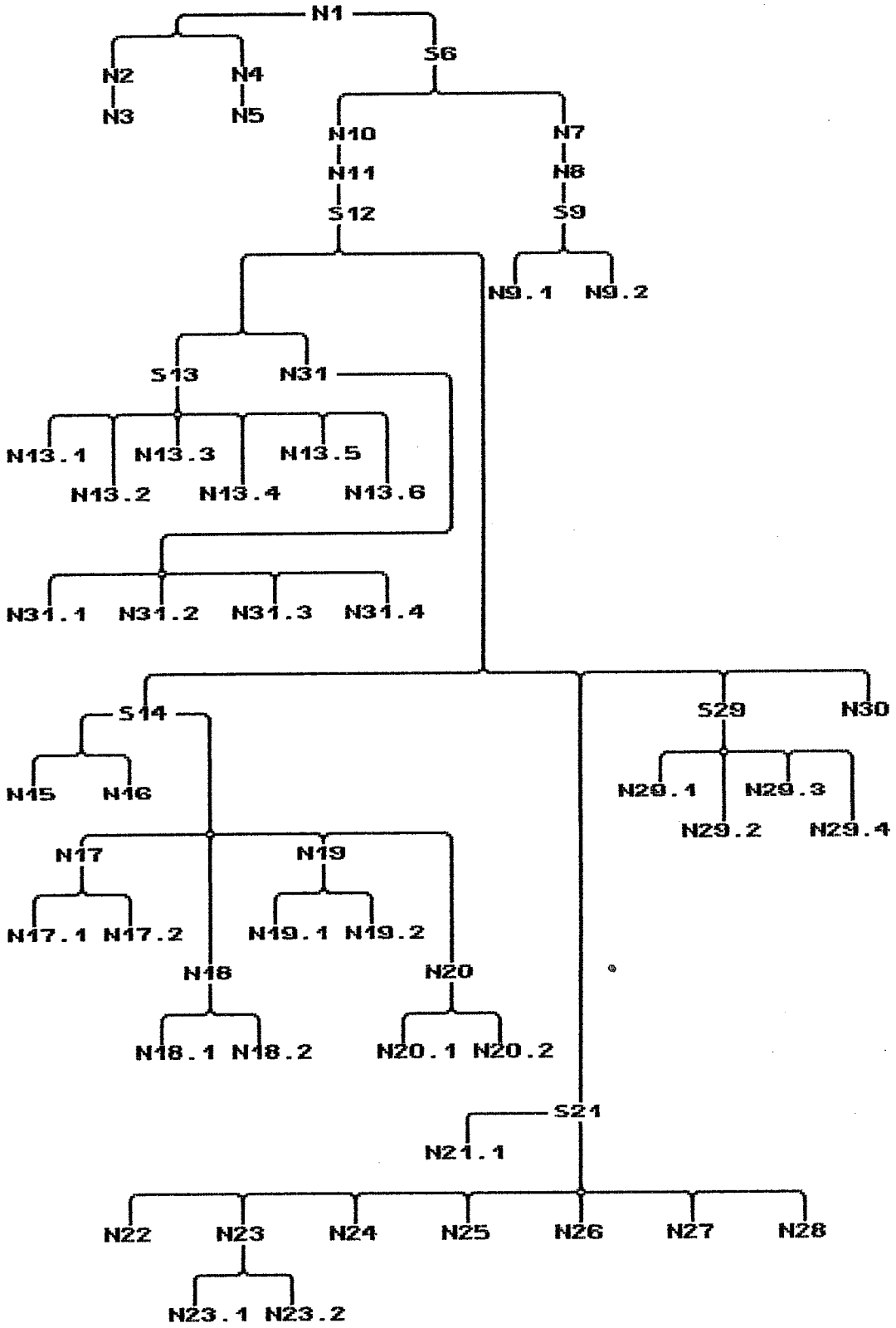


REDE DE CONHECIMENTO: Método SSA (continuação)

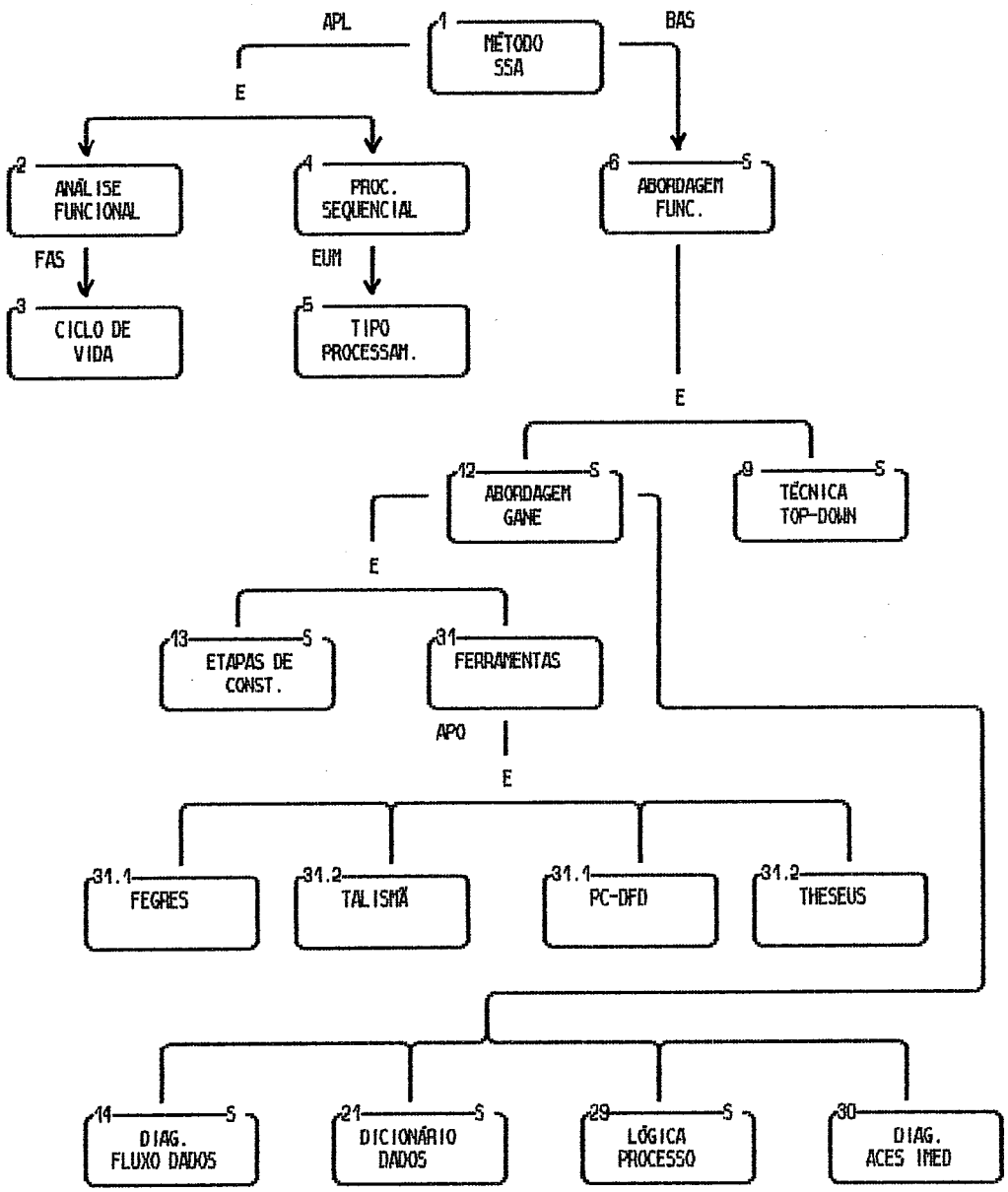




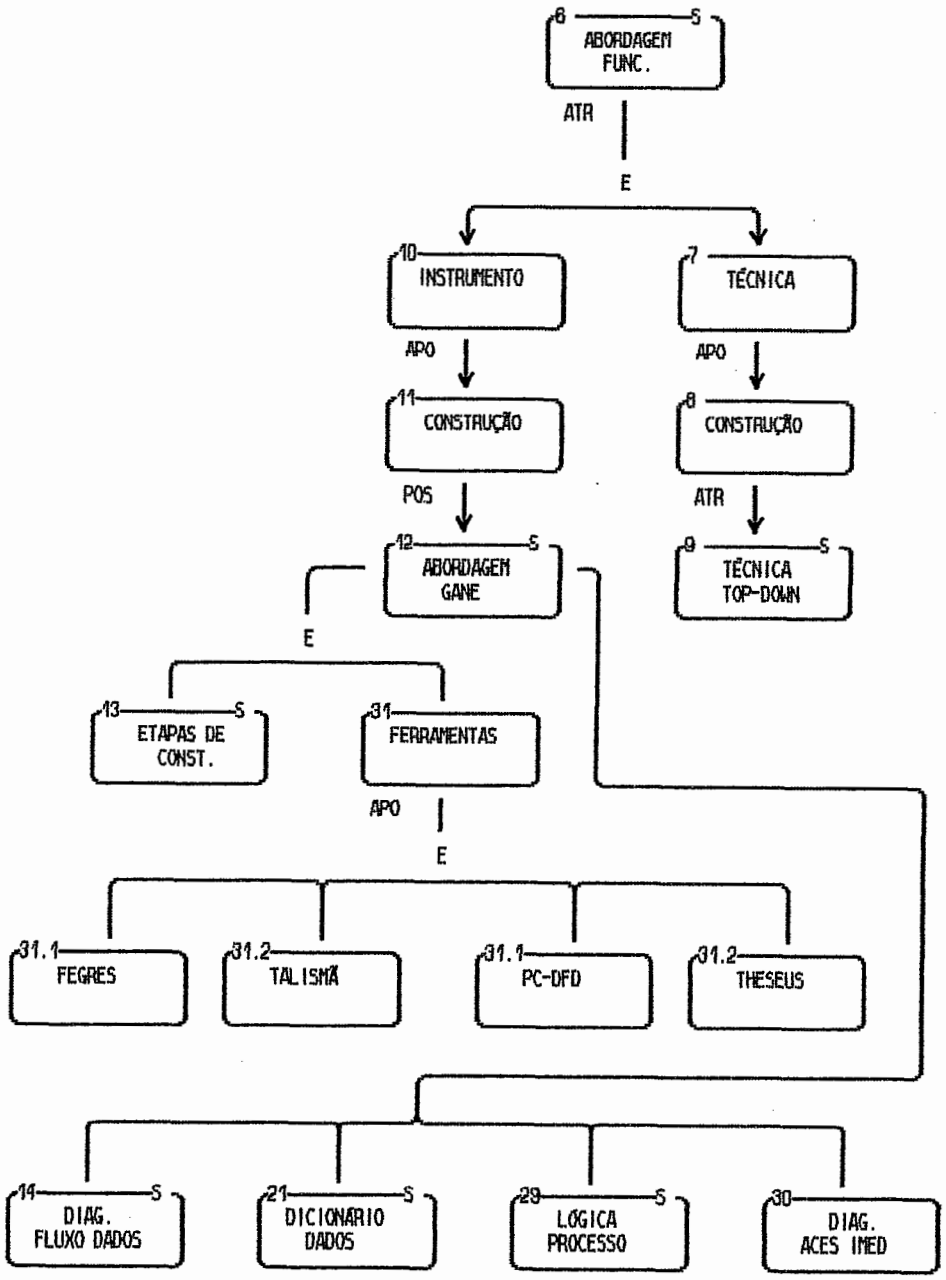
. IMAGEM MODULAR: Rede de conhecimento Método SSA



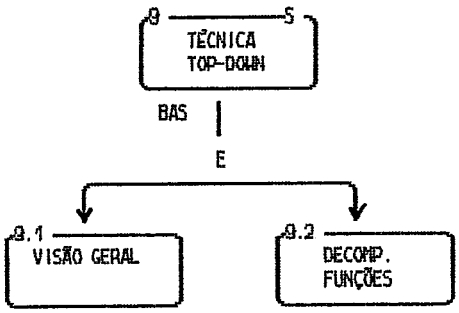
SEGMENTO 01:


APL
1
MÉTODO SSA
BAS
E
2
ANÁLISE FUNCIONAL
4
PROC. SEQUENCIAL
6 S
ABORDAGEM FUNC.
FAS
EUM
3
CICLO DE VIDA
5
TIPO PROCESSAM.
E
12 S
ABORDAGEM GANE
9 S
TÉCNICA TOP-DOWN
E
13 S
ETAPAS DE CONST.
31
FERRAMENTAS
APO
E
31.1
FEGRES
31.2
TALISMÃ
31.1
PC-DFD
31.2
THESEUS
14 S
DIAG. FLUXO DADOS
21 S
DICIONÁRIO DADOS
29 S
LÓGICA PROCESSO
30
DIAG. ACES IMED

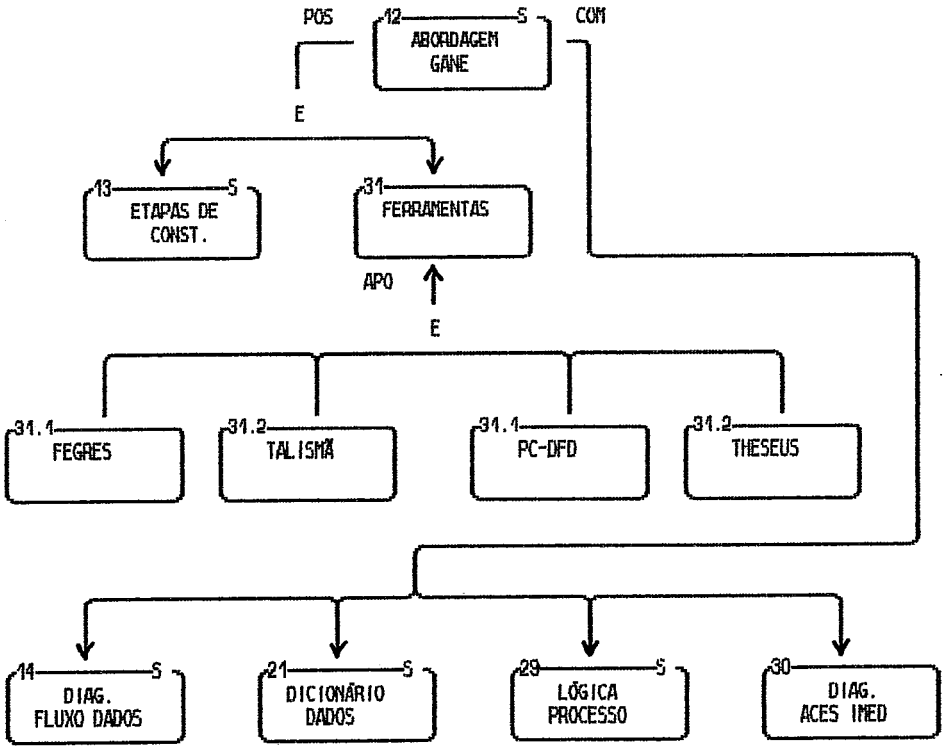
SEGMENTO 02:


6 S
ABORDAGEM FUNC.
ATR
E
10
INSTRUMENTO
7
TÉCNICA
APO
APO
11
CONSTRUÇÃO
8
CONSTRUÇÃO
POS
ATR
12 S
ABORDAGEM GANE
9 S
TÉCNICA TOP-DOWN
E
13 S
ETAPAS DE CONST.
31
FERRAMENTAS
APO
E
31.1
FEGRES
31.2
TALISMÃ
31.1
PC-DFD
31.2
THESEUS
14 S
DIAG. FLUXO DADOS
21 S
DICIONARIO DADOS
29 S
LÓGICA PROCESSO
30
DIAG. ACES IMED

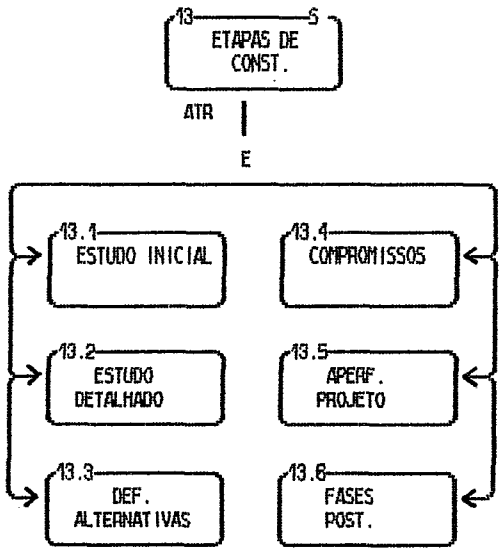
SEGMENTO 03:



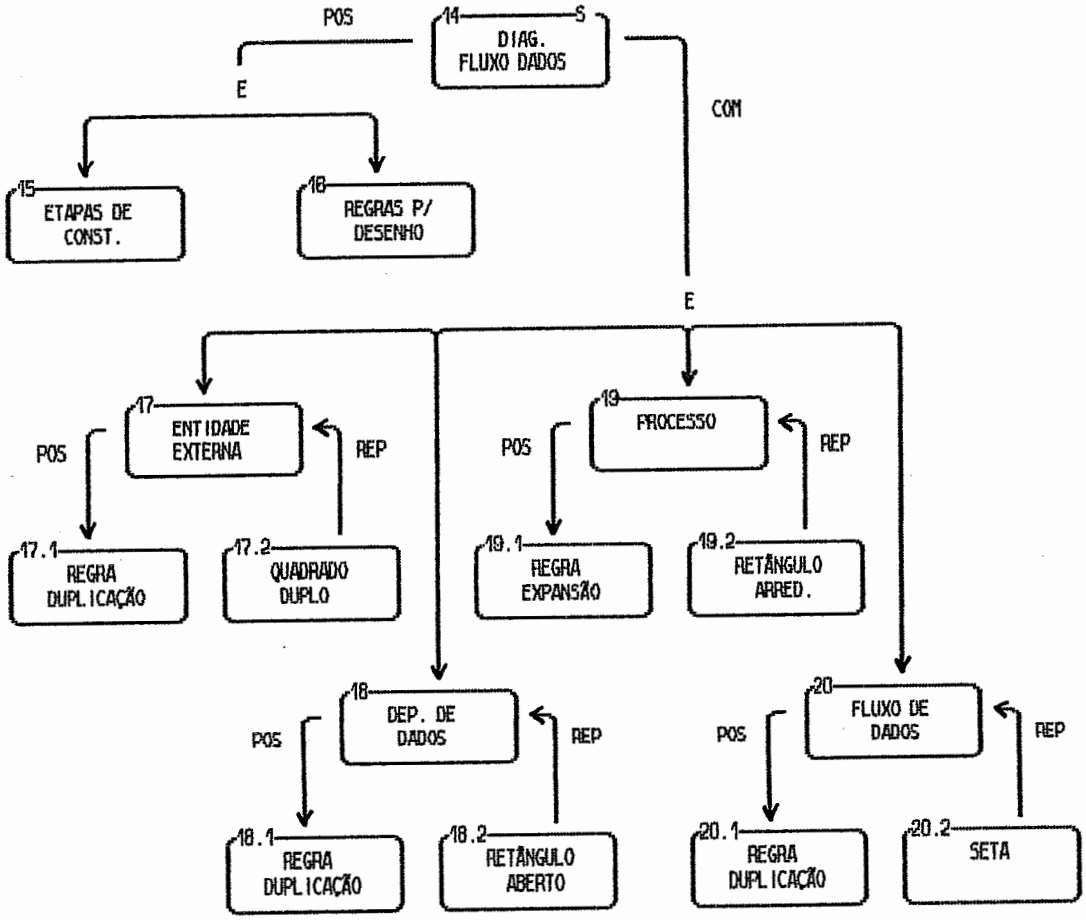
SEGMENTO 04:



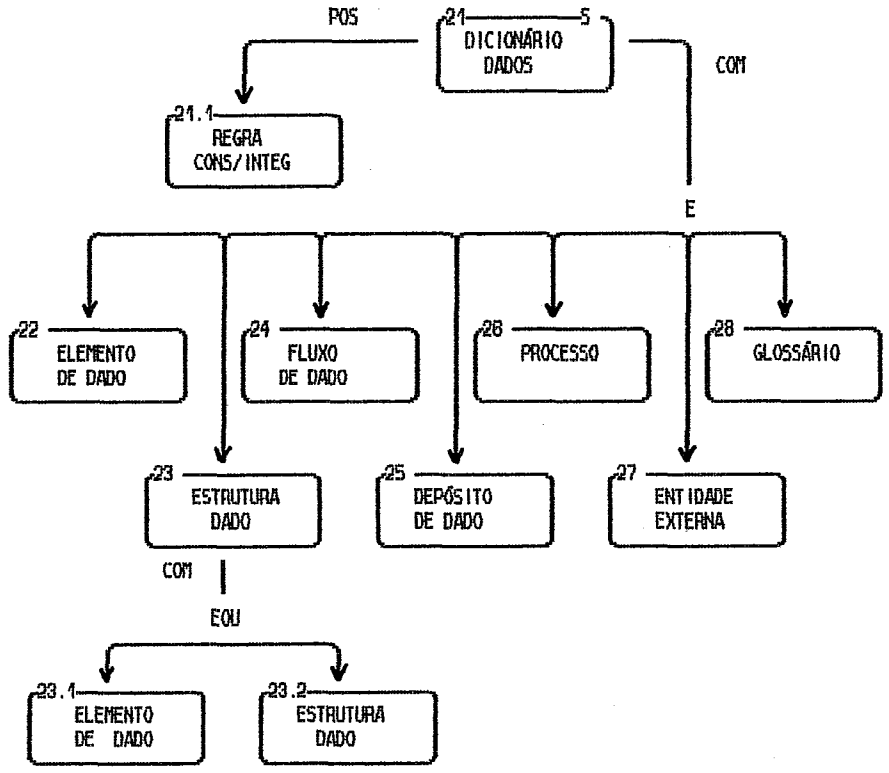
SEGMENTO 05:


13
S
ETAPAS DE CONST.
ATR
E
13.1
ESTUDO INICIAL
13.2
ESTUDO DETALHADO
13.3
DEF. ALTERNATIVAS
13.4
COMPROMISSOS
13.5
APERF. PROJETO
13.6
FASES POST.

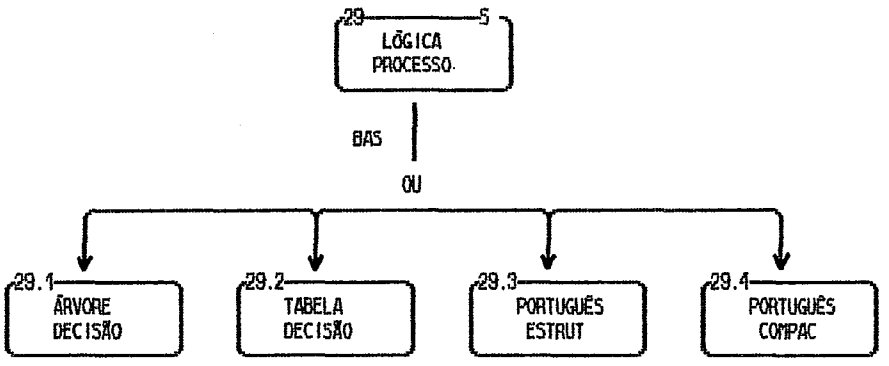
SEGMENTO 06:


14
DIAG. FLUXO DADOS
S
POS
E
COM
15
ETAPAS DE CONST.
16
REGRAS P/ DESENHO
E
17
ENTIDADE EXTERNA
POS
REP
17.1
REGRA DUPLICAÇÃO
17.2
QUADRADO DUPLO
19
PROCESSO
POS
REP
19.1
REGRA EXPANSÃO
19.2
RETÂNGULO ARRED.
18
DEP. DE DADOS
POS
REP
18.1
REGRA DUPLICAÇÃO
18.2
RETÂNGULO ABERTO
20
FLUXO DE DADOS
POS
REP
20.1
REGRA DUPLICAÇÃO
20.2
SETA

SEGMENTO 07:



SEGMENTO 08:



```
================================================================
RCQ-Metodos            BASE DE CONHECIMENTO            pag:   1
REDE    : Metodo SSA                                    data:
DOMINIO: Metodos de desenvolvimento de software       01/12/90
================================================================

ESTATISTICAS

      COMENTARIO: Estatisticas quantitativas da rede de
                  conhecimento.

          NOS.....................:    58   100,00%

          NOS-ATIVOS..............:    48    82,75%

          NOS-FRACOS..............:    10    17,24%

          NOS-DE-FUNCAO-REDE......:    27    46,55%

          NOS-DE-FUNCAO-QUADRO....:    31    53,44%

          NOS-DE-SEGMENTACAO......:     7    12,06%

          NOS-DE-CLASSIFICACAO....:     4     6,89%
----------------------------------------------------------------
CLASSIFICACAO

      METODO      : n6

      INSTRUMENTO: n11

      TECNICA     : n8, n9
----------------------------------------------------------------
n1      - Metodo SSA

      * DESCRICAO

          Metodo de auxilio ao desenvolvimento de
           software.

          FUNCAO: r-rede              TIPO: a-ativo

          SEGMENTO: n-nao        CLASSIF.: n-nao

      * CONSTRUCAO-POSTERIOR

              QTDE:   3
               NOS: E(n2,n4), n6
          RELACOES: ap1, bas
          SENTIDOS: o, o
----------------------------------------------------------------
n2      - Analise Funcional

      * DESCRICAO

          Definicao das funcoes ou transformacoes
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:   2
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n2      - Analise Funcional

          necessarias ao produto de software.

          FUNCAO: r-rede            TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n1
        RELACOES: ap1
        SENTIDOS: d

      * CONSTRUCAO-POSTERIOR

            QTDE:   1
             NOS: n3
        RELACOES: fas
        SENTIDOS: o
----------------------------------------------------------
n3      - Ciclo de vida

      * DESCRICAO

          Definicao das fases do processo de
          desenvolvimento.

          FUNCAO: r-rede            TIPO: f-fraco

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n2
        RELACOES: fas
        SENTIDOS: d
----------------------------------------------------------
n4      - Proc. sequencial

      * DESCRICAO

          Estabelece o uso em software de carac-
          teristica de processamento sequencial.

          FUNCAO: r-rede            TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao
```

```
==============================================================
RCQ-Metodos          BASE DE CONHECIMENTO              pag:   3
REDE    : Metodo SSA                                    data:
DOMINIO: Metodos de desenvolvimento de software      01/12/90
==============================================================
n4      - Proc. sequencial

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n1
        RELACOES: apl
        SENTIDOS: d

      * CONSTRUCAO-POSTERIOR

            QTDE:   1
             NOS: n5
        RELACOES: eum
        SENTIDOS: o
--------------------------------------------------------------
n5      - Tipo processam.

      * DESCRICAO

          Determina a caracteristica do software
          em relacao ao seu funcionamento.

          FUNCAO: r-rede            TIPO: f-fraco

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n4
        RELACOES: eum
        SENTIDOS: d
--------------------------------------------------------------
n6      - Abordagem func.

      * DESCRICAO

          Determina a forma de tratar do metodo
          atraves das funcoes do software.

          FUNCAO: r-rede            TIPO: a-ativo

          SEGMENTO: s-sim       CLASSIF.: m-metodo

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n1
        RELACOES: bas
        SENTIDOS: d
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:   4
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n6      - Abordagem func.

      * CONSTRUCAO-POSTERIOR

            QTDE:   2
             NOS: E(n10,n7)
        RELACOES: atr
        SENTIDOS: o
----------------------------------------------------------
n7      - Tecnica

      * DESCRICAO

        Identifica recursos para execucao de
        tarefas no processo de desenvolvimento.

        FUNCAO: r-rede            TIPO: f-fraco

        SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n6
        RELACOES: atr
        SENTIDOS: d

      * CONSTRUCAO-POSTERIOR

            QTDE:   1
             NOS: n8
        RELACOES: apo
        SENTIDOS: o
----------------------------------------------------------
n8      - Construcao

      * DESCRICAO

        Determina o uso de tecnicas construti-
        vas no processo de desenvolvimento.

        FUNCAO: r-rede            TIPO: a-ativo

        SEGMENTO: n-nao       CLASSIF.: t-tecnica

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n7
        RELACOES: apo
        SENTIDOS: d
```

```
==============================================================
RCQ-Metodos          BASE DE CONHECIMENTO              pag:   5
REDE    : Metodo SSA                                    data:
DOMINIO: Metodos de desenvolvimento de software       01/12/90
==============================================================
n8      - Construcao

      * CONSTRUCAO-POSTERIOR

            QTDE:   1
             NOS: n9
        RELACOES: atr
        SENTIDOS: o
--------------------------------------------------------------
n9      - Tecnica top-down

      * DESCRICAO

        Estabelece a tecnica de orientacao de-
        dutiva, ou seja, do geral p/ particular

        FUNCAO: r-rede              TIPO: a-ativo

        SEGMENTO: s-sim       CLASSIF.: t-tecnica

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n8
        RELACOES: atr
        SENTIDOS: d

      * CONSTRUCAO-POSTERIOR

            QTDE:   2
             NOS: E(n9.1,n9.2)
        RELACOES: bas
        SENTIDOS: o
--------------------------------------------------------------
n9.1    - Visao geral

      * DESCRICAO

        Determina que as decisoes de mais alto
        nivel sejam tomadas primeiramente.

        FUNCAO: r-rede              TIPO: a-ativo

        SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n9
        RELACOES: bas
        SENTIDOS: d
--------------------------------------------------------------
```

RCQ-Metodos **BASE DE CONHECIMENTO** pag: 6
REDE : Metodo SSA data:
DOMINIO: Metodos de desenvolvimento de software 01/12/90

n9.2 - Decomp. funcoes

*** DESCRICAO**

Determina que as funcoes do software
sejam decompostas recursivamente.

FUNCAO: r-rede TIPO: a-ativo

SEGMENTO: n-nao CLASSIF.: n-nao

*** CONSTRUCAO-ANTERIOR**

QTDE: 1
NOS: n9
RELACOES: bas
SENTIDOS: d

---

n10 - Instrumento

*** DESCRICAO**

Identifica recursos para suportarem ati
vidades especificas do desenvolvimento.

FUNCAO: r-rede TIPO: f-fraco

SEGMENTO: n-nao CLASSIF.: n-nao

*** CONSTRUCAO-ANTERIOR**

QTDE: 1
NOS: n6
RELACOES: atr
SENTIDOS: d

*** CONSTRUCAO-POSTERIOR**

QTDE: 1
NOS: n11
RELACOES: apo
SENTIDOS: o

---

n11 - Construcao

*** DESCRICAO**

Identifica um recurso para a construcao
de software.

```
==============================================================
RCQ-Metodos          BASE DE CONHECIMENTO              pag:   7
REDE    : Metodo SSA                                    data:
DOMINIO: Metodos de desenvolvimento de software       01/12/90
==============================================================
n11     - Construcao

          FUNCAO: r-rede            TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: i-instrumento

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n10
         RELACOES: apo
         SENTIDOS: d

      * CONSTRUCAO-POSTERIOR

              QTDE:   1
               NOS: n12
         RELACOES: pos
         SENTIDOS: o
--------------------------------------------------------------
n12     - Abordagem Gane

      * DESCRICAO

          Identifica um tipo de abordagem de
          caracteristica particular.

          FUNCAO: r-rede            TIPO: a-ativo

          SEGMENTO: s-sim       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n11
         RELACOES: pos
         SENTIDOS: d

      * CONSTRUCAO-POSTERIOR

              QTDE:   6
               NOS: e(n13,n31), e(n14,n21,n29,n30)
         RELACOES: pos, com
         SENTIDOS: o, o
--------------------------------------------------------------
n13     - Etapas de const.

      * DESCRICAO

          Identifica a existencia de etapas para
          guiarem a construcao de software.
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:   8
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n13     - Etapas de const.

          FUNCAO: r-rede              TIPO: a-ativo

          SEGMENTO: s-sim        CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n12
          RELACOES: pos
          SENTIDOS: d

      * CONSTRUCAO-POSTERIOR

              QTDE:   6
               NOS: E(n13.1,n13.2,n13.3,n13.4,n13.5,n13.6)
          RELACOES: atr
          SENTIDOS: o
----------------------------------------------------------
n13.1   - Estudo inicial

      * DESCRICAO

          O estudo inicial deve ser rapido e
          barato, e avaliar a solicitacao.

          FUNCAO: q-quadro            TIPO: a-ativo

          SEGMENTO: n-nao        CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n13
          RELACOES: atr
          SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: const_estini

             COMENTARIO: Identifica as atividades/produtos
                         para o estudo inicial.

          TIPO: p-procedimento

          QTDE-FATOS:   3

              FATO1: Analise da situacao atual.
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:   9
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n13.1   - Estudo inicial

            FATO2: Quais melhoramentos possiveis.

            FATO3: Consequencia do novo sistema.
----------------------------------------------------------
n13.2   - Estudo detalhado

      * DESCRICAO

          O estudo detalhado baseia-se no estudo
          inicial, para uma documentacao refinada

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n13
          RELACOES: atr
          SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: const_estdet

             COMENTARIO: Identifica as atividades/produto
                         para o estudo detalhado.

          TIPO: p-procedimento

          QTDE-FATOS:   4

              FATO1: Detalhes dos usuarios do novo sistema.

              FATO2: Modelo logico do sistema existente.

              FATO3: Declaracao aperfeicoada de rendimento.

              FATO4: Estimativas de custo e orcamento.
----------------------------------------------------------
n13.3   - Def.alternativas

      * DESCRICAO

          A definicao de alternativas permite se
          tomar decisoes sobre o novo sistema.

          FUNCAO: q-quadro          TIPO: a-ativo
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  10
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n13.3   - Def.alternativas

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n13
        RELACOES: atr
        SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: const_defalt

             COMENTARIO: Identifica as atividades/produtos
                         para a definicao de alternativas

          TIPO: p-procedimento

          QTDE-FATOS:   3

              FATO1: Objetivos para o novo sistema.

              FATO2: Modelo logico do novo sistema.

              FATO3: Projetos fisicos alternat./experim.
----------------------------------------------------------
n13.4   - Compromissos

      * DESCRICAO

          Esta etapa busca obter o compromisso
          dos usuarios tomadores de decisao.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n13
        RELACOES: atr
        SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: const_compus
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  11
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n13.4   - Compromissos

          COMENTARIO: Identifica os pontos de apresen-
                      tacao do sistema para o grupo de
                      usuarios.

        TIPO: p-procedimento

        QTDE-FATOS:   5

            FATO1: Diag. fluxo dados do sistema existente

            FATO2: Limitacoes do sistema/situacao atual

            FATO3: Modelo logico do novo sistema.

            FATO4: Cada sistema alternativo formulado.

            FATO5: Pedido de orientacao s/ alternativas.
----------------------------------------------------------
n13.5   - Aperf. projeto

    * DESCRICAO

        O aperfeicoamento do projeto busca ob-
        ter um projeto fisico mais seguro.

        FUNCAO: q-quadro          TIPO: a-ativo

        SEGMENTO: n-nao       CLASSIF.: n-nao

    * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n13
        RELACOES: atr
        SENTIDOS: d

    * ESPECIFICACAO

        IDENTIFICACAO: const_apepro

          COMENTARIO: Identifica as atividades/produtos
                       do aperfeicoamento do projeto
                      fisico.

        TIPO: p-procedimento

        QTDE-FATOS:   5

            FATO1: Aperfeicoamento do modelo logico.
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  12
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software     01/12/90
==========================================================
n13.5   - Aperf. projeto

            FATO2: Projeto do banco de dados fisico.

            FATO3: Hierarquia modular das funcoes.

            FATO4: Definicao de novas tarefas de rotina.

            FATO5: Observacao sobre estimativa.
----------------------------------------------------------
n13.6   - Fases post.

      * DESCRICAO

          Estabelece as fases do desenvolvimento
          do projeto.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n13
          RELACOES: atr
          SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: const_faspos

             COMENTARIO: Identifica as fases no desenvol-
                         vimento.

          TIPO: p-procedimento

          QTDE-FATOS:   9

              FATO1: Plano implementacao, teste e aceitacao

              FATO2: Desenvolvimento paralelo dos programas

              FATO3: Conversao e formacao do banco de dados

              FATO4: Teste e aceitacao do sistema

              FATO5: Teste do sistema sob condicoes reais

              FATO6: Compromisso do sistema c/operacao real
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  13
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software     01/12/90
==========================================================
n13.6   - Fases post.

              FATO7: Medida do desempenho do sistema

              FATO8: Comparacao dos recursos/desempenho

              FATO9: Analise e priorizacao da manutencao
----------------------------------------------------------
n14     - Diag.fluxo dados

      * DESCRICAO

          Linguagem grafica utilizada para o en-
          tendimento do sistema e de suas funcoes

          FUNCAO: r-rede              TIPO: a-ativo

          SEGMENTO: s-sim        CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n12
          RELACOES: com
          SENTIDOS: d

      * CONSTRUCAO-POSTERIOR

              QTDE:   6
               NOS: E(n15,n16), E(n17,n18,n19,n20)
          RELACOES: pos, com
          SENTIDOS: o, o
----------------------------------------------------------
n15     - Etapas de const.

      * DESCRICAO

          Identifica a existencia de etapas de
          construcao para uso do DFD.

          FUNCAO: q-quadro            TIPO: a-ativo

          SEGMENTO: n-nao        CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n14
          RELACOES: pos
          SENTIDOS: d
```

```
==============================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  14
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software     01/12/90
==============================================================
n15     - Etapas de const.

      * ESPECIFICACAO

          IDENTIFICACAO: const_dfd

             COMENTARIO: Identifica as etapas que guiam a
                         construcao do Diag. fluxo de
                         dados.

          TIPO: p-procedimento

          QTDE-FATOS:   9

              FATO1: Identificar ent. externas envolvidas

              FATO2: Identificar entradas/saidas esperadas

              FATO3: Identificar consultas/pedidos de inf.

              FATO4: Desenhar o primeiro esboco do diagrama

              FATO5: Verificar entradas/saidas requeridas

              FATO6: Produzir segundo esboco mais claro

              FATO7: Rever segundo esboco com o usuario

              FATO8: Produzir exp.inferior p/ cada processo

              FATO9: Rever novamente o diagrama produzido
--------------------------------------------------------------
n16     - Regras p/desenho

      * DESCRICAO

          Identifica a existencia de regras espe-
          cificas para o desenho de DFD's.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n14
          RELACOES: pos
          SENTIDOS: d

      * ESPECIFICACAO
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  15
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n16     - Regras p/desenho

          IDENTIFICACAO: reg_cons

             COMENTARIO: Descreve a regra da "conservacao"

          TIPO: h-heuristica

          QTDE-FATOS:   2

              FATO1: Fluxo de dados de saida do processo

              FATO2: Fluxo de dados das entradas processo

          QTDE-ACOES:   1

              ACAO1: Fluxo de dados correto.

          QTDE-REGRAS:   1

              REGRA1: se fato1 "funcao" fato2 entao acao1

      * ESPECIFICACAO

          IDENTIFICACAO: reg_inde

             COMENTARIO: Descreve a regra "independencia".

          TIPO: h-heuristica

          QTDE-FATOS:   2

              FATO1: Entradas/saidas do processo

              FATO2: Entradas/saidas do proprio processo

          QTDE-ACOES:   1

              ACAO1: Processo correto.

          QTDE-REGRAS:   1

              REGRA1: se fato1 "funcao" fato2 entao acao1

      * ESPECIFICACAO

          IDENTIFICACAO: reg_orde

             COMENTARIO: Descreve a regra "ordem".

          TIPO: h-heuristica
```

```
==============================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  16
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software     01/12/90
==============================================================
n16     - Regras p/desenho

          QTDE-FATOS:   2

              FATO1: Primeiro item a entrar no fluxo dados

              FATO2: Informacao de arquivo

          QTDE-ACOES:   2

              ACAO1: Primeiro item a sair do fluxo dados

              ACAO2: Pode ser processado em qualquer ordem

          QTDE-REGRAS:  2

              REGRA1: se fato1 entao acao1

              REGRA2: se fato2 entao acao2

      * ESPECIFICACAO

          IDENTIFICACAO: reg_parc

             COMENTARIO: Descreve a regra "parcimonia".

          TIPO: h-heuristica

          QTDE-FATOS:   2

              FATO1: Entradas no processo

              FATO2: Entradas necess. p/produzir as saidas

          QTDE-ACOES:   3

              ACAO1: Processo correto.

              ACAO2: Falta definir entradas p/ processo.

              ACAO3: Processo c/ entradas em excesso.

          QTDE-REGRAS:  3

              REGRA1: se fato1 = fato2 entao acao1

              REGRA2: se fato1 < fato2 entao acao2

              REGRA3: se fato1 > fato2 entao acao3

      * ESPECIFICACAO
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  17
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software     01/12/90
==========================================================
n16     - Regras p/desenho

          IDENTIFICACAO: reg_perm

             COMENTARIO: Descreve a regra "permanencia".

          TIPO: h-heuristica

          QTDE-FATOS:   2

              FATO1: Utiliza item de fluxo de dados

              FATO2: Utiliza informacao de arquivo

          QTDE-ACOES:   2

              ACAO1: Remover item do fluxo de dados.

              ACAO2: Nao remover informacao do arquivo.

          QTDE-REGRAS:   2

              REGRA1: se fato1 entao acao1

              REGRA2: se fato2 entao acao2

      * ESPECIFICACAO

          IDENTIFICACAO: reg_pers

             COMENTARIO: Descreve a regra "persistencia".

          TIPO: h-heuristica

          QTDE-FATOS:   2

              FATO1: Processo em execucao

              FATO2: Processo aguardando fluxo de dados

          QTDE-ACOES:   1

              ACAO1: Processo em procedimento correto.

          QTDE-REGRAS:   1

              REGRA1: se fato1 ou (.nao.fato1 e fato2) entao
                      acao1

      * ESPECIFICACAO
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  18
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n16     - Regras p/desenho

          IDENTIFICACAO: regras_dfd

             COMENTARIO: Identifica os tipos de regras
                         para auxilio no desenho de DFD's.

          TIPO: h-heuristica

          QTDE-FATOS:   6

              FATO1: Conservacao

              FATO2: Parcimonia

              FATO3: Independencia

              FATO4: Persistencia

              FATO5: Ordem

              FATO6: Permanencia

          QTDE-ACOES:   6

              ACAO1: ACESSAR (ident_espec = reg_cons)

              ACAO2: ACESSAR (ident_espec = reg_parc)

              ACAO3: ACESSAR (ident_espec = reg_inde)

              ACAO4: ACESSAR (ident_espec = reg_pers)

              ACAO5: ACESSAR (ident_espec = reg_orde)

              ACAO6: ACESSAR (ident_espec = reg_perm)

          QTDE-REGRAS:   6

              REGRA1: se fato1 entao acao1

              REGRA2: se fato2 entao acao2

              REGRA3: se fato3 entao acao3

              REGRA4: se fato4 entao acao4

              REGRA5: se fato5 entao acao5

              REGRA6: se fato6 entao acao6
----------------------------------------------------------
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  19
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n17     - Entidade externa


      * DESCRICAO

          Sao categorias de coisas/pessoas que
          representam fonte/destino p/ transacoes

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n14
          RELACOES: com
          SENTIDOS: d

      * CONSTRUCAO-POSTERIOR

              QTDE:   2
               NOS: n17.1, n17.2
          RELACOES: pos, rep
          SENTIDOS: o, d

      * ESPECIFICACAO

          IDENTIFICACAO: ent_refer

             COMENTARIO: Determina como uma entidade
                         externa e' referenciada.

          TIPO: r-regra

          QTDE-FATOS:   3

              FATO1: Simbolo de entidade externa

              FATO2: Contem letra minuscula

              FATO3: Localizacao no canto sup. esquerdo

          QTDE-ACOES:   1

              ACAO1: Referencia correta.

          QTDE-REGRAS:   1

              REGRA1: se fato1 e fato2 e fato3 entao acao1.
----------------------------------------------------------
```

RCQ-Metodos **BASE DE CONHECIMENTO** pag: 20
REDE : Metodo SSA data:
DOMINIO: Metodos de desenvolvimento de software 01/12/90

n17.1 - Regra duplicacao

*** DESCRICAO**

Descreve como duplicar simbolos de entidade externa no DFD.

FUNCAO: q-quadro TIPO: a-ativo

SEGMENTO: n-nao CLASSIF.: n-nao

*** CONSTRUCAO-ANTERIOR**

QTDE: 1
NOS: n17
RELACOES: pos
SENTIDOS: d

*** ESPECIFICACAO**

IDENTIFICACAO: dup_entext

COMENTARIO: Regra para duplicacao de entidade externa.

TIPO: r-regra

QTDE-FATOS: 2

FATO1: Existe mais de um simbolo p/ mesma ent ext.

FATO2: Existe linha inclinada no canto inf.direito

QTDE-ACOES: 1

ACAO1: Simbolo de duplicacao correto.

QTDE-REGRAS: 1

REGRA1: se fato1 e fato2 entao acao1

---

n17.2 - Quadrado duplo

*** DESCRICAO**

Representa a simbologia de uma entidade externa.

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  21
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n17.2   - Quadrado duplo

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n17
          RELACOES: rep
          SENTIDOS: o

      * ESPECIFICACAO

          IDENTIFICACAO: Con_entext

             COMENTARIO: Especifica a convencao do simbolo

          TIPO: r-regra

          QTDE-FATOS:   2

              FATO1: Simbolo e' quadrado

              FATO2: Lados cima e esquerda com tracos duplo

          QTDE-ACOES:   1

              ACAO1: Simbolo correto.

          QTDE-REGRAS:   1

              REGRA1: se fato1 e fato2 entao acao1
----------------------------------------------------------
n18     - Dep. de dados

      * DESCRICAO

          Lugar onde sao definidos os dados entre
           os processos.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n14
          RELACOES: com
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  22
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software     01/12/90
==========================================================
n18     - Dep. de dados

          SENTIDOS: d

      * CONSTRUCAO-POSTERIOR

            QTDE:   2
             NOS: n18.1, n18.2
        RELACOES: pos, rep
        SENTIDOS: o, d

      * ESPECIFICACAO

          IDENTIFICACAO: dep_refer

             COMENTARIO: Determina como um deposito de
                         dados e' referenciado.

          TIPO: r-regra

          QTDE-FATOS:   4

              FATO1: Simbolo de deposito de dados

              FATO2: Contem letra "D"

              FATO3: Contem numero arbitrario

              FATO4: Contem caixa na extremidade esquerda

          QTDE-ACOES:   1

              ACAO1: Referencia correta

          QTDE-REGRAS:   1

              REGRA1: se fato1 e fato4 e fato2 e fato3 entao
                      acao1
----------------------------------------------------------
n18.1   - Regra duplicacao

      * DESCRICAO

          Descreve como duplicar simbolos de
          deposito de dados no DFD.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  23
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n18.1   - Regra duplicacao

            QTDE:   1
             NOS: n18
        RELACOES: pos
        SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: dup_depdad

             COMENTARIO: Regra para duplicacao de deposito
                         de dados.

          TIPO: r-regra

          QTDE-FATOS:   2

              FATO1: Existe mais de um simbolo p/ mesmo dep
                     dados

              FATO2: Existe linhas verticais adicionadas `a
                     esq.

          QTDE-ACOES:   1

              ACAO1: Simbolo de duplicacao correto.

          QTDE-REGRAS:   1

              REGRA1: se fato1 e fato2 entao acao1
----------------------------------------------------------
n18.2   - Retangulo aberto

      * DESCRICAO

          Representa o simbolo de deposito de
          dados.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n18
        RELACOES: rep
        SENTIDOS: o

      * ESPECIFICACAO
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  24
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n18.2   - Retangulo aberto

          IDENTIFICACAO: con_depdad

             COMENTARIO: Especifica a convencao do simbolo

          TIPO: r-regra

          QTDE-FATOS:   2

              FATO1: Simbolo 'e retangulo aberto

              FATO2: Simbolo ligado em uma das extremidades

          QTDE-ACOES:   1

              ACAO1: Simbolo correto

          QTDE-REGRAS:   1

              REGRA1: se fato1 e fato2 entao acao1
----------------------------------------------------------
n19     - Processo

      * DESCRICAO

          Lugar onde sao especificadas as trans-
          formacoes nos dados.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n14
          RELACOES: com
          SENTIDOS: d

      * CONSTRUCAO-POSTERIOR

              QTDE:   2
               NOS: n19.1, n19.2
          RELACOES: pos, rep
          SENTIDOS: o, d

      * ESPECIFICACAO

          IDENTIFICACAO: pro_refer
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  25
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n19     - Processo

            COMENTARIO: Determina como um processo e
                        referenciado.

          TIPO: r-regra

          QTDE-FATOS:   4

              FATO1: Simbolo de processo

              FATO2: Contem um numero na parte superior

              FATO3: Contem verbo-ativo mais objeto no meio

              FATO4: Contem referencia fisica de localizaca
                     o

          QTDE-ACOES:   1

              ACAO1: Referencia correta

          QTDE-REGRAS:   1

              REGRA1: se fato1 e fato2 e fato3 e fato4 entao
                      acao1
----------------------------------------------------------
n19.1   - Regra expansao

      * DESCRICAO

          Descreve convencoes para expansao de
          processos.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n19
          RELACOES: pos
          SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: exp_proces

             COMENTARIO: Convencoes de expansao de
                         processos.
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  26
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n19.1   - Regra expansao

          TIPO: r-regra

          QTDE-FATOS:   2

              FATO1: E' um processo

              FATO2: E' um processo de nivel inferior

          QTDE-ACOES:   4

              ACAO1: Pode tornar-se um novo DFD

              ACAO2: Tem que se relacionar c/proc.superior

              ACAO3: Processo inf. e' numerado c/ prefixo
                     do sup

              ACAO4: O restante numeracao nivel inf. e' seq

          QTDE-REGRAS:   2

              REGRA1: se fato1 entao acao1

              REGRA2: se fato2 entao acao2 e acao3 e acao4

      * ESPECIFICACAO

          IDENTIFICACAO: exp_repres

             COMENTARIO: Descreve como representar a
                         expansao de processo.

          TIPO: p-procedimento

          QTDE-FATOS:   5

              FATO1: DFD menor nivel dentro da divisa proc.
                     sup.

              FATO2: Flx.dados proc. sup. devem e/s divisa

              FATO3: Flx.dados excl.nivel inf. podem sair
                     divisa

              FATO4: Identificar flx.dados nivel inf. c/ x

              FATO5: Dep. dados so' sao mostrados dentro
                     divisa
----------------------------------------------------------
```

```
==============================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  27
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software     01/12/90
==============================================================
n19.2    - Retangulo arred.


      * DESCRICAO

          O retangulo arredondado representa o
          simbolo de um processo.

          FUNCAO: r-rede            TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n19
          RELACOES: rep
          SENTIDOS: o
--------------------------------------------------------------
n20      - Fluxo de dados

      * DESCRICAO

          Especifica as informacoes que sao
          transportadas num DFD.

          FUNCAO: r-rede            TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n14
          RELACOES: com
          SENTIDOS: d

      * CONSTRUCAO-POSTERIOR

              QTDE:   2
               NOS: n20.1, n20.2
          RELACOES: pos, rep
          SENTIDOS: o, d
--------------------------------------------------------------
n20.1    - Regra duplicacao

      * DESCRICAO

          Descreve como duplicar simbolos de
          fluxo de dados.
```

```
==============================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  28
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software     01/12/90
==============================================================
n20.1   - Regra duplicacao

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n20
          RELACOES: pos
          SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: dup_fludad

             COMENTARIO: Regra para duplicacao de pares de
                         fluxo de dados.

          TIPO: r-regra

          QTDE-FATOS:   1

              FATO1: Existe par flx. dados iguais numa rela
                     -cao

          QTDE-ACOES:   1

              ACAO1: Utilizar setas com duas pontas.

          QTDE-REGRAS:   1

              REGRA1: se fato1 entao acao1
--------------------------------------------------------------
n20.2   - Seta

      * DESCRICAO

          Representa o simbolo de fluxo de dados.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n20
          RELACOES: rep
          SENTIDOS: o
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  29
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n20.2   - Seta

      * ESPECIFICACAO

          IDENTIFICACAO: con_fludad

             COMENTARIO: Especifica a convencao do simbolo
                         seta.

          TIPO: r-regra

          QTDE-FATOS:   2

              FATO1: Simbolo e' seta

              FATO2: Existe descricao no simbolo

          QTDE-ACOES:   1

              ACAO1: Simbolo correto.

          QTDE-REGRAS:   1

              REGRA1: se fato1 e fato2 entao acao1
----------------------------------------------------------
n21     - Dicionario dados

      * DESCRICAO

          Local onde se mantem as especificacoes
          do DFD.

          FUNCAO: r-rede            TIPO: a-ativo

          SEGMENTO: s-sim       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n12
          RELACOES: com
          SENTIDOS: o

      * CONSTRUCAO-POSTERIOR

              QTDE:   8
               NOS: n21.1, E(n22,n23,n24,n25,n26, n27, n28)
          RELACOES: pos, com
          SENTIDOS: o, o
----------------------------------------------------------
n21.1   - Regra cons/integ

      * DESCRICAO
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  30
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software     01/12/90
==========================================================
n21.1    - Regra cons/integ

          Descreve as verificacoes necessarias
          apos criacao dic. de dados.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n21
          RELACOES: pos
          SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: reg_dicdad

             COMENTARIO: Descreve as perguntas a serem
                         feitas apos criacao dic. dados.

          TIPO: p-procedimento

          QTDE-FATOS:   4

              FATO1: Ha' flx.dado sem fonte ou destino

              FATO2: Ha' elem.dado que nao possa ser aces-
                     sado

              FATO3: Ha' elem.dado nao existente entrada
                     proc.

              FATO4: Ha' elem.dado definido e nao utilizado
----------------------------------------------------------
n22      - Elemento de dado

      * DESCRICAO

          Sao informacoes que desejamos manter
          num dicionario de dados.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  31
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software     01/12/90
==========================================================
n22     - Elemento de dado

              NOS: n21
        RELACOES: com
        SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: des_eledad

             COMENTARIO: Descreve as informacoes sobre um
                         elemento de dados.

          TIPO: p-procedimento

          QTDE-FATOS:   8

              FATO1: Nome

              FATO2: Descricao

              FATO3: Pseudonimos

              FATO4: Elementos de dados relacionados

              FATO5: Dominio e significados dos valores

              FATO6: Comprimento

              FATO7: Codificacao

              FATO8: Outras informacoes para averiguacao
----------------------------------------------------------
n23     - Estrutura dado

      * DESCRICAO

          Descreve elementos de dados e outras
          estruturas de dados.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n21
        RELACOES: com
        SENTIDOS: d
```

```
==============================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  32
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software     01/12/90
==============================================================
n23     - Estrutura dado

      * CONSTRUCAO-POSTERIOR

            QTDE:   2
             NOS: EOU(n23.1,n23.2)
        RELACOES: com
        SENTIDOS: o

      * ESPECIFICACAO

          IDENTIFICACAO: tip_estdad

             COMENTARIO: Apresenta os tipos de estruturas
                         de dados

          TIPO: p-procedimento

          QTDE-FATOS:   3

              FATO1: Estruturas opcionais

              FATO2: Estruturas alternativas

              FATO3: Iteracoes de estruturas
--------------------------------------------------------------
n23.1   - Elemento de dado

      * DESCRICAO

          Sao informacoes que desejamos manter
          num dicionario de dados.

          FUNCAO: r-rede              TIPO: f-fraco

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n23
        RELACOES: com
        SENTIDOS: d
--------------------------------------------------------------
n23.2   - Estrutura dado

      * DESCRICAO

          Descreve elementos de dados e outras
          estruturas de dados.
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  33
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n23.2   - Estrutura dado

          FUNCAO: r-rede              TIPO: f-fraco

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n23
          RELACOES: com
          SENTIDOS: d
----------------------------------------------------------
n24     - Fluxo de dado

      * DESCRICAO

          Descreve o conteudo de um fluxo de dado
           a partir de suas estruturas de dados.

          FUNCAO: q-quadro            TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n21
          RELACOES: com
          SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: des_fludad

             COMENTARIO: Descreve as informacoes de um
                         fluxo de dados.

          TIPO: p-procedimento

          QTDE-FATOS:   4

              FATO1: A origem do fluxo de dado

              FATO2: O destino do fluxo de dado

              FATO3: Volumes de cada est.dados ou transacao

              FATO4: Implementacao fisica atual do flx dado
----------------------------------------------------------
n25     - Deposito de dado

      * DESCRICAO
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  34
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n25     - Deposito de dado

          Uma estrutura estatica onde seu conteu-
          do e' escrito em termos estrut.dados.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n21
         RELACOES: com
         SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: des_estdad

             COMENTARIO: Descreve os conteudos de um
                         deposito de dados.

          TIPO: p-procedimento

          QTDE-FATOS:   7

              FATO1: Estrutura de dados

              FATO2: Descricao estrutura de dado

              FATO3: Fluxos de dados que entram

              FATO4: Conteudo dos fluxos dados que entram

              FATO5: Flx dados que saem c/ argumento pesq

              FATO6: Analise do acesso imediato

              FATO7: Organizacao fisica
----------------------------------------------------------
n26     - Processo

      * DESCRICAO

          Descricao dos processos num dicionario
          de dados.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  35
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n26     - Processo

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n21
       RELACOES: com
       SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: des_proc

             COMENTARIO: Descreve as informacoes do pro-
                         cesso necessarias num dicionario
                         de dados.

          TIPO: p-procedimento

          QTDE-FATOS:   7

              FATO1: Nome

              FATO2: Descricao

              FATO3: Entradas

              FATO4: Resumo logico

              FATO5: Saidas

              FATO6: Referencias fisicas

              FATO7: Onde se encontra detalhes da logica
----------------------------------------------------------
n27     - Entidade externa

      * DESCRICAO

          Descricao das entidades externas num
          dicionario de dados.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

            QTDE:   1
             NOS: n21
       RELACOES: com
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  36
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n27     - Entidade externa

          SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: des_entext

             COMENTARIO: Descreve as informacoes da enti-
                         dade externa num dicionario de
                         dados.

          TIPO: p-procedimento

          QTDE-FATOS:   7

              FATO1: Nome

              FATO2: Fluxos de dados associados

              FATO3: Numeros de ocorrencia

              FATO4: Identificacao

              FATO5: Linguagem

              FATO6: Equipamento

              FATO7: Fonte de informacao
----------------------------------------------------------
n28     - Glossario

      * DESCRICAO

          Descricao das informacoes do glossario
          num dicionario de dados.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n21
          RELACOES: com
          SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: des_glos
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  37
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software     01/12/90
==========================================================
n28     - Glossario

            COMENTARIO: Descreve o conteudo do glossario
                        no dicionario de dados.

          TIPO: p-procedimento

          QTDE-FATOS:   9

              FATO1: Item do glossario

              FATO2: Descricao

              FATO3: Pseudonimos

              FATO4: Tipo

              FATO5: Valor

              FATO6: Significado

              FATO7: Dominio de valores

              FATO8: Outras informacoes p/ averiguacao

              FATO9: Elem/est.dados relacionados
----------------------------------------------------------
n29     - Logica processo

      * DESCRICAO

          Determina como expressar logica dos
          processos.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: s-sim       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n12
          RELACOES: com
          SENTIDOS: d

      * CONSTRUCAO-POSTERIOR

              QTDE:   4
               NOS: OU(n29.1,n29.2,n29.3,n29.4)
          RELACOES: bas
          SENTIDOS: o
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  38
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n29     - Logica processo

      * ESPECIFICACAO

          IDENTIFICACAO: esc_logpro

             COMENTARIO: Descreve normas para escolha de
                         tab.decisao vs arv.decisao.

          TIPO: r-regra

          QTDE-FATOS:   5

              FATO1: Numero de decisoes pequena

              FATO2: Toda combinacao nao for possivel

              FATO3: Numero de acoes grande

              FATO4: Ocorre muitas combinacoes de condicoes

              FATO5: A arv.dec. nao mostrar complexidade

          QTDE-ACOES:   2

              ACAO1: Utilizar arvore de decisao

              ACAO2: Utilizar tabela de decisao

          QTDE-REGRAS:   4

              REGRA1: se fato1 e fato2 entao acao1

              REGRA2: se fato3 e fato4 entao acao2

              REGRA3: se fato5 entao acao2

              REGRA4: se fato1 e fato2 e fato3 e fato4 entao
                      acao1

      * ESPECIFICACAO

          IDENTIFICACAO: est_proc

             COMENTARIO: Especifica os tipos de estruturas
                         possiveis na logica de processo.

          TIPO: p-procedimento

          QTDE-FATOS:   4
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  39
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n29     - Logica processo

            FATO1: Instrucoes sequenciais

            FATO2: Instrucoes de decisao

            FATO3: Instrucoes de decisao tipo "caso"

            FATO4: Instrucoes de repeticao
----------------------------------------------------------
n29.1   - Arvore decisao

      * DESCRICAO

          Especifica um tipo de recurso para se
          expressar logica de processo.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n29
          RELACOES: bas
          SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: uso_arvdec

             COMENTARIO: Descreve o uso adequado das
                         arvores de decisao.

          TIPO: p-procedimento

          QTDE-FATOS:   8

              FATO1: Moderado : verificacao logica

              FATO2: Muito bom: exibir estrutura logica

              FATO3: Muito bom: simplicidade

              FATO4: Bom      : verificacao pelo usuario

              FATO5: Moderado : especificacao de programa

              FATO6: Pobre    : leitura pelo computador
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  40
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software     01/12/90
==========================================================
n29.1   - Arvore decisao

              FATO7: Pobre    : averiguacao pelo computador

              FATO8: Moderado : alterabilidade
----------------------------------------------------------
n29.2   - Tabela decisao

      * DESCRICAO

          Especifica um tipo de recurso para se
          expressar logica de processo.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n29
          RELACOES: bas
          SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: uso_tabdec

             COMENTARIO: Descreve o uso adequado das
                         tabelas de decisao.

          TIPO: p-procedimento

          QTDE-FATOS:   8

              FATO1: Muito bom: verificacao logica

              FATO2: Moderado : exibir estrutura logica

              FATO3: De muito pobre a pobre: simplicidade

              FATO4: Pobre    : verificacao pelo usuario

              FATO5: Muito bom: especificacao de programa

              FATO6: Muito bom: leitura pelo computador

              FATO7: Muito bom: averiguacao pelo computador

              FATO8: Pobre    : alterabilidade
----------------------------------------------------------
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  41
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n29.3   - Portugues estrut


      * DESCRICAO

          Especifica um tipo de recurso para se
          expressar logica de processo.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n29
          RELACOES: bas
          SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: uso_porest

             COMENTARIO: Descreve o uso adequado do
                         portugues estruturado.

          TIPO: p-procedimento

          QTDE-FATOS:   8

              FATO1: Bom      : verificacao logica

              FATO2: Bom      : exibir estrutura logica

              FATO3: Moderado : simplicidade

              FATO4: De pobre a moderado:verificacao usuari
                     o

              FATO5: Muito bom: especificacao de programa

              FATO6: Muito bom: leitura pelo computador

              FATO7: Moderado : averiguacao pelo computador

              FATO8: Bom      : alterabilidade
----------------------------------------------------------
n29.4   - Portugues compac

      * DESCRICAO
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  42
REDE    : Metodo SSA                                data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n29.4   - Portugues compac

          Especifica um tipo de recurso para se
          expressar logica de processo.

          FUNCAO: q-quadro          TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n29
          RELACOES: bas
          SENTIDOS: d

      * ESPECIFICACAO

          IDENTIFICACAO: uso_porcom

             COMENTARIO: Descreve o uso adequado do
                         portugues compactado.

          TIPO: p-procedimento

          QTDE-FATOS:   8

              FATO1: Moderado : verificacao logica

              FATO2: Moderado : exibir estrutura logica

              FATO3: Bom      : simplicidade

              FATO4: Bom      : verificacao pelo usuario

              FATO5: Moderado : especificacao de programa

              FATO6: Muito bom: leitura pelo computador

              FATO7: Pobre    : averiguacao pelo computador

              FATO8: Bom      : alterabilidade
----------------------------------------------------------
n30     - Diag. aces imed

      * DESCRICAO

          Estabelece os acessos imediatos aos de-
          positos de dados e de que tipo serao.

          FUNCAO: r-rede            TIPO: a-ativo
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  43
REDE    : Metodo SSA                                 data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n30     - Diag. aces imed

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n12
         RELACOES: com
         SENTIDOS: d
----------------------------------------------------------
n31     - Ferramentas

      * DESCRICAO

          Identifica recursos automatizados para
          suporte ao desenvolvimento

          FUNCAO: r-rede            TIPO: a-ativo

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n12
         RELACOES: pos
         SENTIDOS: d

      * CONSTRUCAO-POSTERIOR

              QTDE:   4
               NOS: e(n31.1,n31.2,n31.3,n31.4)
         RELACOES: apo
         SENTIDOS: d
----------------------------------------------------------
n31.1   - FEGRES

      * DESCRICAO

          Ferramenta para apoio ao desenvolvi-
          mento de software.

          FUNCAO: r-rede            TIPO: f-fraco

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n31
```

```
==============================================================
RCQ-Metodos          BASE DE CONHECIMENTO            pag:  44
REDE    : Metodo SSA                                  data:
DOMINIO: Metodos de desenvolvimento de software     01/12/90
==============================================================
n31.1   - FEGRES

          RELACOES: apo
          SENTIDOS: o
--------------------------------------------------------------
n31.2   - TALISMA

      * DESCRICAO

          Ferramenta para apoio ao desenvolvi-
          mento de software.

          FUNCAO: r-rede            TIPO: f-fraco

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n31
          RELACOES: apo
          SENTIDOS: o
--------------------------------------------------------------
n31.3   - PC-DFD

      * DESCRICAO

          Ferramenta para apoio ao desenvolvi-
          mento de software.

          FUNCAO: r-rede            TIPO: f-fraco

          SEGMENTO: n-nao       CLASSIF.: n-nao

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n31
          RELACOES: apo
          SENTIDOS: o
--------------------------------------------------------------
n31.4   - THESEUS

      * DESCRICAO

          Ferramenta para apoio ao desenvolvi-
          mento de software.

          FUNCAO: r-rede            TIPO: f-fraco

          SEGMENTO: n-nao       CLASSIF.: n-nao
```

```
==========================================================
RCQ-Metodos          BASE DE CONHECIMENTO          pag:  45
REDE    : Metodo SSA                                 data:
DOMINIO: Metodos de desenvolvimento de software    01/12/90
==========================================================
n31.4   - THESEUS

      * CONSTRUCAO-ANTERIOR

              QTDE:   1
               NOS: n31
         RELACOES: apo
         SENTIDOS: o
----------------------------------------------------------
==========================================================
```

III.5 - CONCLUSÕES.

Apresentamos neste capítulo as características básicas e principais dos métodos de representação de conhecimento.

Não obstante aos demais métodos, a lógica de primeira ordem tem uma força expressiva bastante geral e uma semântica bem definida. Entretanto, devido sua linguagem de construção ser com variáveis atômicas e de não permitir facilidades adequadas para definição de construções mais complexas, a representação de certos domínios não é adequada quando se utiliza esse método, como ainda, há dificuldade de se entender o conhecimento expresso por ele. Também, as generalidades da lógica de primeira ordem tem tido uma barreira significante para o desenvolvimento de facilidades de dedução para utilizar o conhecimento representado.

Essas dificuldades motivaram o desenvolvimento de representações baseadas em Redes Semânticas e Quadros.

Uma representação em quadros permite um construtor da base de conhecimento desenvolver facilidades significantes descrevendo tipos de objetos do domínio que um sistema deve modelar.

Tendo em vista a dificuldade em se declarar como o conhecimento armazenado em quadros é para ser utilizado, tem-se utilizado as regras de produção para permitir a descrição de regras de inferências, regras para análise de decisão, ações que podem ser tomadas por vários agentes do