

UNIVERSIDADE ESTADUAL DE CAMPINAS

MODELO DE UTILIZAÇÃO DO PARADIGMA BLACKBOARD

Este exemplar corresponde à redação final da tese
defendida por GILBERTO SHIGEO NAKAMITI
e aprovada pela Comissão
Julgadora em 28 / 10 / 1991.

Orientador

DISSERTAÇÃO APRESENTADA À FACULDADE DE ENGENHARIA ELÉTRICA
DA UNICAMP COMO REQUISITO PARCIAL PARA A OBTENÇÃO
TÍTULO DE MESTRE EM ENGENHARIA ELÉTRICA.

GILBERTO SHIGUEO NAKAMITI

ORIENTADORA : PROFa. DRa. BEATRIZ MASCIA DALTRINI

CAMPINAS , OUTUBRO DE 1991.

UNICAMP

4697762

Aqueles que fazem da luta por seus ideais um ato de amor.

AGRADECIMENTOS

Especialmente à minha orientadora, Profa. Dra. Beatriz Mascia Daltrini pelo grande apoio, pela confiança, pela valiosa orientação e amizade;

Aos professores do Departamento de Engenharia de Computação e Automação Industrial, em especial aos professores Fernando Antônio Campos Gomide e Paulo César Bezerra, membros da banca de qualificação pelas observações e novas idéias para o trabalho, e ao professor Mário Jino pelo apoio e sugestões, especialmente nas fases iniciais do trabalho;

À professora Ariadne Maria Brito Rizzoni Carvalho, do Departamento de Ciência da Computação, pelo grande incentivo para que submetesse meus trabalhos para publicação;

A todos os amigos e companheiros. Em especial a Silvia R. Vergílio, Klaus e Elisa Schlünzen e Fani B. Camargo pelo apoio, amizade e paciência durante as diversas fases do curso;

Aos meus pais, sem dúvida, por todos esses anos, sempre um apoio fundamental e seguro;

À CAPES pelo imprescindível suporte financeiro;

E àqueles todos que de alguma forma contribuíram para a execução deste trabalho,

Meu muito, muito obrigado.

SUMÁRIO

Os sistemas Blackboard representam um tipo de arquitetura para resolução de problemas em Inteligência Artificial Distribuída relativamente recente, e caracterizam-se por seu comportamento dinâmico, cooperativo e oportunístico. Este trabalho apresenta o paradigma Blackboard e resultados obtidos nos principais centros de pesquisa do mundo.

Este trabalho propoe também o sistema ASTUTO - Arquitetura para a ReSolução de Problemas aTravés do *Uso* de Técnicas de Inteligência Artificial e Oportunismo. ASTUTO é um sistema que segue a filosofia Blackboard, e que propicia o uso desse paradigma para a resolução de problemas em diferentes domínios de aplicação. A abordagem utilizada privilegia a visão que as fontes de conhecimento têm do blackboard e o comportamento do sistema dirigido por eventos, aproximando-se daquilo que entendemos por essência do paradigma. O Modelo Entidade-Relacionamento é utilizado pela primeira vez para o modelamento e implementação da estrutura do blackboard. São propostas e implementadas fontes de conhecimento genéricas para sua manipulação, bem como o mecanismo de controle, heurísticas e o processo de reconfiguração do sistema.

ABSTRACT

Blackboard Systems are a relatively recent problem-solving framework in Distributed Artificial Intelligence. They are well-known for their cooperative, opportunistic, dynamic behavior. This work presents the Blackboard paradigm and a survey of world research results.

This work also proposes ASTUTO - Architecture for Problem Solving Through the Use of Artificial Intelligence Techniques and Opportunism. ASTUTO follows the Blackboard paradigm, and permits its use for problem-solving in different application domains. Its approach favors the knowledge sources' view of the blackboard, and an event-driven behavior, getting close to what we understand as the paradigm essence. The Entity-Relationship Model is for the first time used to model and implement the blackboard structure. Generic knowledge sources are proposed and implemented for the blackboard handling, as well as the control mechanism, heuristics and the system reconfiguration process.

CONTEÚDO

1	INTRODUÇÃO	01
2	SISTEMAS BLACKBOARD	08
2.1	O MODELO BLACKBOARD	08
2.2	HISTÓRICO	10
2.3	ARQUITETURA	13
2.3.1	FONTES DE CONHECIMENTO	14
2.3.2	BLACKBOARD	16
2.3.3	CONTROLE	17
2.4	ESTRATÉGIAS	19
2.5	EXEMPLO DE OPERAÇÃO	20
2.5.1	ENCONTRANDO COALAS	21
2.6	EXEMPLOS DE SISTEMAS	23
2.6.1	HEARSAY-II	23
2.6.2	HASP	26
2.6.3	CRYSALIS	31
2.6.4	DICE	35
2.7	ABORDAGENS	39
2.7.1	BLACKBOARDS EM AMBIENTES DISTRIBUÍDOS.....	39
2.7.2	BLACKBOARDS TRANSACIONAIS	41
2.7.3	BLACKBOARDS ORIENTADOS A OBJETOS	44
2.8	CONCLUSÃO	46
3	LINGUAGENS PARA IMPLEMENTAÇÃO	53
3.1	LISP	53
3.2	PROLOG	56
3.3	OPS5	58
3.4	CONSIDERAÇÕES	60

4	O SISTEMA ASTUTO	65
4.1	INTRODUÇÃO	65
4.2	O MODELO ENTIDADE-RELACIONAMENTO	66
4.3	ABORDAGEM QUANTO AO COMPORTAMENTO	70
4.4	ESPECIFICAÇÃO DO SISTEMA	72
4.5	O BLACKBOARD EM ASTUTO	73
4.6	OPERAÇÃO SOBRE O BLACKBOARD	73
4.7	ESPECIFICAÇÃO DO SISTEMA	78
4.7.1	DEFINIÇÃO DOS MÓDULOS	78
4.7.2	TABELAS DE DESCRIÇÃO DE DADOS	84
4.7.3	ARQUITETURA DE CONTROLE	101
4.7.4	ASTUTO EM PDL	102
5	EXEMPLO DE APLICAÇÃO	112
5.1	INTRODUÇÃO	112
5.2	O PROBLEMA	112
5.3	AS FONTES DE CONHECIMENTO	115
5.4	CONHECIMENTO SOBRE O DOMÍNIO	122
5.5	GERAÇÃO DE APLICAÇÕES	124
6	CONCLUSÃO	132

LISTA DE FIGURAS

2.1	UM EXEMPLO DE ARQUITETURA BLACKBOARD	14
2.2	ESTRUTURA DO BLACKBOARD DO SISTEMA HASP/SIAP	27
2.3	PANELS DO SISTEMA CRYSLIS	32
2.4	CONFIGURAÇÃO DO SISTEMA CRYSLIS	34
2.5	VISÃO CONCEITUAL DO SISTEMA DICE	36
2.6	ORGANIZAÇÃO DOS NODOS	40
2.7a	REDE DE NODOS DE PROCESSAMENTO	42
2.7b	CONTROLE E FONTES DE CONHECIMENTO EM UM NODO	43
4.1	EXEMPLO DE MODELAGEM	67
4.2	EXEMPLO DE AGREGAÇÃO	68
4.3	EXEMPLO DE GENERALIZAÇÃO	68
4.4	OUTRO EXEMPLO DE GENERALIZAÇÃO	69
4.5	ESQUEMA CONCEITUAL DO SISTEMA ASTUTO	71
4.6	ILUSTRAÇÃO DE RELACIONAMENTO	74
4.7	GERAÇÃO DE SOLUÇÕES ALTERNATIVAS	75
4.8	CLASSIFICAÇÃO DE ENTIDADES	75
4.9	DIVISÃO DE RELACIONAMENTO	76
4.10	GERAÇÃO DE ENTIDADE	77
4.11	GERAÇÃO DE RELACIONAMENTO	77
4.12	GERAÇÃO DE HIERARQUIA	78
4.13	SADT DE PRIMEIRO NÍVEL - <i>MÓDULO PRINCIPAL</i>	79
4.14	SADT DE SEGUNDO NÍVEL - <i>INICIALIZAR SISTEMA</i>	81
4.15	SADT DE SEGUNDO NÍVEL - <i>REALIZAR RESOLUÇÃO PROBLEMA</i> ..	82
4.16	SADT DE SEGUNDO NÍVEL- <i>INFORMAR RESULTADO PROCESSAMENTO</i>	83
4.17	RELAÇÃO CONTROLE- ENTRADA- SAÍDA - 01	85
4.18	RELAÇÃO CONTROLE- ENTRADA- SAÍDA - 02	86
4.19	RELAÇÃO CONTROLE- ENTRADA- SAÍDA - 03	87
4.20	RELAÇÃO CONTROLE- ENTRADA- SAÍDA - 04	88
4.21	DESCRIÇÃO DOS FLUXOS DE DADOS E DE CONTROLE - 01	89
4.22	DESCRIÇÃO DOS FLUXOS DE DADOS E DE CONTROLE - 02	90
4.23	DESCRIÇÃO DOS FLUXOS DE DADOS E DE CONTROLE - 03	91
4.24	DESCRIÇÃO DOS FLUXOS DE DADOS E DE CONTROLE - 04	92
4.25	DESCRIÇÃO DOS FLUXOS DE DADOS E DE CONTROLE - 05	93

4.26	DESCRIÇÃO DOS FLUXOS DE DADOS E DE CONTROLE - 06		94
4.27	DESCRIÇÃO DOS FLUXOS DE DADOS E DE CONTROLE - 07		95
4.28	DESCRIÇÃO DOS FLUXOS DE DADOS E DE CONTROLE - 08		96
4.29	DESCRIÇÃO DOS FLUXOS DE DADOS E DE CONTROLE - 09		97
4.30	DESCRIÇÃO DOS FLUXOS DE DADOS E DE CONTROLE - 10		98
4.31	DESCRIÇÃO DOS FLUXOS DE DADOS E DE CONTROLE - 11		99
4.32	DESCRIÇÃO DOS FLUXOS DE DADOS E DE CONTROLE - 12		100
4.33	ARQUITETURA DE CONTROLE DO SISTEMA ASTUTO		101
5.1a	MAPA PARCIAL DA CIDADE DE ANDRADINA		113
5.1b	LISTA DOS PRINCIPAIS ESTABELECIMENTOS DA CIDADE		114
5.2	EXEMPLO DE PERCURSO ALTERNATIVO		115
5.3	EXEMPLO DE CLASSIFICAÇÃO DE ESTABELECIMENTO		116
5.4	EXEMPLO DE DIVISÃO DE RELACIONAMENTO		118
5.5	EXEMPLO DE GERAÇÃO DE ESTABELECIMENTO		119
5.6	EXEMPLO DE GERAÇÃO DE RELACIONAMENTO		120
5.7	EXEMPLO DE GERAÇÃO DE HIERARQUIA		121
5.8	EXEMPLOS DE CAMINHOS		124

LISTA DE TRABALHOS CORRELATOS

[NAKA-91a] Nakamiti G.S., *ASTUTO - Uma Arquitetura para o Desenvolvimento de Sistemas Blackboard em Prolog*, Anais da XVII Conferência Latino-Americana de Informática, Caracas, Venezuela, Julho de 1991.

[NAKA-91b] Nakamiti G.S., Daltrini B.M., *Especificação e Implementação do Sistema ASTUTO*, Relatório Técnico RT-DCA 018/91, Departamento de Engenharia de Computação e Automação Industrial, Unicamp, Junho de 1991.

[NAKA-91c] Nakamiti G.S., *Blackboard - Uma Visão do Modelo e da Arquitetura*, submetido para publicação.

[NAKA-91d] Nakamiti G.S., *ASTUTO - Modelling Blackboards Through an Entity-Relationship Approach*, submetido para publicação.

CAPÍTULO 1 - INTRODUÇÃO

A Engenharia de Software preocupa-se em assegurar a qualidade do software produzido através da integração de métodos, ferramentas e procedimentos para o seu desenvolvimento [PRES-87]. Ambientes de Desenvolvimento de Software [CHAR-86] podem ser utilizados para a evolução racional, ordenada e administrável do software. Esses ambientes são compostos por um grande número de ferramentas que devem atuar cooperativamente. As ferramentas utilizadas podem ser de várias naturezas e, em particular, fazerem parte dos chamados "Software para Inteligência Artificial" [PRES-87], que vêm sendo objeto de muita pesquisa.

A Inteligência Artificial (IA) vincula-se ao projeto de sistemas inteligentes de computação, isto é, sistemas que apresentam características que normalmente associamos à inteligência no comportamento humano, como compreensão da linguagem, aprendizagem, raciocínio e resolução de problemas [GOLD-86].

A maior parte da pesquisa em Inteligência Artificial investiga como um único "agente" (e.g. em [KATZ-89]) pode apresentar tal tipo de comportamento na resolução de problemas, por exemplo. No entanto, com a constatação de que muitos problemas e atividades humanas envolvem grupos de pessoas, com o desenvolvimento de computadores concorrentes mais poderosos, e com a proliferação das redes de computadores, aumenta o interesse no estudo da concorrência e distribuição (de recursos e tarefas) em Inteligência Artificial.

A Inteligência Artificial Distribuída é o ramo da Inteligência Artificial que ocupa-se das questões de concorrência. Lida com problemas como o da distribuição e

coordenação de conhecimento e ações na resolução distribuída de problemas. Resultados obtidos em Inteligência Artificial Distribuída tornam-se importantes para a pesquisa em outras áreas, tais como bancos de dados distribuídos, computação paralela e distribuída, interação homem-máquina, trabalho cooperativo auxiliado por computador, na ciência cognitiva (como modelos mentais e cognição social), e na pesquisa básica de Inteligência Artificial (representação de problemas, raciocínio cooperativo, resolução de problemas).

Os sistemas Blackboard representam um tipo de arquitetura para resolução de problemas em Inteligência Artificial Distribuída relativamente recente, e que vem sendo constantemente estendida. Em [LUCE-89], por exemplo, propõe-se que a arquitetura interna de software de seu Ambiente de Desenvolvimento de Software siga o modelo Blackboard servindo, portanto, como mais uma ferramenta para atingir os objetivos da Engenharia de Software. Os sistemas Blackboard constituem-se em uma área de pesquisa de ponta, com muitas oportunidades, tanto em pesquisa básica quanto aplicada.

Este trabalho representa o primeiro esforço dentro do grupo de pesquisa na área de sistemas Blackboard, onde o objetivo inicial de construir um sistema Blackboard para um Ambiente de Desenvolvimento de Software foi substituído pelo objetivo de construir um núcleo Blackboard que pudesse ser utilizado em várias aplicações. O trabalho apresenta os resultados obtidos nos principais centros de pesquisa do mundo, e fornece também uma base para futuros trabalhos nesta promissora área. Por tratar-se de um trabalho pioneiro, foi dada bastante ênfase na apresentação do modelo e da arquitetura Blackboard. Objetiva-se, com isto, fornecer uma melhor noção sobre o tema, suas características, operacionalidade e aplicabilidade.

Outro objetivo deste trabalho foi a proposta de uma arquitetura Blackboard que resgatasse alguns conceitos fundamentais do modelo, mas que têm sido transgredidos em vários sistemas, na forma discutida em [NII -88c].

Além disso, objetivou-se a criação de um núcleo que contivesse as funções básicas para permitir a execução de diferentes sistemas Blackboard, para diferentes domínios de aplicação, através de sua reconfiguração. O núcleo deveria apresentar funções para o controle de execução dos sistemas segundo o paradigma Blackboard, e também funções para a manipulação dos dados e do conhecimento envolvidos em cada aplicação. O sistema deveria também permitir uma melhor compreensão do domínio da aplicação, obtida através de seu uso, sendo também útil, portanto, como um gerador de protótipos para refinamento de requisitos de software, como citado, por exemplo, em [PRESS-87].

O desenvolvimento deste trabalho foi marcado pela presença constante de novas publicações sobre sistemas Blackboard nos periódicos e revistas científicas internacionais. A observação dessas publicações acarretou uma grande evolução e amadurecimento do trabalho. A presença de tantas novas pesquisas na área mostra sua pujança e seu grande potencial como arquitetura genérica para resolução de problemas utilizando Inteligência Artificial.

No Brasil, pouco existe sobre sistemas Blackboard. Não acredito, entretanto, que isto tenha limitado a qualidade deste trabalho. Criou, sim, fortes motivações e a busca constante do "estado da arte" através de publicações internacionais. Foi bastante grande o esforço na pesquisa bibliográfica, tendo sido visitadas várias bibliotecas universitárias, em centros de pesquisa e também em empresas privadas. A pesquisa bibliográfica como fator de mudanças neste trabalho só terminou após encerrada a especificação do projeto, no primeiro semestre de 1990.

Assim, após conhecer o modelo Blackboard e estar consciente de suas qualidades e potencial, o objetivo passou a ser o de concretizá-lo. Características conceituais importantes do modelo Blackboard vêm sendo transgredidas em vários sistemas (cf. [NII-88c]) para aumentar a eficiência em determinadas aplicações. Com isso, o comportamento dos sistemas passa a ser semelhante àquele

obtido através de outras arquiteturas de resolução de problemas já conhecidas, como nos sistemas especialistas. Durante as etapas iniciais do projeto, denominado ASTUTO - Arquitetura para a ReSolução de Problemas ATravés do Uso de Técnicas de IA e Oportunismo [NAKA-91a, NAKA-91b], a maior preocupação foi a manter as características essenciais do modelo Blackboard, que permitem seu comportamento dinâmico, cooperativo e oportunístico.

De posse do modelo de funcionamento, foi necessário um estudo para identificar uma metodologia que pudesse representá-lo adequadamente. A metodologia escolhida foi SADT - Structured Analysis and Design Technique [SOFT-76]. Infelizmente não foi obtido um software do SADT, o que propiciaria mecanismos de especificação e validação automatizados e provavelmente mais abrangentes para o projeto. Desta forma, os diagramas e tabelas do SADT tiveram que ser realizados quase artesanalmente.

O modelo hierárquico de representação do blackboard existente nos sistemas presentes na literatura foi também repensado. Pela generalidade e flexibilidade da arquitetura Blackboard, deveria existir um modelo de representação que também refletisse essas características.

Também a linguagem de programação escolhida deveria ser capaz de descrever as estruturas modeladas de forma simples e direta. ASTUTO foi implementado em micro-computadores IBM-PC compatíveis. A implementação de ASTUTO já estava em fase bastante adiantada quando ficaram disponíveis as estações de trabalho. Além disso, em seu período inicial de funcionamento, não estava disponível nenhum compilador PROLOG, a linguagem de programação escolhida, e não havia prazo determinado para sua instalação. Decidiu-se então que não seria viável abandonar o trabalho já realizado e aguardar por tempo indeterminado para desenvolver o projeto no novo equipamento, embora pudesse oferecer possibilidades muito interessantes.

ASTUTO foi desenvolvido para poder ser reconfigurado e utilizado em inúmeros domínios de aplicação. Houve então a

preocupação de gerar um exemplo de aplicação bastante simples, para que pudesse ser facilmente entendido, e ao mesmo tempo bastante abrangente, para que pudesse proporcionar ao leitor uma vasta gama de aplicações possíveis.

Os documentos gerados nas diversas fases de desenvolvimento de ASTUTO encontram-se neste trabalho, para facilitar seu entendimento e possibilitar sua manutenção e extensão.

No Capítulo 2, são introduzidos o modelo e a arquitetura Blackboard para resolução de problemas. No Capítulo 3, são mostradas algumas abordagens, representadas por linguagens de programação, que podem viabilizar a implementação de um sistema deste tipo. O sistema ASTUTO é apresentado no Capítulo 4, e no capítulo seguinte, mostra-se uma possível aplicação, para ilustrar o uso do sistema. Finalmente, no Capítulo 6 são apresentadas as conclusões deste trabalho.

REFERÊNCIAS

- [CHAR-86] Charette R.N., "Software Engineering Environments", McGraw Hill, 1986.
- [CHEN-76] Chen P.P., *The Entity-Relationship Model: Toward a Unified View of Data*, ACM Transactions on Database Systems, vol.01, no.01, 1976.
- [GOLD-86] Goldszajn M., Carnota R., *Inteligência Artificial Aplicada - Lógica y Representación del Conocimiento*, EBAI - Editora da Unicamp, 1986.
- [KATZ-89] Katz M.J, Rosenschein J.S., *Plans for Multiple Agents*, Research Notes in Distributed Artificial Intelligence - volume II, Pitman Publishing, pp.197-228, 1989.
- [LUCE-89] Lucena C.J., Takahashi E.T, *A Anatomia de um Ambiente de Desenvolvimento de Software*, V Reunião de Trabalho - Coletânea de Resultados de Pesquisas - Volume I, Projeto Estra, Outubro 1988-Abril 1989, pp. 1-46.
- [NAKA-91a] Nakamiti G.S., *ASTUTO - Uma Arquitetura para o Desenvolvimento de Sistemas Blackboard em Prolog*, Anais da XVII Conferência Latino-Americana de Informática, Caracas, Venezuela, Julho de 1991.
- [NAKA-91b] Nakamiti G.S., Daltrini B.M., *Especificação e Implementação do Sistema ASTUTO*, Relatório Técnico RT-DCA 018/91, Departamento de Engenharia de Computação e Automação Industrial, Faculdade de Engenharia Elétrica, Unicamp, Junho de 1991.
- [NII -88c] Nii H.P., Aiello N., Rice J., *Frameworks for Concurrent Problem Solving: A Report on CAGE and POLIGON*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 475-501.

[PRES-87] Pressman R.S., *Software Engineering - A Practitioner's Approach*, McGraw-Hill Book Co., 1987.

[SOFT-76] SofTech Inc., *An Introduction to SADT*, Manual, 1976.

CAPÍTULO 2 - SISTEMAS BLACKBOARD

Blackboard é um paradigma de resolução de problemas em Inteligência Artificial, capaz de utilizar várias fontes de conhecimento de naturezas diferentes para solucionar um problema.

Neste capítulo, é apresentado o modelo Blackboard, a generalidade de sua aplicação, e diferentes abordagens na utilização do modelo.

2.1 O MODELO BLACKBOARD

A maior parte das estruturas e implementações de máquinas de inferência em Inteligência Artificial não oferecem muita flexibilidade. São fortemente restritas a procedimentos de busca de soluções, como forward e backward. A resolução de problemas reais exige flexibilidade e generalidade. O **modelo Blackboard** [NAKA-91c, ENGE-88a] é uma abordagem que surgiu a fim de prover essas características.

Trata-se de uma idéia simples e poderosa de se lidar com problemas que necessitam manipular conhecimento e dados incertos, aplicando estratégias não determinísticas para sua solução. Sua aplicação cobre domínios dos mais diversos, como visão, planejamento, determinação de estruturas de proteína, diagnose médica e reconhecimento de voz.

Na **arquitetura Blackboard**, diversas fontes de conhecimento (Knowledge Sources - KSs) podem cooperar na formação e modificação do estado da solução do problema. O processo de formação da

solução é incremental, sendo que ela e as estratégias para a sua formação podem ser avaliadas, trocadas ou abandonadas a qualquer momento. Além disso, as fontes de conhecimento contribuem oportunisticamente, pois têm acesso permanente ao estado da solução, podendo alterá-la no momento mais "oportuno".

O **modelo Blackboard** pode ser visto como uma evolução dos sistemas especialistas clássicos, onde o conhecimento é segmentado em módulos, e a cada módulo é associada uma máquina de inferência. A comunicação entre os módulos se dá através de uma memória de trabalho comum, denominada **blackboard**.

Este modelo é composto pelas fontes de conhecimento e pelo blackboard propriamente dito, que nada mais é que uma estrutura de dados. Desta forma, não há qualquer componente de controle especificado nele. O modelo apenas determina a organização do domínio do conhecimento, as entradas e as soluções parciais necessárias para a solução do problema. O conhecimento a ser aplicado é baseado no estado da solução. As fontes de conhecimento são auto-ativadas quando puderem contribuir em sua solução, dinamicamente.

O **modelo Blackboard** é uma entidade conceitual, e não uma especificação computacional [ENGE-88a]. Ele apenas provê direções suficientes para esboçar a solução de um determinado problema. Para existir um sistema, é necessária uma especificação mais detalhada.

Para ilustrar a idéia, pode-se considerar a montagem de um quebra-cabeças por muitas pessoas. Cada uma delas vê o estado da solução e, de acordo com as peças que tiver, pode auxiliar na solução incremental do problema. Não existe ordem a priori para isto acontecer. As pessoas agem paralela e simultaneamente. Caso nem todos possam colocar suas peças no instante em que desejarem, é necessário haver um esquema que eles respeitem para evitar confusão. Este esquema é muito importante pois pode, por exemplo, forçar que o quebra-cabeças seja montado de cima para baixo, violando uma das características do modelo, a da construção

oportunistica da solução.

Outro ponto importante é determinar o que é uma solução para o problema, que tipo de informação os dados contêm, o que pode ser inferido deles, e quais conhecimentos levarão à solução do problema. Normalmente, uma solução pode ser desmembrada em partes, que podem ser apenas hipotéticas quando tratamos de conhecimento impreciso. Um dos objetivos é manter as partes mais confiáveis e promissoras em um espaço de soluções. A solução final depende das contribuições individuais de cada especialista. Seu conhecimento é logicamente independente e eles devem atuar cooperativamente. A cooperação é obtida assumindo que qualquer informação necessária será fornecida por algum deles.

2.2 HISTÓRICO

Allen Newell, em 1962, citou pela primeira vez o termo "*blackboard*" na literatura de Inteligência Artificial [ENGE-88a]:

"Metaforicamente, podemos pensar em um conjunto de trabalhadores, todos olhando para um mesmo quadro-negro ("*blackboard*") : cada um é capaz de ler o que está escrito nele, e julgar se tem algo que valha a pena lhe adicionar".

O primeiro trabalho experimental foi o projeto **HEARSAY-II** [ENGE-88g, ERMA-88a, LESS-88a], no início da década de 1970. Seu intuito era o de reconhecer e produzir sons da fala humana.

Na segunda metade da década, começaram a surgir outros sistemas [ENGE-88b], onde o que mais se destaca é o **HASP** [NII -88b]. Sua função era a de interpretar sinais contínuos de sonares oceânicos para detectar a presença de submarinos inimigos. As principais idéias do modelo Blackboard que motivaram sua adoção foram a de juntar soluções incertas e parciais para construir soluções melhores, e a estratégia "*island driving*" [ENGE-88b]. Segundo essa estratégia de resolução de problemas, soluções

parciais relativamente seguras são exploradas, dando origem a soluções mais complexas e mais seguras.

A proposta de solução de Newell levou ao sistema **OPM** [HAYE-79], onde as sub-rotinas, que podem ser vistas como fontes de conhecimento, são representadas por regras de produção (do tipo condição-ação). Trata-se de um sistema de simulação e planejamento resultante dos sistemas HEARSAY-II e HASP.

Ainda nesta época, surgiu o sistema **CRYSLIS** [TERR-83] para análise de estrutura de proteínas. Este sistema foi importante por introduzir duas estruturas de blackboard ao invés de uma como nos projetos anteriores. Houve também o sistema **VISIONS** [HANS-87] para interpretação de imagens estáticas coloridas de cenas naturais. Outro sistema para processamento de imagens foi desenvolvido no Japão - o **SACAP** (Structural Analysis of Complex Aerial Photographs) [NAGA-88].

A partir do final da década de 1970 começaram a surgir arquiteturas mais genéricas [ENGE-88c], que se constituiriam em ferramentas para auxiliar no desenvolvimento de aplicações. A primeira delas foi o sistema **AGE** (Attempt to Generalize) [NII-88a], que aproveitou-se do conhecimento obtido no desenvolvimento dos sistemas HASP e CRYSLIS. Este sistema permite a construção de uma classe restrita de sistemas blackboard devido a restrições de projeto. Baseado no **AGE**, foi desenvolvido o **ESHELL**, citado em [ENGE-88c], o mais popular no Japão. Seguiu-se o **HEARSAY-III** [ERMA-88b], uma evolução do Hearsay-II, desta vez com independência do domínio de conhecimento. Seu objetivo era servir como veículo para explorar os princípios de resolução de problemas. O sistema **BB1** [HAYE-88a] foi uma generalização que priorizou a estrutura de controle. Utiliza-se de uma arquitetura Blackboard para controlar a execução das fontes de conhecimento. De sua influência, surgiu o sistema **ERASMUS** [JAGA-87]. O sistema **MXA** [TAIL-88] para reconhecimento de sinais foi originalmente inspirado no HASP, tendo recebido influência do HEARSAY-II na fase de projeto. No entanto, permite facilidades para o desenvolvimento de outras aplicações.

BLOBS [ZANC-88] é um sistema que dá ênfase aos aspectos temporais, permitindo o descarte de informações que não mais interessam. Permite também a modificação do modelo Blackboard, restringindo o acesso aos dados para simplificar seu gerenciamento durante um certo período. Trata-se de uma mistura do conceito de Blackboard com o conceito de estruturas de dados orientadas a objetos.

O sistema **DVMT** (Distributed Vehicle Monitoring Testbed) [LESS-88b] foi baseado no sistema Hearsay-II. É uma ferramenta que leva redes distribuídas de nodos semi-autônomos a trabalharem juntas na solução de um único problema [ENGE-88d].

As generalizações auxiliaram no desenvolvimento de uma série de aplicações na década de 1980.

A partir do AGE, surgiu um outro sistema para reconhecimento de sinais, o **TRICERO** [WILL-88]. Esse sistema foi uma tentativa de modelar uma hierarquia de especialistas usando vários sistemas Blackboard. Do sistema BB1, derivou-se o **PROTEAN** [HAYE-88b] para a determinação da estrutura tri-dimensional de moléculas de proteína através da informação de experimentos de ressonância magnética.

A partir de 1985, têm surgido sistemas que estendem e refinam a arquitetura Blackboard [ENGE-88e]. **CAGE** e **POLIGON** [NII -88c, RICE-89] trazem propostas para lidar com concorrência, uma característica intrínseca ao modelo Blackboard.

Dada a preocupação de prover características de eficiência e estruturação, surgiu o sistema **GBB** (Generic Blackboard) [CORK-86]. Sob sua perspectiva, a mesma estrutura blackboard pode ser usada com diferentes mecanismos de controle.

O sistema **MUSE** [REYN-88] foi desenvolvido para microprocessadores de 16/32 bits e implementado em POP-2 (desenvolvido em Edinburgh) e Smalltalk. Visa aplicações contínuas, em tempo real.

Houve também um sistema implementado em Prolog - o **EPBS** (Edinburgh Prolog Blackboard Shell) [JONE-88]. Trata-se de um sistema com características diferentes daqueles vistos até aqui. As estruturas do blackboard são cláusulas da linguagem e o sistema não faz distinção entre regras de fontes de conhecimento diferentes.

O sistema **BB*** [HAYE-88c] foi desenvolvido com o objetivo de obter arquiteturas Blackboard mais poderosas. Trata-se de um projeto para adequar o sistema **BB1** para certas classes de aplicação.

Pesquisas recentes do MIT levaram ao desenvolvimento do sistema **DICE** [SRIR-89], que consiste de um ambiente distribuído que facilita a comunicação, coordenação e controle durante o desenvolvimento de projetos de produtos de engenharia.

2.3 ARQUITETURA BLACKBOARD

O paralelismo é fortemente associado ao modelo Blackboard [ENGE-88a]. Já que a maioria dos ambientes computacionais atuais é serial, criou-se uma **arquitetura Blackboard** para prover os recursos necessários para o projeto de sistemas Blackboard. Essa arquitetura contém descrições dos componentes básicos de um sistema Blackboard baseado nesse tipo de ambiente.

Na Figura 2.1, extraída de [ENGE-88a], é mostrado um exemplo de arquitetura de um sistema Blackboard. Nela, os dados no blackboard são organizados hierarquicamente. As fontes de conhecimento (KSs) são logicamente independentes e auto-ativadas. Somente elas podem alterar o blackboard. De acordo com as últimas alterações na informação do blackboard, o módulo de controle seleciona a próxima fonte de conhecimento para execução.

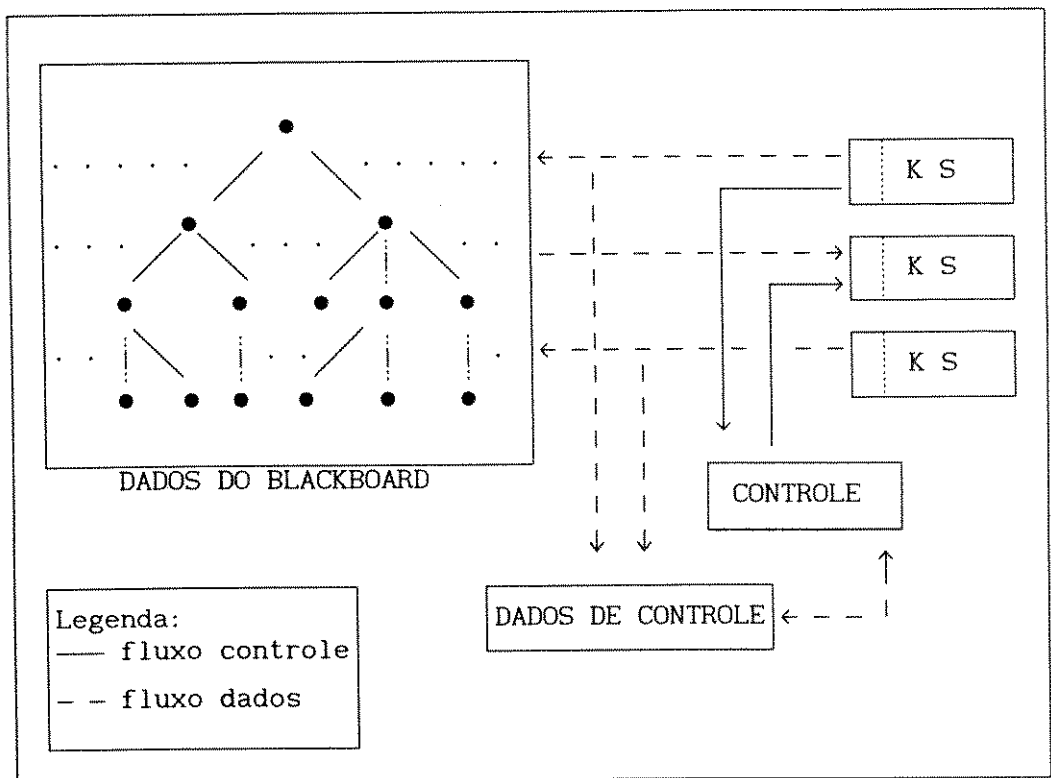


Figura 2.1 - Um Exemplo de Arquitetura Blackboard

2.3.1 FONTES DE CONHECIMENTO

O domínio do conhecimento necessário para a solução do problema é particionado entre fontes de conhecimento, que são mantidas separadas e independentes, com o intuito de que levem à solução através da informação provida.

Elas podem modificar somente o blackboard e as estruturas de controle (que também poderão estar no blackboard). O blackboard só é modificado pelas fontes de conhecimento, e todas as modificações no espaço de soluções são explícitas e visíveis.

Na maioria dos sistemas, as fontes de conhecimento são auto-ativadas. Neste caso, devem conhecer as condições nas quais poderão contribuir para o problema [ENGE-88a].

Outra possibilidade é não haver pré-condições para ativação, como no sistema CRYSLIS. Neste caso, os módulos de controle são representados por fontes de conhecimento, que executam a seleção para ativação das fontes.

As fontes de conhecimento podem ser representadas por conjuntos de regras de produção, como no sistema AGE [NII -88a]. No sistema SACAP, cada fonte de conhecimento é representada por uma única regra, existindo assim várias regras (KSs) para o reconhecimento de um tipo específico de elemento do espaço de soluções.

As fontes de conhecimento geram hipóteses utilizando regras que:

- a) adicionam ou modificam elementos da hipótese; ou
- b) adicionam ou modificam relações entre elementos.

A forma como as regras são organizadas numa fonte de conhecimento e quais regras estarão presentes nela, dependem de sua função no plano global de solução do problema. Por exemplo, uma fonte de conhecimento poderia ser organizada para representar modelos, onde as regras seriam agrupadas em função dos objetos ou conceitos, de forma similar a "*frames*". Uma fonte de conhecimento poderia ainda ser organizada em função de eventos, onde funções similares seriam agrupadas.

Dois pontos referentes às fontes de conhecimento que devem ser tratados com cuidado são: resolução de conflitos entre fontes de conhecimento e granularidade de ações das fontes. O sistema deve saber como agir quando duas ou mais fontes querem alterar um mesmo elemento da solução de formas diferentes, e ao mesmo tempo deve-se garantir que cada fonte de conhecimento altere um único elemento da solução.

Em qualquer caso, as fontes de conhecimento devem depositar suas inferências no blackboard.

2.3.2 BLACKBOARD

O blackboard nada mais é que uma estrutura, uma base de dados consistindo de objetos (elementos) do espaço de soluções. Seu objetivo é servir como um meio através do qual as fontes de conhecimento interagem e conduzem à formação incremental da solução do problema. Constitui-se de dados de entrada, soluções parciais, alternativas, soluções finais e opcionalmente, dados de controle [ENGE-88a].

Os elementos que constituem as soluções são organizados hierarquicamente e mantêm relações entre eles. O blackboard também pode ser particionado em seções, hierarquicamente. Os elementos do espaço de soluções de cada nível podem ser descritos como formados da abstração da informação dos elementos do nível inferior. Cada elemento do blackboard representa uma hipótese. Estas hipóteses podem conter atributos, como por exemplo, seu nível de confiança.

No caso de se lidar com fontes de conhecimento que manipulem conhecimento incerto, o blackboard pode conter várias hipóteses alternativas.

É possível também considerar como solução a informação espalhada por todo o blackboard e não apenas no nível superior, o que é feito no sistema HASP. Nele, a rede de soluções parciais existente no blackboard é considerada uma solução aceitável para o problema.

Uma outra opção de organização é manter mais de uma estrutura de blackboard ("*panel*"), como no sistema CRYALIS. Nesse sistema, há dois "*panels*". A informação do primeiro é produzida antes do início do processo de interpretação. As soluções parciais são armazenadas no outro "*panel*", derivadas de dados do primeiro. Um dos "*panels*" pode inclusive conter informações de controle e arbitrar a invocação das fontes de conhecimento.

2.3.3 CONTROLE

Dada a restrição de processamento serial, é necessário haver um módulo de controle para monitorar o acesso ao blackboard, e também para decidir os passos a seguir.

A ação de uma fonte de conhecimento altera o blackboard, sendo criada uma informação de controle. Através dela, cada uma das fontes indica sua possível contribuição. A partir disso, o módulo de controle decide qual caminho seguir.

Para o controle de um sistema Blackboard, podem existir várias categorias de decisão [HAYE-85]:

a) Decisões de problemas.

Uma decisão de problema define o problema que o sistema resolverá. Esta decisão guia todo o processo de resolução.

b) Decisões de estratégias.

As decisões de estratégias estabelecem planos gerais e sequenciais de resolução de problemas. Elas podem ocorrer em qualquer ponto do processo, substituindo a estratégia anterior ou fazendo com que operem simultaneamente;

c) Decisões de foco de atenção.

Estabelecem objetivos locais para execução das fontes de conhecimento. As decisões de foco de atenção são temporárias e podem influenciar decisões de "*scheduling*";

d) Decisões políticas.

Estabelecem critérios de "*scheduling*" global, favorecendo fontes de conhecimento com determinados atributos;

e) Decisões de tarefas a serem cumpridas.

Identificam as fontes de conhecimento pendentes em cada ciclo de processamento, e gerenciam uma lista de

fontes de conhecimento prontas para executar, efetuando a resolução de conflitos.

f) Decisões de ações.

Identificam as fontes de conhecimento escolhidas para serem executadas em cada ciclo de processamento.

Para um controle inteligente do processo de resolução, alguns critérios são propostos em [HAYE-85]:

- a) tornar explícitas as decisões de controle;
- b) decidir quais ações executar, conciliando decisões sobre quais ações são desejáveis e quais são factíveis;
- c) adotar heurísticas de controle de granularidades diferentes.
- d) adotar heurísticas de controle que privilegiem atributos de ação úteis para a situação atual do problema;
- e) adotar, manter e descartar heurísticas de controle, de acordo com situações dinâmicas na solução do problema;
- f) decidir como integrar várias heurísticas de controle;
- g) planejar dinamicamente sequências de ações;
- h) ponderar as prioridades relativas das ações.

O controle pode resumir-se a um escalonador síncrono ou ser composto por uma coleção de fontes de conhecimento, como no sistema CRYVALIS.

Deve haver critérios para que os módulos de controle reconheçam quando o processo chegou ao fim. Normalmente, esta tarefa é incumbida a uma fonte de conhecimento, que deve reconhecer se uma solução satisfatória foi encontrada, ou se o sistema não pode continuar por falta de conhecimento ou dados [NII-86a].

2.4 ESTRATÉGIAS

O comportamento que um sistema construído sob a arquitetura *Blackboard* assume para chegar à solução de um problema depende das estratégias que este utiliza, codificadas na forma de mecanismos de controle e pré-condições das fontes de conhecimento [NII -86a]. Basicamente, as estratégias utilizadas por um sistema determinam como ele deve escolher a região da solução que deve ser alterada em um dado instante, e que fonte de conhecimento deve ser executada sobre esta região.

Sendo o blackboard organizado hierarquicamente em níveis, uma fonte de conhecimento utiliza informações de um nível como "entrada", e produz como "saída", informações para um outro nível. Existem duas estratégias clássicas para a construção de um elemento ou solução parcial de um nível a partir de elementos de outros níveis:

bottom-up: por esta estratégia, um elemento de um nível é criado a partir da combinação de vários elementos do nível inferior;

top-down: esta estratégia utiliza um modelo da solução ou um elemento da solução de um nível alto para derivar elementos de nível inferior.

Cada fonte de conhecimento normalmente é direcionada a utilizar uma estratégia, e o mecanismo de controle deve ser capaz de reconhecer qual das estratégias é a mais apropriada em um determinado ponto da construção da solução, escolhendo uma das fontes de conhecimento que a utilize.

Existem ainda outras linhas de conduta que podem conduzir a escolha das fontes de conhecimento durante a execução do sistema, e que são inerentes ao tipo de restrições contidas na pré-condição de cada uma das fontes de conhecimento.

São basicamente duas as estratégias utilizadas pelas fontes de conhecimento para determinar sua auto-execução. Uma delas consiste em analisar o estado atual da solução e verificar se a fonte de conhecimento pode acrescentar algo a ela. A outra consiste em deixar a ativação da fonte de conhecimento dependente de fatores externos ao estado da solução, como por exemplo: subordinar uma fonte de conhecimento à execução prévia de uma outra fonte de conhecimento; pré-determinar a execução de uma fonte de conhecimento por intervalo de tempo; estabelecer prioridades a cada fonte de conhecimento pelo número de vezes que cada uma já foi executada.

É desejável também que o sistema possua estratégias para verificar a consistência entre os elementos da solução e, eventualmente, recuperar soluções errôneas.

Em [TERR-88] admite-se que a estratégia de *backtracking* não contraria o modelo Blackboard e pode ser utilizada principalmente quando o grau de incerteza dos dados com que o sistema trabalha for muito grande. Naturalmente, um sistema eficiente não deve necessitar utilizar este método com muita frequência, mas não deve rejeitá-lo quando este se fizer necessário, bastando para isto fornecer ferramentas para que os últimos passos do sistema sejam recordados. Na falta de uma estratégia específica pra cuidar deste assunto, alguns sistemas armazenam um histórico das decisões tomadas para que a solução do problema fosse construída. Posteriormente, um especialista no domínio de aplicação do sistema pode utilizar este histórico para verificar a consistência da solução [NII -86b].

2.5 EXEMPLO DE OPERAÇÃO

O exemplo de operação da **arquitetura Blackboard** dado a seguir foi extraído de [ENGE-88a]. Nele, existe uma restrição encontrada na implementação de sistemas em ambientes computacionais seriais.

A construção da solução ocorre passo a passo, em alguma ordem determinada pelo monitor/módulo de controle (quando vários passos são possíveis e desejáveis).

Na literatura, podem ser encontrados outros exemplos. Em [HAYE-88a, HAYE-85], por exemplo, ilustra-se um caso de planejamento de rotas, onde o funcionamento do sistema é observado sob o ponto de vista do módulo de controle.

2.5.1 ENCONTRANDO COALAS

Imagine-se na Austrália. Uma das coisas mais interessantes para um turista é observar coalas em seu habitat natural. Você vai a uma reserva e começa a procurá-los. Não encontra nenhum. Você sabe que eles são relativamente pequenos, cinzentos, e parecem-se com ursos.

A floresta é densa, e a combinação das folhas cor-de-ferrugem e a luz do sol refletindo nas folhas aumenta a dificuldade de encontrar tais criaturas, cuja cor é parecida com a do ambiente (em outras palavras, a relação sinal/ruído é baixa).

Você finalmente desiste e pergunta a um guarda florestal como encontrá-los. O guarda conta a seguinte história sobre coalas: "Coalas geralmente vivem em grupos, e sazonalmente migram para diferentes partes da floresta. Agora, eles devem estar na parte noroeste da reserva. Eles geralmente sentam-se nas partes tortas dos galhos, e sobem e descem nas árvores durante o dia para tomar a quantidade certa de sol." (conhecimento sobre o comportamento-padrão dos coalas. O guarda sugeriu uma abordagem dirigida por modelo para encontrá-los). "Se você não tiver certeza de ter identificado um deles, observe-o por um instante: ele vai mover-se, embora lentamente." (método de detecção e também de confirmação).

De posse do novo conhecimento, você retorna à floresta sabendo exatamente o que procurar. Você observa a dez metros de altura e não encontra nada, e então a quinze metros quando, de

repente, encontra toda uma colônia deles.

Vamos considerar um modo de formular este problema seguindo o modelo Blackboard. Muitos tipos de conhecimento podem ser utilizados no problema: a cor e a estrutura dos coalas, a cor e a textura do ambiente (características de ruído), o comportamento dos coalas, efeitos da estação, hora do dia, e assim por diante. Algum conhecimento é informal - os locais mais prováveis para encontrar coalas numa certa época, ou seu lugar favorito de descanso. Como essas fontes de conhecimento podem ser efetivamente usadas? Primeiro, precisamos definir o que constitui uma solução para o problema. Então, podemos considerar que tipos de informação estão presentes nos dados, o que pode ser inferido deles, e que conhecimento pode ser utilizado para o objetivo de encontrar coalas.

Pode-se pensar na solução para este problema como uma série de detalhes em uma série de fotos da floresta. Os detalhes poderiam dizer: "Este é certamente um coala, pois tem cabeça, tronco e membros, e porque mudou de posição desde a última foto", ou "Este poderia ser um coala, porque tem uma parte que se parece com uma cabeça", ou "Estes poderiam ser coalas, pois estão próximos de um coala, e algumas partes poderiam ser cabeças, pernas, ou troncos". A característica importante da solução é que ela consiste de pedaços de informação, e é uma solução racional, com evidências e linhas de raciocínio.

Decidimos que a solução consiste de identificações parciais e hipotéticas, e também de soluções completas construídas de outras parciais. Precisamos de uma organização do espaço de soluções, que possa manter descrições das partes de coalas. Uma estrutura assim pode ser uma hierarquia de partes. A cada nível existem descritores apropriados, como por exemplo, sexo e peso no nível mais alto. Cada parte do corpo é descrita nos níveis inferiores em termos de entidades geométricas, tais como traços e segmentos de linha. Cada traço tem cor e textura associados. Para identificar uma parte da fotografia como um coala, é necessário marcar a figura com segmentos de linha e regiões. As regiões e segmentos de

linha devem eventualmente ser combinados ou sintetizados de forma que o objeto construído possa ser analisado como partes de um coala, ou como um coala completo. Por exemplo, um objeto pequeno, preto e circular poderia ser um olho, mas deveria ser circundado por um outro objeto maior, que deveria ser a cabeça. Quanto mais pedaços de informação se adequem à descrição do coala, mais confiável ela pode ser. Além das partes do corpo que dão suporte à existência do animal, se o coala hipotético estiver entre dez e dezoito metros do solo, pode-se ficar mais seguro que se o mesmo objeto tivesse sido encontrado a dois metros de altura.

2.6 EXEMPLOS

Apresentamos aqui, em ordem cronológica, uma descrição de alguns **sistemas Blackboard** [NII -86b], visando exemplificar os conceitos apresentados nas seções anteriores, e evidenciar a generalidade de sua aplicação.

2.6.1 HERSAY-II

A TAREFA DO SISTEMA

O objetivo do sistema HERSAY-II [ENGE-88g, ERMA-88a, LESS-88a] é entender frases pronunciadas oralmente.

Este problema é caracterizado especialmente por duas dificuldades:

- a) *falta de exatidão nos dados* de entrada, isto é, o desvio entre a frase que se pretendeu dizer e a recebida pelo sistema;
- b) *imprecisão das regras* de compreensão da fala.

Devido a estas dificuldades, é impossível fazer um mapeamento direto entre os sinais sonoros e uma sequência de

palavras que formam uma sentença [NII -86b]. A solução adotada no sistema HERSAY-II foi organizar um espaço de soluções parciais hierárquicas e gerar soluções mais completas através de buscas neste espaço. Um problema colateral poderia ser gerado por este processo, que consistiria na explosão combinatorial de soluções. Para evitar isto, o sistema deve impor restrições à formação de soluções parciais e modos de seleção de hipóteses.

A ESTRUTURA DO BLACKBOARD

O blackboard é dividido em níveis, que formam a hierarquia dos elementos do espaço de soluções, onde cada elemento é uma abstração de elementos do nível imediatamente inferior. Segundo [NII -86b], o blackboard em HERSAY-II deve ser visto como um espaço tridimensional onde:

- o eixo x representa o tempo, ou seja, a sequência em que os elementos foram pronunciados;
- o eixo y representa os níveis das hipóteses;
- o eixo z representa alternativas de soluções.

Cada par elemento-hipótese é constituído por uma estrutura de pares atributo-valor. Alguns atributos devem ser constantes para hipóteses de qualquer nível, como por exemplo:

- a) identificação do nível a que pertence a hipótese,
- b) avaliação da validade da hipótese,
- c) "links" para hipóteses de outros níveis,
- d) "links" para hipóteses alternativas.

AS FONTES DE CONHECIMENTO

Cada *fonte de conhecimento* divide-se em duas partes:

condição: prescreve as situações em que a fonte de conhecimento pode contribuir para a solução do problema. Uma vez em execução, procura no blackboard todas as hipóteses que podem ser processadas pela ação correspondente e as passam para a mesma.

ação: especifica como a fonte de conhecimento pode contribuir para a solução do problema em uma determinada situação. Uma vez em execução, deve processar todas as hipóteses que foram

destinadas pela condição.

Cada uma das partes consiste de um procedimento independente.

O CONTROLE

O controle é efetuado por um *monitor* e um *scheduler*, com auxílio de uma fila.

O *monitor* memoriza as alterações feitas no blackboard e as novas hipóteses geradas. Através destes dados e das informações fornecidas pelas condições das fontes de conhecimento, seleciona as condições que podem ser executadas e insere na fila um apontador para cada uma delas. A fila, além de apontadores para condições, pode conter também apontadores para ações de fontes de conhecimento, que são chamadas "fontes de conhecimento invocadas" (KSI - knowledge source invoqued). Uma ação torna-se uma KSI quando sua condição é satisfeita. Condições e KSIs apontadas na fila são chamadas *atividades*.

O *scheduler* calcula, no início de cada ciclo do sistema, uma prioridade para cada atividade da fila. Para isto, utiliza heurísticas na forma de procedimentos, que determinam quais as atividades capazes de fazer as alterações mais importantes no blackboard com o mínimo de processamento.

Em resumo, o algoritmo de controle é o seguinte:

- I. Chama o scheduler para selecionar uma atividade da fila.
- II. Se a atividade selecionada for uma condição e esta for satisfeita, insere na fila um apontador para a ação correspondente, junto com a relação das hipóteses que satisfizeram a condição.
- III. Se a atividade selecionada for uma ação, ela é executada e o blackboard alterado. O monitor insere na fila apontadores para as condições que potencialmente podem ser satisfeitas pelas alterações que foram feitas no blackboard.

- IV. Repete os passos I, II e III, até que uma solução o mais completa possível seja encontrada para o problema.

O controle de HERSAY-II utiliza três tipos de estratégias para gerar as soluções parciais: bottom-up, top-down e teste de hipóteses. Segundo esta última estratégia, uma fonte de conhecimento testa as hipóteses geradas por outra fonte de conhecimento. Isso se dá através do confronto destas hipóteses com elementos de uma base de conhecimento, gerenciada pela primeira fonte. Estas hipóteses correspondem às hipóteses aceitáveis em uma determinada situação. Com isto pretende-se evitar a explosão combinatorial de hipóteses, que podem ser geradas pelas estratégias bottom-up e top-down.

2.6.2 HASP

A TAREFA DO SISTEMA

A função do sistema HASP [NII -88b] é identificar a presença de submarinos e navios em uma certa área do oceano através de sinais acústicos emitidos por vários conjuntos de hidrofones colocados na região em observação, e relatórios sobre características de embarcações amigas, inimigas e comerciais.

As principais dificuldades encontradas pelo sistema são:

- a) o espaço de soluções possíveis é muito grande e pouco conhecido;
- b) a sintaxe e a semântica das soluções possíveis são pouco conhecidas, uma vez que os dados sobre os submarinos inimigos são limitados, e estes ainda adotam várias táticas para não serem reconhecidos;
- c) os sinais acústicos recebidos muitas vezes são bastante confusos, pois pode haver interferência entre ondas sonoras emitidas por um objeto sobre ondas emitidas por outro.

Assim, a construção de uma solução deve basear-se principalmente nos métodos e heurísticas adotados por especialistas no domínio do problema para identificar e classificar as embarcações.

A ESTRUTURA DO BLACKBOARD

A estrutura de dados que constitui o blackboard armazena a melhor interpretação da situação da área do oceano sob observação em um determinado instante. Esta interpretação é chamada CBH (current best hypothesis), ou melhor hipótese corrente. A melhor hipótese corrente é estruturada hierarquicamente em diferentes níveis de abstração, que, como mostra a Figura 2.2, são: *linhas*, *conjuntos harmônicos*, *fontes acústicas*, *plataformas* e *área do oceano*.

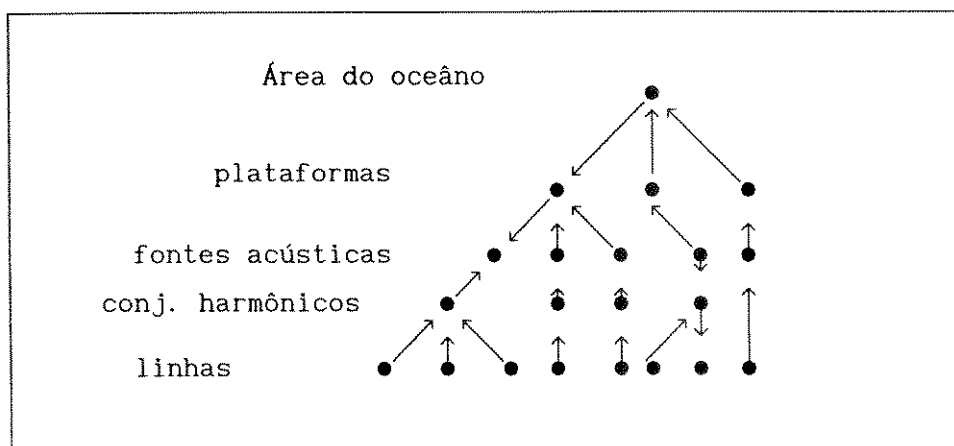


Figura 2.2 - Estrutura do Blackboard do Sistema HASP/SIAP

Cada nó do blackboard é denominado *elemento da hipótese*. Esses elementos formam uma rede em forma de árvore, onde as "folhas" correspondem aos elementos agregados por ele. Cada elemento da hipótese é representado por um conjunto de pares atributo-valor, sendo que os atributos podem variar de um nível para outro do blackboard.

ESTRUTURAS AUXILIARES DE INFORMAÇÃO

Além do blackboard, HASP possui outras estruturas de dados globais, que guardam informações auxiliares ao trabalho das fontes de conhecimento e do controle. Estas estruturas são:

lista de eventos: contém as alterações feitas no blackboard e que ainda não foram processadas. Cada alteração é identificada pelo seu tipo e pelo elemento do blackboard alterado. Esta lista será utilizada pelo controle para a determinação do foco de atenção;

lista de expectativas: relaciona elementos previstos para acontecer no futuro;

lista de problemas: relata problemas encontrados pelas fontes de conhecimento, como por exemplo, a falta de uma informação que supostamente deveria existir;

lista de eventos associados ao relógio: cada elemento desta lista consiste de um horário e um conjunto de regras que deverão ser executadas neste horário. Como o comportamento de algumas embarcações é conhecido, pelo menos sob alguns aspectos, pode-se prever qual será a situação da área sob observação após um certo período de tempo. Com isto, pode-se pré-determinar a ativação de certas fontes de conhecimento, economizando o processamento utilizado pelo controle;

histórico: relata todos os eventos já processados e o contexto em que ocorreram. Estas informações podem ser utilizadas pelo usuário para verificar se uma solução foi encontrada através de deduções coerentes.

FONTES DE CONHECIMENTO

Cada fonte de conhecimento é dividida em duas partes:

situação: determina em que situações a fonte de conhecimento deve ser ativada. É composta por uma lista de tipos de eventos aos quais a fonte de conhecimento deve responder;

ação: determina as alterações que a fonte de conhecimento deve efetuar sobre a hipótese corrente. Estas alterações podem

ser: criação de novos elementos da hipótese, estabelecimento de ligações entre elementos já existentes, ou ainda verificação de consistência de alterações provocadas por outras fontes de conhecimento. É formada por um conjunto de regras, cujas pré-condições fazem uma verificação mais apurada para saber se existem ou não condições satisfatórias à execução da ação.

Ao contrário da estrutura utilizada em HERSAY-II, em HASP as fontes de conhecimento são compactas, ou seja, as duas partes em que elas se dividem fazem parte de um único módulo. Em HASP, as fontes de conhecimento são organizadas hierarquicamente em três níveis de acordo com os seus tipos:

estratégicas: verificam que parte da melhor hipótese corrente deve ser alterada e que classe de eventos deve ser considerada, determinando, por consequência, as fontes ativadoras que devem ser tratadas. Ocupam o nível superior da hierarquia das fontes de conhecimento.

ativadoras: dada uma classe de eventos, selecionam um deles e determinam quais fontes de conhecimento serão ativadas. Ocupam o nível intermediário da hierarquia;

especialistas: alteram efetivamente a melhor hipótese corrente quando suas pré-condições forem satisfeitas. Ocupam o nível inferior da hierarquia;

Cada fonte de conhecimento deve conhecer o vocabulário referente aos atributos que descrevem os elementos da hipótese a que ela se direciona.

O CONTROLE

Os módulos de controle são implementados como fontes de conhecimento. O controle efetivo do sistema é induzido pela disposição hierárquica destas fontes e pela ordem em que elas devem ser executadas. O sistema deve possuir uma relação de tipos de eventos que podem ocorrer durante sua execução e pré-determinar as fontes de conhecimento que devem processar cada um destes tipos.

Em linhas gerais, os passos que constituem um ciclo do controle do sistema HASP são os seguintes:

- I. Uma fonte de conhecimento estratégica determina que classe de eventos deve ser focalizada, isto é, se devem ser processados eventos associados ao relógio, expectativas, problemas ou elementos do blackboard. Além disso, determina qual área do blackboard deve ser prioritariamente alterada para que a melhor hipótese corrente tenha maior evolução.
- II. A fonte de conhecimento ativadora correspondente à classe de eventos selecionada escolhe um dentre os eventos desta classe.
- III. As fontes de conhecimento especialistas que tiverem na sua *situação* o tipo do evento selecionado em II são ativadas.
- IV. As fontes selecionadas em III que tiverem as pré-condições de suas regras aceitas alteram o blackboard.

Um evento é descrito pelo nome de um nó (ou elemento) do blackboard e pelo tipo de modificação que este nó sofreu ou deverá sofrer.

O sistema é aberto à inclusão de novos tipos de eventos e fontes de conhecimento, sem que estes afetem os já existentes.

Regras empíricas de associação de elementos da hipótese, fornecidas por especialistas no domínio de aplicação, constituem os recursos utilizados para a geração de soluções parciais de um nível a partir de elementos de outros níveis.

A maior parte do esforço realizado pelas fontes de conhecimento para a construção de partes da solução é direcionado ao processamento *bottom-up*, *orientado por dados*. No entanto, a estratégia mais poderosa utilizada por HASP é o processamento *top-down*, *orientado por modelo*. Esta estratégia reduz bastante o processamento necessário para as tentativas de combinações de

hipóteses parciais, uma vez que, possuindo um modelo de uma solução parcial que deverá ser encontrada, determina com antecedência a qual elemento de um nível superior corresponde um grupo de elementos de um outro nível inferior a ele [NII -86b].

2.6.3 CRYNALIS

A TAREFA DO SISTEMA

A tarefa do sistema CRYNALIS [TERR-83] é inferir a estrutura tri-dimensional de moléculas de proteína.

A estrutura molecular de uma proteína pode ser obtida a partir de uma interpretação do *mapa de densidade de elétrons* (MDE). Existem dificuldades para chegar-se à solução do problema, como a falta de conhecimentos formais e específicos sobre a interpretação de mapas de densidade de elétrons e construção de modelos de moléculas. Além disto, o espaço de soluções é extraordinariamente grande e impede a busca por tentativa e erro.

Além do mapa de densidade de elétrons, CRYNALIS pode contar com a sequência de amino-ácidos que compõem a proteína. Esta sequência pode não ser correta, ou por estar incompleta ou fora de ordem.

Dados utilizados pelo sistema são abstrações do mapa de densidade de elétrons produzidas por algoritmos matemáticos. Estas abstrações podem ser de três tipos:

- a) funções, que fornecem a densidade de elétrons em cada ponto do mapa. Os picos (valores máximos) representam a presença de um ou mais átomos.
- b) um grafo, chamado de *esqueleto*, onde cada vértice representa um pico ou uma região de alta densidade de elétrons. Este tipo de abstração é essencial para a solução do problema.
- c) uma conexão de regiões do esqueleto (segmentos), que

visa fornecer uma visão mais global da molécula.

CRYBALIS lida com um problema que é comum a sistemas especialistas que utilizam uma quantidade muito grande de conhecimento e heurísticas: o *problema de foco de atenção*. Este problema consiste em determinar quais recursos e heurísticas utilizar quando vários caminhos parecem levar ao progresso da solução.

A ESTRUTURA DO BLACKBOARD

O blackboard é dividido em dois "*panels*": *panel de densidades* e *panel de hipóteses*, que por sua vez são estruturados hierarquicamente em níveis, como mostra a Figura 2.3, adaptada de [NII -86b].

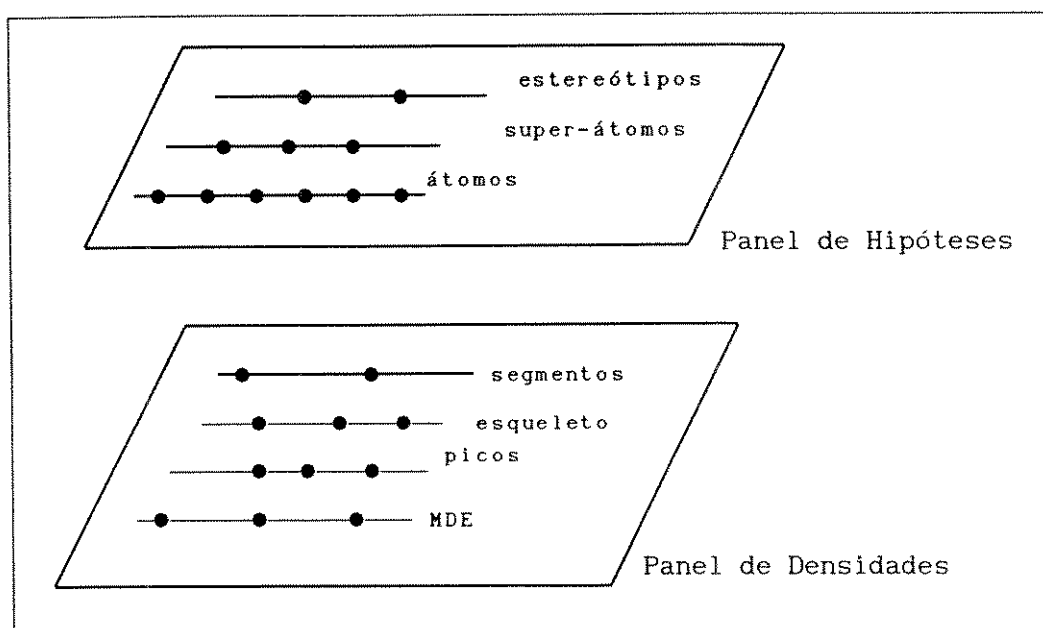


Figura 2.3 - Panels do Sistema CRYBALIS

O *panel de densidades* é dividido (em ordem crescente de abstração) nos níveis: MDE (contém o mapa de elétrons assim como ele é recebido), picos, esqueletos e segmentos. Os elementos deste plano são produzidos por algoritmos que processam os dados coletados antes do processo de interpretação.

O *panel de hipóteses* é dividido (em ordem crescente de abstração) nos níveis: átomos, super-átomos (representam grupos de átomos) e estereótipos (configurações da estrutura da molécula em termos figurativos). Cada elemento deste plano é implementado como um *átomo de LISP* e é derivado a partir de dados provenientes do *panel de densidades*.

AS FONTES DE CONHECIMENTO

As fontes de conhecimento armazenam todo o conhecimento que o sistema utiliza para a interpretação do MDE. Cada fonte de conhecimento é formada por um conjunto de regras formais de produção, dividido em variáveis e regras. As variáveis existem para remover computações redundantes das regras e criar um vocabulário mais significativo para o conhecimento expresso da fonte de conhecimento. Ao contrário das fontes de conhecimento de outros sistemas, em CRYSALIS elas não possuem pré-condições. Neste sistema, as fontes de conhecimento não são auto-ativadas, ou seja, não determinam quando elas próprias devem alterar a solução. Sua ativação é realizada pela estrutura de controle.

O CONTROLE

Em CRYSALIS, os módulos de controle são representados por fontes de conhecimento e são organizados hierarquicamente em três níveis: *estratégias*, *tarefas* e *objetos*. Cada nível contém regras que examinam o estado atual da solução e selecionam uma ação no nível inferior, e os módulos do nível mais baixo alteram a solução efetivamente.

O nível de *estratégias* é ocupado por uma única fonte de conhecimento. Esta fonte tem acesso a uma estrutura auxiliar do blackboard, chamada *lista de características*, que possui um resumo da situação da solução, por regiões. A fonte de conhecimento estratégica seleciona uma região da solução que deve ser trabalhada e, baseada nas propriedades atuais desta região, seleciona e executa uma fonte de conhecimento do nível de *tarefas*.

A fonte-tarefa selecionada, baseando-se nas últimas alterações feitas na região escolhida pela fonte de conhecimento estratégica, determina uma sequência de fontes de conhecimento objeto que devem ser executadas, alterando o estado real do blackboard. A relação das últimas alterações feitas no blackboard deve estar armazenada em uma outra estrutura auxiliar chamada *lista de eventos*.

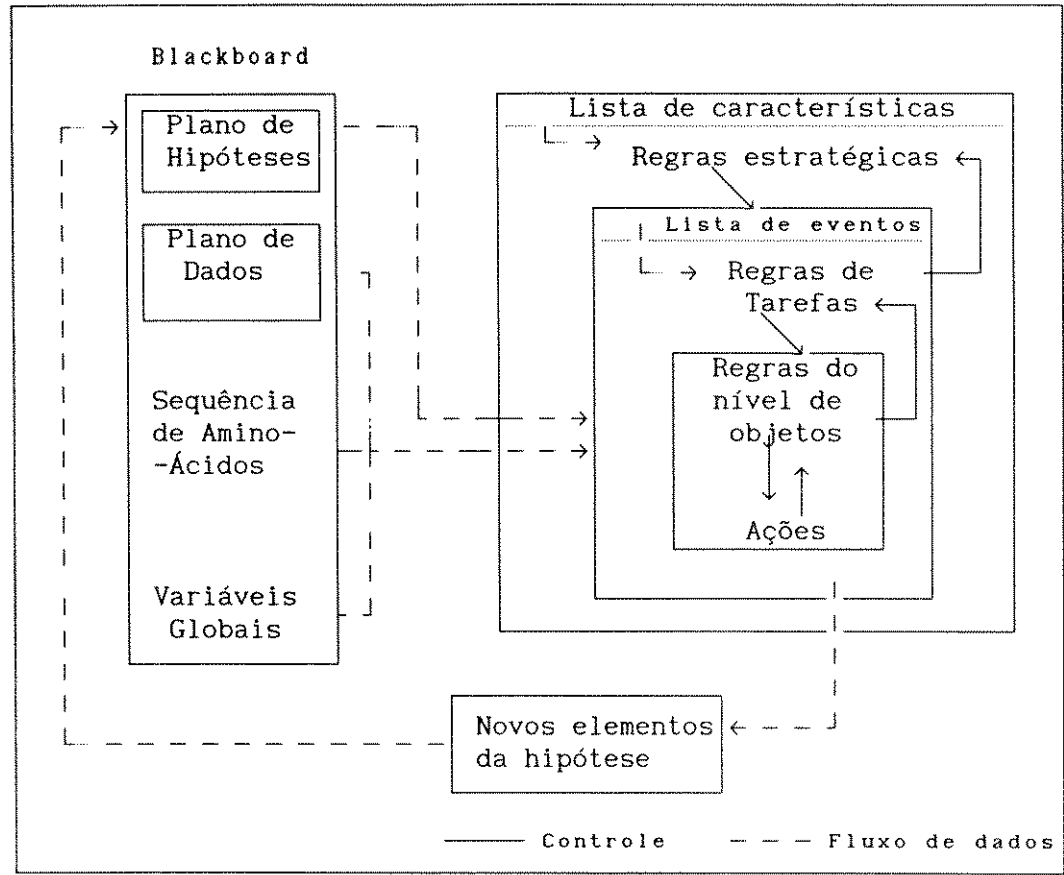


Figura 2.4 - Configuração do Sistema CRYSLIS.

Após a execução das fontes de conhecimento objeto, o controle retorna à fonte-tarefa. Esta, então, atualiza a *lista de características*, de acordo com as alterações realizadas na região em foco. A seguir, seleciona uma nova relação de *objetos* a serem

executados, se a situação assim o exigir. Quando a tarefa chegar ao fim, o controle volta para a fonte de conhecimento estratégica, que inicia um novo ciclo de controle, focalizando uma outra região da solução, e selecionando uma outra fonte de conhecimento *tarefa*. O sistema de controle de CRYSLIS está resumido na Figura 2.4.

O sistema de controle construído em CRYSLIS determina que a busca de uma solução deve ser direcionada dando prioridade ao crescimento de ilhas de soluções parciais. Isto implica que a fonte de conhecimento estratégica deve escolher como foco de atenção a região da solução que esteja mais compacta e que tenha maiores possibilidades de tornar-se mais completa, ainda que seja ligando-se a outras ilhas. Focalizada uma região, a fonte *tarefa* correspondente a esta região determina *oportunisticamente* qual nó deve ser modificado. Este é o único ponto em que o oportunismo é aplicado neste sistema.

2.6.4 DICE

A TAREFA DO SISTEMA

DICE [SRIR-89] é um sistema que visa automatizar o processo de projetos de engenharia, partindo de um modelo que representa as expectativas e as restrições impostas pelos engenheiros ao produto.

A proposta de DICE é ser uma rede de computadores, onde em cada computador reside um usuário chamado *módulo de conhecimento*. Os módulos de conhecimento devem ser coordenados por um mecanismo de controle através de um *blackboard*. A visão conceitual do sistema é ilustrada na Figura 2.5.

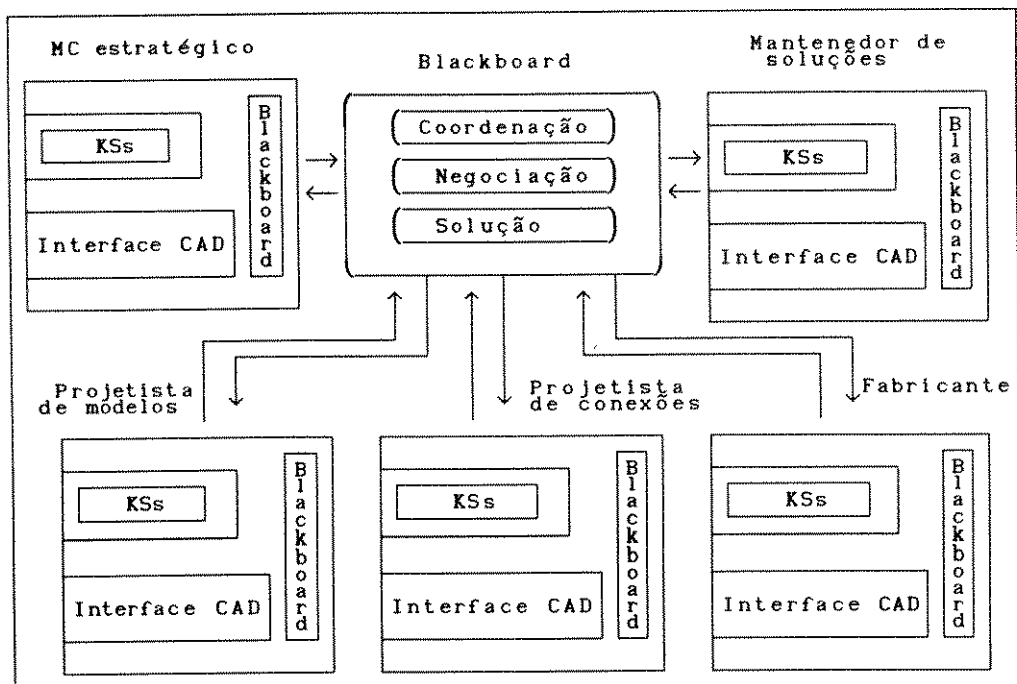


Figura 2.5 - Visão Conceitual do Sistema DICE

DICE consiste de um conjunto de ferramentas auxiliadas por computador, que foram desenvolvidas com os seguintes objetivos:

- a) Facilitar a coordenação e a comunicação envolvidas com a engenharia.
- b) Identificar o processo pelo qual os projetistas tomam decisões, isto é, quais informações são utilizadas e como são utilizadas no desenvolvimento de um projeto.
- c) Prever o impacto do projeto na fabricação do produto, isto é, assegurar que projeto poderá tornar-se um produto real.
- d) Fornecer ao projetista planos detalhados para a fabricação do produto.
- e) Desenvolver interfaces inteligentes para a automação do processo.

O BLACKBOARD

O *blackboard* é o meio pelo qual *toda* a comunicação entre os módulos de conhecimento é feita. Em DICE, o blackboard é dividido em três partes:

partição de soluções: contém informações sobre o projeto gerado pelos módulos de conhecimento. Estas informações são representadas na forma de uma hierarquia de objetos, que descrevem o produto a ser construído e o processo de projetá-lo.

partição de negociações: consiste de linhas de negociação entre vários engenheiros que participam do processo de projeto e fabricação do produto.

partição de coordenação: contém informações necessárias à coordenação de vários módulos de conhecimento.

MÓDULOS DE CONHECIMENTO

Cada *módulo de conhecimento* (MC) pode ser visto como um sistema especialista, também contruído como um sistema blackboard, formado por uma agregação de fontes de conhecimento (KSs), um blackboard particular para a comunicação destas fontes de conhecimento, e opcionalmente, uma interface com o usuário e ferramentas CAD. Cada KS contém uma parte independente do conhecimento utilizado no desenvolvimento do projeto, representado na forma de regras ou objetos.

Os *módulos de conhecimento* são agrupados em três categorias:

estratégicos: auxiliam o mecanismo de controle na coordenação e comunicação dos processos que executam o projeto;

especialistas: executam tarefas especializadas do processo de construção do produto;

quantitativos: ferramentas CAD.

O CONTROLE

O mecanismo de controle é definido por funções de comunicação, coordenação, transferência de dados, etc. que, em resumo, têm como objetivo duas tarefas:

- a) Avaliar e propagar os efeitos de ações tomadas por um módulo de conhecimento.
- b) Auxiliar o processo de negociação entre vários módulos.

Os passos que se executa para realizar estas tarefas são (ver Figura 2.5):

- I. Um esboço preliminar do produto (em forma de composição de objetos) é gerado por um módulo de conhecimento - o conceitual, ou projetista de modelos - e colocado na partição de soluções do blackboard. Este esboço deve incluir as intenções dos projetistas em tomar certas decisões .
- II. Detalhes de conexão são representados pelo objeto tipo conexão. O projetista de conexões envia uma mensagem com detalhes de conexões e suposições feitas durante o projeto.
- III. O mantenedor de soluções verifica se as suposições mais recentes feitas pelo projetista não foram violadas pelo estado atual do sistema.
- IV. Métodos associados ao objeto tipo conexão indicam os módulos de conhecimento que podem modificá-lo. Se um destes módulos de conhecimento for o fabricante, por exemplo, então uma mensagem é enviada ao mesmo para que ele verifique se a conexão em questão pode ser feita ou não.
- V. O projetista de conexões é informado se algum problema foi antecipado.
- VI. Algumas vezes, dois ou mais módulos de conhecimento podem querer modificar ou acessar um objeto particular da partição de soluções. Essa informação é armazenada e utilizada pelo mecanismo de controle.

2.7 ABORDAGENS

São apresentadas aqui algumas abordagens utilizadas no desenvolvimento de sistemas Blackboard. Através de sua observação, pode-se perceber o vasto potencial de uso deste paradigma.

2.7.1 BLACKBOARDS EM SISTEMAS DISTRIBUÍDOS

Um sistema distribuído de Inteligência Artificial (sistema DAI) consiste de múltiplos nodos de processamento fisicamente separados, cada qual tendo pelo menos um sistema especialista [YANG-85].

A abordagem distribuída oferece a possibilidade de respostas mais rápidas, em tempo real, maior segurança, flexibilidade e custos menores de processamento.

Em um sistema DAI para projeto de engenharia, as fontes de conhecimento são distribuídas entre os nodos, e nenhum deles pode resolver sozinho todo o problema. Além disso, os dados de entrada e as restrições de projeto geralmente são exatas e disponíveis para todos os nodos. Cada um dos nodos tem uma visão parcial do problema, conhecimento incompleto, ou ambos. Os dados e o domínio do conhecimento podem também ser incompletos ou inexatos.

Em um sistema distribuído, os custos de comunicação são altos quando comparados aos de processamento. Além disso, quando a taxa de chegada de mensagens se aproxima da taxa de serviço de um canal de comunicação numa rede de computadores, o desempenho do sistema é degradado significativamente [KLEI-76]. Dessa forma, a comunicação torna-se importante no projeto de um sistema DAI.

Os principais pontos envolvidos no projeto de um sistema desse tipo são:

- a) uma distribuição apropriada dos subproblemas entre os

nodos de processamento;

- b) a escolha dos mecanismos de controle deve manter a coerência global durante o processo de solução do problema, deve utilizar eficientemente as fontes de conhecimento, e deve obter desempenho ótimo;
- c) a especificação dos aspectos de comunicação deve ser feita de forma tal que os nodos possam interagir e cooperar entre si.

Os nodos do sistema apresentado em [YANG-85] implementam:

- a) capacidade de planejamento dinâmico, que ajusta o plano de solução do problema;
- b) facilidades de comunicação intra-nodos;
- c) facilidades de comunicação entre-nodos;
- d) capacidade de dedução, através da invocação de fontes de conhecimento;
- e) capacidade de aprendizagem, que permite ao sistema mudar a sua organização e melhorar o seu desempenho.

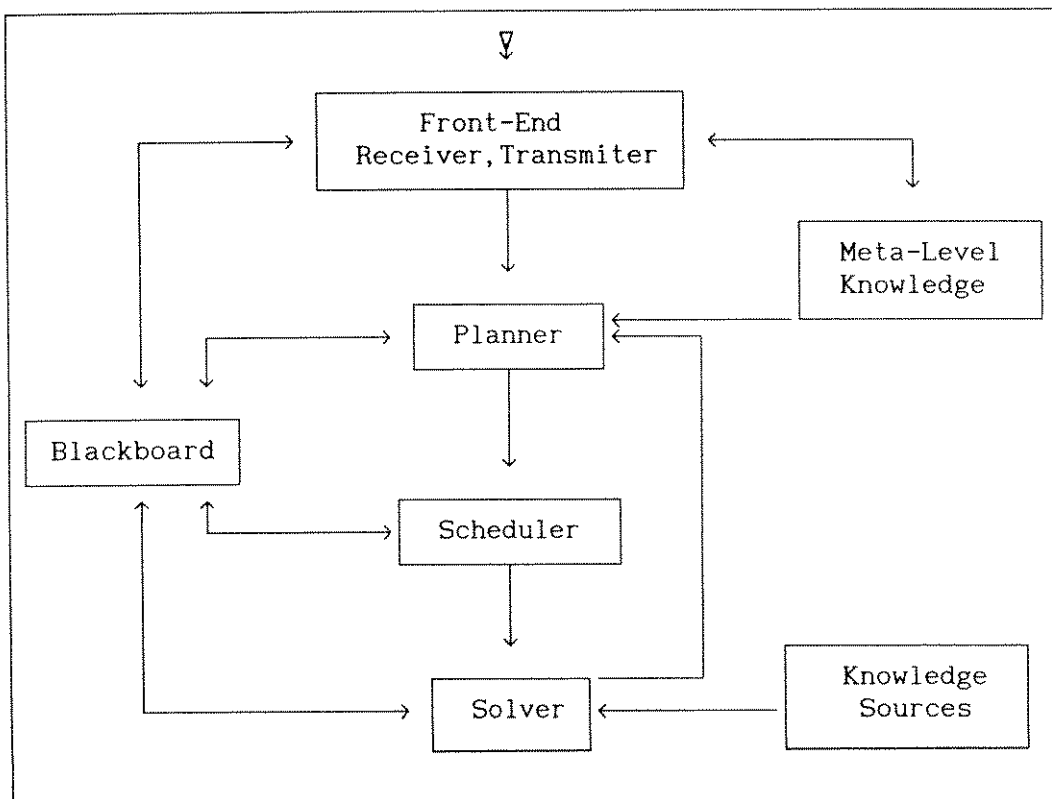


Figura 2.6 - Organização dos Nodos.

A arquitetura dos nodos é mostrada na Figura 2.6.

São componentes da arquitetura:

- a) *Meta-Level Knowledge*. Cada nodo tem acesso ao conhecimento sobre os tipos de problema que os nodos podem solucionar, e uma medida de qualidade;
- b) *Blackboard*. É quem suporta as comunicações entre-nodos;
- c) *Processador Front-End*. Permite aos nodos compartilhar tarefas e resultados;
- d) *Planner*. Evocado quando existem tarefas cujas soluções necessitam de planejamento inicial ou replanejamento;
- e) *Solver*. Determina os sub-planos existentes na solução corrente;
- f) *Knowledge Sources*. Após criados os planos de resolução do problema, as fontes de conhecimento apropriadas são inicializadas para resolvê-lo.
- g) *Scheduler*. Gerencia a execução das tarefas.

2.7.2 BLACKBOARDS TRANSACIONAIS

Os ambientes computacionais multiprocessados podem aumentar muito a capacidade e a utilidade da arquitetura Blackboard. Neste caso, o controle de concorrência e a integridade dos dados passa a ser um problema fundamental. O acesso assíncrono a esses dados pode ser provido eficazmente através de uma abordagem transacional.

Blackboards transacionais são, segundo [CARD-87], uma extensão da arquitetura Blackboard que permite que as fontes de conhecimento sejam processadas em paralelo, em um único ou em vários processadores. Estes podem acessar concorrentemente os dados armazenados no blackboard, que deve estar centralizado sob controle de um só processador. Blackboards transacionais utilizam o conceito de **transações** para disciplinar o acesso à memória compartilhada.

Uma transação seria iniciada com um requisito de início de transação e terminada por uma confirmação ou pedido de cancelamento. Tem três propriedades :

- a) a *atomicidade*, que garante que mesmo em presença de falhas, ou todas as operações ocorrem (a transação se realiza), ou nenhuma delas acontece;
- b) *consistência*, significando que uma transação só movimenta dados de um estado consistente para outro. Existem pelo menos duas formas de preservação da consistência :
 - i) *consistência inter-transacional* (entre várias transações), e
 - ii) *consistência intra-transacional* (uma visão consistente dos dados para todos que atuam em uma mesma transação);
- c) *permanência*, ou seja, o efeito de uma transação confirmada persiste até que a próxima transação envolvendo aquele dado seja confirmada.

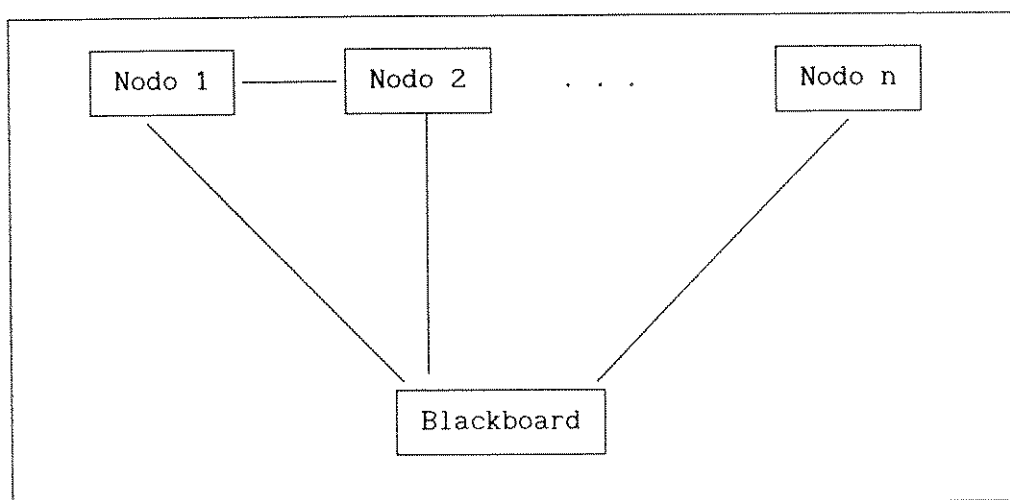


Figura 2.7a - Rede de Nodos de Processamento

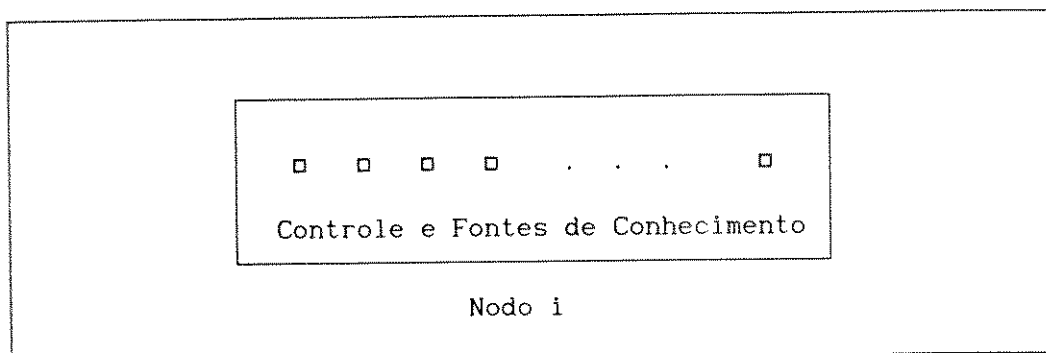


Figura 2.7b - Controle e Fontes de Conhecimento em um Nodo.

Na solução proposta por [ENSO-86], as fontes de conhecimento podem comunicar-se acessando compartilhadamente os dados, em transações separadas (cada fonte de conhecimento acessa os dados através de uma transação diferente, por exemplo). Podem também participar de uma mesma transação (duas fontes de conhecimento utilizam-se de uma mesma transação para obter a mesma visão dos dados, isto é, dados no mesmo estado). A Figura 2.7 ilustra a arquitetura proposta, onde o blackboard reside em uma máquina única, embora possa ser distribuído.

A atomicidade é obtida através do conceito de transação.

Para a obtenção da consistência, utiliza-se um gerente de transações. As fontes de conhecimento acessam o blackboard através de mensagens que são trocadas com este gerente. O gerente de transações controla o acesso aos dados através de *locks*. Existem *locks* de leitura e de escrita. O *lock* de escrita fornece acesso exclusivo ao dado, que pode ser modificado. O *lock* de leitura fornece acesso concorrente ao dado. Nesse último caso, não se dá direito à escrita. Os *locks* persistem até o final da transação. Essa solução garante a consistência inter-transacional [ESWA-76]. Além disso, o gerenciador de transações checa situações de *deadlock*. A consistência intra-transacional é obtida por *time-stamps*, que induzem a uma serialização no tempo, como proposto em [BERN-81]. Essa serialização significa que a cada solicitação de leitura ou escrita, é atribuído um *time-stamp* para

a transação solicitada. Só se pode ler o dado se o *time-stamp* for maior que o de sua escrita, garantindo que o acesso sempre é realizado sobre o estado desejado. Da mesma forma, só se pode escrever se o *time-stamp* for maior que o de sua leitura e escrita.

A permanência é obtida mantendo-se cópias dos dados independentemente, com suporte de protocolos de recuperação.

2.7.3 BLACKBOARDS ORIENTADOS A OBJETOS

Uma outra abordagem possível é a construção de sistemas Blackboard orientados a objetos. Essa abordagem foi utilizada para simulações e é apresentada em [ZANC-88], onde a aplicação é intrinsecamente ligada à resolução contínua em tempo real.

Cada objeto do blackboard consiste de um conjunto de definições e declarações que os descrevem de forma genérica. A partir dessas definições, são criadas instâncias dos objetos.

Os principais componentes da definição de objetos são:

- a) um conjunto de declarações de *variáveis locais*.

Cada instância de um objeto possui uma instância desse conjunto de variáveis que definem totalmente o seu estado. Algumas dessas variáveis podem ser de uso privativo do objeto e outras (públicas) poderão vir a ser acessadas por outros objetos;

- b) um conjunto de definições de *pseudo-variáveis*.

Essas definições fornecem operadores que, quando avaliados, retornam o valor da pseudo-variável. Novamente, podem ser de uso privativo ou não, de forma a prover o escopo apropriado;

- c) um conjunto de definições de *procedimentos*.

Esses procedimentos são locais à definição do objeto e auxiliam na leitura do código.

- d) um conjunto de *comportamentos*.

Os comportamentos são a unidade principal do processamento de um objeto. Tratam-se de blocos de código

ativados por eventos.

- e) uma lista de *heranças*.

Essas heranças estendem o conjunto de definições e declarações de objetos, fazendo com que passem a possuir o conjunto de propriedades das definições mais genéricas.

Várias formas de comunicação entre os objetos podem co-existir. Por exemplo:

- a) pedido do valor de variáveis públicas de outros objetos;
- b) invocação de operadores públicos, que avaliarão as pseudo-variáveis;
- c) criação de uma nova instância de um objeto;
- d) envio de mensagens a instâncias de um objeto;
- e) atualização da instância de um objeto, que leva à ativação de procedimentos ligados às suas variáveis;
- f) ativação de procedimentos monitorando a criação ou deleção de instâncias de objetos;
- g) execução de algum trecho de código de programa.

Dessa forma, um objeto pode tomar conhecimento sobre um evento lendo variáveis públicas de outros objetos ou recebendo mensagens.

Para otimizar o uso da arquitetura, deve-se estender as definições dos objetos, dando a eles mais "inteligência" (por exemplo, inserindo neles conhecimento sobre o mundo, restrições de tempo, etc.).

2.8 CONCLUSÃO

O **modelo Blackboard** é bastante genérico e flexível, não induzindo a procedimentos rígidos, permitindo vastas possibilidades de implementação.

A **arquitetura Blackboard** é uma dentre várias alternativas possíveis para resolução automática de problemas. Existem razões que podem nos levar a preferir a abordagem **Blackboard** [ENGE-88f]:

- *modularidade* (tanto das fontes de conhecimento como dos módulos de controle), o que permite que o sistema possa ser testado e mantido com mais facilidade;
- *controle dinâmico*. A arquitetura fornece várias alternativas de controle do comportamento do sistema durante a resolução do problema, de modo que este seja o mais eficiente possível em diferentes situações;
- *eficiência*. Um projeto cuidadoso dos módulos e uma política de controle que selecione as fontes de conhecimento mais eficazes, ou escolha as regiões mais profícuas, pode levar a ganhos consideráveis de eficiência.
- *concorrência*, uma vez que os dados do blackboard podem ser compartilhados por várias fontes de conhecimento e estas trabalham sobre áreas semanticamente diferentes da solução. A arquitetura permite que diferentes partes de um problema sejam processadas concorrentemente.

Num projeto de sistemas Blackboard, existem alguns pontos que devem merecer atenção especial, pois podem comprometer sua eficiência [GASS-89]:

- *esquemas para acesso ao conhecimento contido no blackboard*, para que as fontes de conhecimento tenham melhores condições de contribuir para a solução do problema, conduzindo a melhorias conceituais e

otimização;

- *granularidade na representação dos elementos da solução*, para facilitar o acesso e melhorar a visão que as fontes de conhecimento têm do estado do problema;
- *sincronização e resolução de conflitos entre fontes de conhecimento*, como forma de otimização;
- *granularidade das ações das fontes de conhecimento*, seguindo inclusive um plano de resolução de mais alto nível, como forma de coordenar suas ações.

Arquiteturas Blackboard são particularmente apropriadas para sistemas que lidem com problemas que possuam as seguintes características:

- a) A construção de uma solução implica na utilização de vários domínios de conhecimento.
- b) O problema pode ser naturalmente dividido em níveis hierárquicos.
- c) Os dados utilizados possuem um alto grau de incerteza.

As propriedades a e b induzem-nos a concluir que Blackboards podem ser utilizados como ferramentas de engenharia de software [DODH-89].

O **modelo Blackboard** antecipa arquiteturas computacionais paralelas e multiprocessamento, permitindo que muito seja feito ainda nesta área.

Em geral, muito ainda pode ser pesquisado e desenvolvido sobre Blackboard, tanto no que se refere a novas formas de organização e implementação de sistemas deste tipo, como em outras áreas de aplicação para este modelo.

REFERÊNCIAS

- [BERN-81] Bernstein P.A., Goodman N., *Concurrency Control in Distributed Database Systems*, ACM Computing Surveys, vol.13, no.2, June 1981, pp.185-221.
- [CARD-87] Cardoso E., *DPSK: A Kernel for Distributed Problem Solving*, Tese de Doutorado, Carnegie Mellon University, 1987.
- [CORK-86] Corkill D., Gallagher K., Murray K., *GBB: A Generic Blackboard Development System*, Proceedings of the National Conference on Artificial Intelligence, Philadelphia, August 1986, pp. 1008-1014. Também em *Blackboard Systems*, Addison-Wesley Publ. Co., 1988, pp. 503-517.
- [DODH-89] Dodhiawala R., Jagannathan V., Baum L., Skillman T., *The First Workshop on Blackboard Systems*, AI Magazine, vol.10, no.1, Spring 1989, pp. 77-80.
- [ENGE-88a] Englemore R.S., Morgan A.J., *Introduction*, *Blackboard Systems*, Addison-Wesley Publ. Co., 1988, pp. 01-22.
- [ENGE-88b] Englemore R.S., Morgan A.J., Nii H., *Early Applications (1975-1980)*, *Blackboard Systems*, Addison-Wesley Publ. Co., 1988, pp. 125-134.
- [ENGE-88c] Englemore R.S., Morgan A.J., *Generalizations (1977-1980)*, *Blackboard Systems*, Addison-Wesley Publ. Co., 1988, pp. 247-250.
- [ENGE-88d] Englemore R. S., Morgan A.J., *Recent Applications (1981-1985)*, *Blackboard Systems*, Addison-Wesley Publ. Co., 1988, pp. 349-352.

[ENGE-88e] Engelmores R.S., Morgan A.J., *Current Directions (1985-1987)*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 461-464.

[ENGE-88f] Engelmores R.S., Morgan A.J., *Conclusion*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 561-574.

[ENGE-88g] Engelmores R.S., Morgan A.J., Nii H.P., *Hearsay-II*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 25-29.

[ENSO-86] Ensor J.R., Gabbie J.D., *Transactional Blackboards*, Proceedings of the 1985 International Joint Conference on Artificial Intelligence, August 1985, pp. 340-344. Também em International Journal for Artificial Intelligence in Engineering, vol.1, no.2, 1986, pp. 80-84.

[ERMA-88a] Erman L.D., Hayes-Roth F., Lesser V.R., Reddy D.R., *The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 31-86.

[ERMA-88b] Erman L.D., London P.E., Fickas S.F., *The Design and an Example Use of Hearsay-III*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 281-295.

[ESWA-76] Eswaran K.P., et alia, *The Notions of Consistency and Predicate Locks in a Database System*, CACM, pp.624-633, vol 19, no. 11, November 1976, pp. 624-633.

[GASS-89] Gasser L., *Distributed Artificial Intelligence*, AI Expert, July 1989, pp. 26-33.

[HANS-87] Hanson A., Riseman E., *The VISIONS Image Understanding System 1986*, Advances in Computer Vision, Editado por Chris Brown, Erlbaum, 1987.

[HAYE-79] Hayes-Roth B., Hayes-Roth F., Rosenschein F., Cammarata S., *Modelling Planning as an Incremental, Opportunistic Process*, Proceedings of the 6th International Joint Conference on Artificial Intelligence, Los Altos, California: William Kaufman Inc., 1979, pp. 375-383.

[HAYE-85] Hayes-Roth B., *A Blackboard Architecture for Control*, Artificial Intelligence, vol.26, no.3, July 1985, pp. 251-321.

[HAYE-88a] Hayes-Roth B., Hewett M., *BB1: An Implementation of the Blackboard Control Architecture*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 297-313.

[HAYE-88b] Hayes-Roth B., et alia, *PROTEAN: Deriving Protein Structure from Constraints*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 417-431.

[HAYE-88c] Hayes-Roth B., Johnson M.V., Garvey A., Hewett M., *Building Systems in the BB* Environment*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 543-560.

[JAGA-87] Jagannathan V., Baum L., Dodhiawala R., *ERASMUS: Reconfigurable Object-Oriented Blackboard System*, Proceedings of the Second International Symposium on Methodologies for Intelligent Systems, Charlotte, 1987.

[JONE-88] Jones J., Millington M., Ross P., *A Blackboard Shell in Prolog*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 533-542.

[KLEI-76] Kleinrock L., *Queueing Systems*, Volume 2 - Computer Applications, New York, Wiley, 1976.

[LESS-88a] Lesser V.R., Erman L.D., *A Retrospective View of the Hearsay-II Architecture*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 87-122.

[LESS-88b] Lesser V.R., Corkill D.D., *The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 353-386.

[NAGA-88] Nagao M., Matsuyama T., Mori H., *Structural Analysis of Complex Aerial Photographs*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 219-230.

[NAKA-91c] Nakamiti G.S, *Blackboard - Uma Visão do Modelo e da Arquitetura*, submetido para publicação.

[NII -86a] Nii H.P., *Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures*, AI Magazine, vol.7, no.2, Summer 1986, pp. 38-53.

[NII -86b] Nii H.P., *Blackboard Application Systems, Blackboard Systems from a Knowledge Engineering Perspective*, AI Magazine, vol.7, no.3, August 1986, pp. 82-106.

[NII -88a] Nii H.P., Aiello N., *AGE (Attempt to Generalize): A Knowledge-Based Program for Building Knowledge-Based Programs*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 251-280.

[NII -88b] Nii H.P., Feigenbaun E. A., Anton J. J., Rockmore A. J., *Signal-to Signal Transformation: HASP/SIAP Case Study*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 135-157.

[NII -88c] Nii H.P., Aiello N., Rice J., *Frameworks for Concurrent Problem Solving: A Report on CAGE and POLIGON*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 475-501.

[REYN-88] Reynolds D., *MUSE: A Toolkit for Embedded, Real-time AI*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 519-532.

[RICE-89] Rice J. P., *Problems with Problem Solving in Parallel: The Polygon System*, Artificial Intelligence, Simulation & Modeling, editado por Widman, L. E., Loparo, K. A., e Norman, N. R., 1989.

[SRIR-89] Sriran D., et alia, *Knowledge-based System Applications in Engineering Desing: Research at MIT*, AI Magazine, vol.10 no.3, Fall 1989, pp. 79-96.

[TAIL-88] Tailor A., *MXA - A Blackboard Expert System Shell*, Blackboard Systems, Addison-Wesley Publ.Co., 1988, pp. 315-333.

[TERR-83] Terry A., *The CHRYSALIS Project: Hierarchical Control of Production Systems*, Technical Report HPP-83-19, Stanford University, Heuristic Programming Project, 1983.

[TERR-88] Terry A., *Using Explicit Strategic Knowledge to Control Expert Systems*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 159-188.

[WILL-88] Williams M.A., *Hierarchical Multi-Expert Signal Understanding*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 387-415.

[YANG-85] Yang J.D., Huhns M.N., Stephens L.M., *An Architecture for Control and Communications in Distributed Artificial Intelligence Systems*, IEEE Transactions on Systems, Man and Cybernetics, vol.SMC-15, no.3, May/June 1985.

[ZANC-88] Zanconato R., *Blobs - An Object-Oriented Blackboard System Framework for Reasoning in Time*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 335-345.

CAPÍTULO 3 - LINGUAGENS PARA IMPLEMENTAÇÃO

O desenvolvimento e implementação de **sistemas Blackboard** vêm sendo realizados principalmente através da utilização de linguagens como LISP e seus dialetos, citados por exemplo em [CORK-86, HAYE-88a, NII -88a, PEAR-88]; e PROLOG ([JONE-88, MASO-89]). O uso de regras de produção também pode ser usado, especialmente no que se refere às fontes de conhecimento, como citado em [TAIL-88].

Neste capítulo são apresentadas estas linguagens de programação e uma linguagem de sistemas de produção. O objetivo é fornecer subsídios para uma escolha racional para a implementação do protótipo.

3.1 LISP

A linguagem LISP (de LISt Programming) é baseada no artigo [McCA-60], de John McCarthy, considerado seu criador. Seu interesse deveu-se à constatação dos benefícios do processamento de listas para a Inteligência Artificial [CHAR-85].

A primeira versão, LISP 1, foi desenvolvida para computadores IBM. LISP 1.5, introduzido em 1962, foi o primeiro dialeto amplamente utilizado. Em 1964, LISP já possuía versões para vários tipos de computadores, e a partir de meados da década de 1960, começaram a surgir implementações incompatíveis entre si. [TOUR-90].

Trata-se da linguagem mais difundida na área de Inteligência Artificial, tendo relevante importância em aplicações como robótica, processamento de linguagem natural, prova de teoremas, diagnose médica, jogos e verificação de programas.

Essa grande difusão deve-se às seguintes características:
[MICR-81]

- a) LISP é uma linguagem aplicativa e recursiva, o que a torna um formalismo adequado para descrever conceitos matemáticos complexos;
- b) A árvore binária, sua principal estrutura de dados, pode ser isomórfica aos dados na maior parte dos problemas de Inteligência Artificial;
- c) LISP provê alocação dinâmica, facilitando consultas e buscas de dificuldade variável, e reciclagem de dados armazenados ("automatic garbage collection");
- d) Um ambiente altamente interativo é essencial para uma comunicação homem-máquina eficiente. A facilidade que permite que definições de funções LISP possam ser vistas como dados, encoraja um desenvolvimento incremental do sistema e torna LISP uma eficiente linguagem interativa.

LISP é uma linguagem muito flexível para a definição de novas funções por parte do programador. O código escrito em LISP pode ser dado de entrada de um programa LISP, o que permite, por exemplo, escrever em LISP um compilador LISP [CARN-88].

Graças à sua grande difusão e ampla história, possui inúmeros dialetos: IQLISP, MULISP, GCLISP, INTERLISP, COMMON LISP, dentre outros.

Em LISP, os objetos fundamentais são denominados *átomos*. Grupos de átomos formam *listas*. Listas podem ser agrupadas entre si e formar listas de nível superior. Os átomos e as listas são chamados coletivamente de *expressões simbólicas*, *S-expressões*, ou somente *expressões*. A possibilidade de formar grupos hierárquicos

é de fundamental importância para a linguagem. Um programa de manipulação simbólica utiliza expressões simbólicas para trabalhar com dados e procedimentos, e possui, tipicamente, procedimentos que reconhecem expressões particulares e as alteram [WINS -84]. Pelo conceito de "ordem superior", pode-se realizar abstrações sobre funções, fazendo com que sejam representadas por variáveis que sejam argumento de outras funções [TASI-88].

No caso geral das linguagens procedimentais, há uma grande diferença conceitual entre os programas e seus argumentos. Esses últimos (chamados *dados*) denotam endereços de memória com valores associados, enquanto que os programas representam uma transformação das associações entre endereços e valores. Nas linguagens onde haja esta heterogeneidade de significado entre programas e argumentos, o conceito de ordem superior não pode ser incorporado. Nas linguagens funcionais como LISP, todas as expressões denotam um valor, representando tanto o conceito de *dado* como o de *função* ou *programa*. A diferença é apenas uma questão de interpretação. Essa propriedade é uma característica das linguagens funcionais [TASI-88].

Os dialetos mais utilizados para sistemas Blackboard, segundo a bibliografia consultada são Interlisp ([ENGE-88d, NII -88a]) e Common Lisp ([CORK-86, HAYE-88a]).

3.2 PROLOG

PROLOG foi proposto em 1972 por Alain Colmerauer, na França, a partir de idéias de Robert Kowalski e outros autores. Colmerauer estava trabalhando nesta época com problemas relacionados com o processamento de linguagem natural.

Apesar do grande desenvolvimento de LISP na época em que surgiu, e da grande quantidade de ferramentas que existiam para facilitar o seu uso, PROLOG tem obtido grande êxito e aceitação [CARN-88]. Em 1983, o Japão publicou planos para um projeto nacional ambicioso, envolvendo o projeto e a produção de computadores de quinta geração, para os quais PROLOG foi escolhida como a linguagem fundamental (correspondendo ao uso da linguagem "assembly" nas arquiteturas tradicionais).

A idéia central da programação em lógica é sua utilização como uma linguagem prática de computação, onde os programas não indiquem *como* realizar um cálculo, mas *que* coisas são verdadeiras (hipóteses), e questões sejam vistas como teoremas que se deduzem a partir dessas hipóteses.

Alguns pontos tiveram particular importância para o projeto da linguagem: a restrição à regra de resolução, de maneira tal que o algoritmo gere uma estrutura dedutiva linear; e a restrição quanto ao tipo de fórmulas, utilizando o que se denominou cláusulas regulares, ou cláusulas de Horn.

De forma sucinta, podemos dizer que PROLOG é um sistema baseado no método de resolução linear aplicado a cláusulas de Horn, juntamente com uma interpretação particular da execução do sistema, que consiste em ver o processo de demonstração de teoremas como uma sucessão de invocações a procedimentos com argumentos de entrada, e retornando resultados como acontece nas linguagens de computação mais tradicionais [GOLD-86].

A programação lógica utiliza relações em lugar de funções, pois acredita-se que a programação através de relações seja mais flexível que através de funções. Informalmente, relações não têm senso de direção, ou seja, não possuem a distinção do que é computado a partir do quê [SETH-89].

PROLOG encontra vasta aplicação, cobrindo as seguintes áreas, dentre outras [CLOC-84, MEIR-88] :

- a) lógica matemática;
- b) resolução de problemas abstratos;
- c) resolução de equações simbólicas;
- d) análise de estruturas bioquímicas;
- e) implementação de bases de dados relacionais;
- f) processamento de linguagem natural;
- g) construção de sistemas especialistas e "shells";
- h) prova de teoremas;
- i) produção de protótipos em inúmeras áreas de aplicação;
- j) interpretadores;
- l) compiladores.

PROLOG é uma linguagem de programação baseada em lógica (de onde vem o seu nome). Em Programação Lógica, um programa é uma série de asserções (declarações), que ou são verdadeiras ou se quer tornar verdadeiras através do instanciamento de uma ou mais variáveis. O mecanismo de computação associado à linguagem, chamado de inferência, é que permite obter novos fatos sobre a área de aplicação a partir da base de conhecimento [MEIR-88].

Trata-se de uma implementação prática e eficiente de muitos aspectos de uma execução "inteligente" de programas, tais como não determinismo, paralelismo e chamada de procedimentos dirigida a padrões.

PROLOG provê uma estrutura de dados uniforme, chamada *termo*, sobre a qual todos os dados e programas são construídos. Um programa PROLOG consiste de um conjunto de cláusulas, onde cada

cláusula representa um fato ou uma regra sobre como o solução pode se relacionar ou ser inferida dos fatos [CLOC-84]. A utilização dessa forma de representação pode ser particularmente útil para a representação de relacionamentos entre diferentes tipos de entidades.

3.3 OPS5

Os sistemas de produção, ou sistemas baseados em regras, foram formulados inicialmente por Allen Newell e Herbert Simon como um modelo da arquitetura cognitiva humana. Representam o conhecimento como um conjunto de regras do tipo condição-ação (*produções*). Os sistemas de produção dão ênfase ao uso de módulos de conhecimento independentes. Idealmente, seria possível adicionar e retirar conhecimento (*produções*) sem que se preocupasse como as produções interagem [MOSK-86].

Este formalismo tem-se tornado muito popular, pois permite uma expressão natural do conhecimento em muitos de seus domínios. Além disso, a possibilidade de independência entre grupos de produções permite um certo grau de modularidade, desenvolvimento incremental do sistema, e eventual execução de certas partes em paralelo. No entanto, a independência entre os grupos pode acarretar falta de estrutura global, e a adição de regras pode dar lugar a interações indesejadas e resultados imprevisíveis, particularmente se a estratégia de resolução de conflitos for complexa. Além disso, estruturas de controle não são facilmente expressáveis, e seu custo computacional é alto [CARN-88, KVIT-88].

O ambiente de programação de um sistema de produção possui a seguinte arquitetura:

- a *memória de trabalho* contém uma coleção de fatos que descrevem o estado corrente do sistema sendo manipulado;
- a *base de regras* contém a coleção de regras do tipo condição -> ação;

- o *interpretador* é um programa supervisor que executa continuamente uma sequência de operações, chamada *ciclo de reconhecimento de ações*.

Numa primeira fase de reconhecimento, o interpretador lê a base de regras, compilando uma lista chamada *lista de conflitos* (conflict-set), com aquelas regras cujas condições "casam" com descrições da memória de trabalho. Na fase posterior, o interpretador realiza a chamada *resolução de conflitos*, isto é, escolhe uma regra da *lista de conflitos*. Finalmente, na terceira fase, também conhecida como *set-phase*, o interpretador executa as ações correspondentes à regra escolhida. As ações envolvem mudanças na memória de trabalho. A seguir, o interpretador inicia um novo ciclo de reconhecimento de ações, até que alguma condição de término seja encontrada.

OPS5 é a linguagem de sistemas de produção mais difundida para o desenvolvimento de sistemas em Inteligência Artificial. Foi criada por Forgy no início da década de 1980, destinada ao desenvolvimento de sistemas especialistas.

Trata-se de uma linguagem genérica, podendo ser utilizada em vários tipos de aplicação, com diferentes estilos de programação. Embora a execução de suas regras seja realizada somente "forward chaining", pode-se, por exemplo, implementar sobre ela uma estratégia "backward" [CARN-88].

Um programa OPS5 é dividido em duas seções: a seção de declarações e a seção de produções. Na primeira, são definidos os *elementos de dados*, que comporão a memória de trabalho e declaradas as funções externas chamadas. A seção de produções contém as regras [VELA-88b].

Não é feita distinção entre fatos (informações imutáveis) e dados (informações dinâmicas). Ambos residem na memória de trabalho. A diferença está em que os fatos não são alterados no curso da execução. A memória de trabalho e a memória de regras constituem a base de conhecimento utilizada para resolver o

problema [VELA-88a]. Aos elementos da memória de trabalho são associados "time-tags", valores determinados pelo sistema e usados para avaliar as idades relativas dos fatos da memória. São importantes para a resolução de conflitos.

O ciclo de reconhecimento de ações é relativamente rápido em OPS5, por usar uma tabela pré-compilada que relaciona nomes de regras às condições. Quando uma série de dados é modificada durante a "set-phase" de um ciclo, OPS5 pode derivar a lista de conflitos para o próximo ciclo. Para tanto, utiliza a atual lista de conflitos e testa somente as regras cujos dados foram modificados.

As estruturas de dados não são tão genéricas quanto se poderia desejar. OPS5 não possui, por exemplo, listas dentro de listas.

3.4 CONSIDERAÇÕES

LISP é, sem dúvida, a linguagem de programação que mais vem sendo utilizada na implementação de sistemas Blackboard. Isso porque é uma linguagem muito flexível e eficiente, razoavelmente portátil, e disponível para inúmeras máquinas.

Apesar das características que a têm difundido, trata-se de uma linguagem de baixo nível, quando comparada a PROLOG e OPS5. Em [PRES-87], sugere-se que para a prototipação sejam utilizados ambientes de programação ou linguagens de mais alto nível, como forma de depuração de requisitos e obtenção mais rápida do sistema.

A maior limitação encontrada com a linguagem OPS5 é a restrição que impõe à estruturação de dados. A versão disponível é não comercial, além de bastante recente e muito pouco testada no departamento.

A linguagem escolhida foi PROLOG, por ser suficientemente genérica e adequada à construção de protótipos. Além disso, PROLOG é uma linguagem aplicável à representação de relações e à implementação de bases de dados relacionais, como citado anteriormente. No Capítulo 4, veremos que esta sua característica teve importância relevante na implementação do protótipo. Dentre os compiladores existentes (ARITY, IF e TURBO PROLOG, todos para IBM-PC compatíveis), a opção foi pelo compilador ARITY por ser um bom ambiente de programação, suficientemente testado e disseminado, e disponível em máquinas de fácil acesso.

REFERÊNCIAS

- [CARN-88] Carnota R.J., Teszkiewicz A.D., *Sistemas Expertos y Representación del Conocimiento*, EBAI, 1988.
- [CHAR-85] Charniak E., McDermott, D., *Introduction to Artificial Intelligence*, Addison-Wesley Publ. Co., 1985.
- [CLOC-84] Clocksin W.F., Mellish C.S., *Programming in Prolog*, Springer-Verlag, 1984.
- [CORK-86] Corkill D., Gallagher K., Murray K., *GBB: A Generic Blackboard Development System*, Proceedings of the National Conference on Artificial Intelligence, Philadelphia, August 1986, pp. 1008-1014. Também em *Blackboard Systems*, Addison-Wesley Publ. Co., 1988, pp. 503-517.
- [ENGE-88d] Engelmores R. S., Morgan A.J., *Recent Applications (1981-1985)*, *Blackboard Systems*, Addison-Wesley Publ. Co., 1988, pp. 349-352.
- [GOLD-86] Goldsztein M., Carnota R., *Inteligência Artificial Aplicada - Lógica y Representación del Conocimiento*, EBAI - Editora da Unicamp, 1986.
- [HAYE-88a] Hayes-Roth B., Hewett M., *BB1: An Implementation of the Blackboard Control Architecture*, *Blackboard Systems*, Addison-Wesley Publ. Co., 1988, pp. 297-313.
- [JONE-88] Jones J., Millington M., Ross P., *A Blackboard Shell in Prolog*, *Blackboard Systems*, Addison-Wesley Publ. Co., 1988, pp. 533-542.
- [KVIT-88] Kvitca A.M., *Resolución de Problemas con Inteligência Artificial*, Ebai, 1988.

- [MASO-89] Mason K.P., Hood S.T., *KBESM - A Prolog Blackboard System*, The Australian Computer Journal, vol. 21, no.2, August 1989, pp. 100-107.
- [McCA-60] McCarthy J., *Recursive Functions of Symbolic Expressions and their Computation by Machine*, CACM, vol. 3, no. 4, 1960, pp. 184-195.
- [MEIR -88] Meira S., *Introdução à Programação Funcional*, VI Escola de Computação, Unicamp, 1988.
- [MICR-81] MULISP/MUSTAR-80, Manual, Microsoft Corporation, 1981.
- [MOSK-86] Moskowitz L., *Rule-Based Programming*, Byte, vol.11, no.12, November 1986, pp. 217-224.
- [NII -88a] Nii H.P., Aiello N., *AGE (Attempt to Generalize): A Knowledge-Based Program for Building Knowledge-Based Programs*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 251-280.
- [PEAR-88] Pearson G., *Mission Planning within the Framework of the Blackboard Model*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 433-442.
- [PRES-87] Pressman R.S., *Software Engineering - A Practitioner's Approach*, McGraw-Hill Book Co., 1987.
- [SETH-89] Sethi R., *Programming Languages - Concepts and Constructs*, Addison-Wesley Publ. Co., 1989.
- [TAIL-88] Tailor A., *MXA - A Blackboard Expert System Shell*, Blackboard Systems, Addison-Wesley Publ.Co., 1988, pp. 315-333.
- [TASI-88] Tasistro A., Vidart J., *Programación Lógica y Funcional*, EBAI, 1988.

[TOUR-90] Touretzky D.S., *Common LISP: A Gentle Introduction to Symbolic Computation*, The Benjamin/Cummings Publ. Co., 1990.

[VELA-88a] Velasco F.R., *Uma Introdução à Linguagem OPS5*, INPE-Departamento de Processamento de Imagens, 1988.

[VELA-88b] Velasco F.R., *Manual da Linguagem OPS5 para Micro-Computadores*, versão 1.0, INPE-Departamento de Processamento de Imagens, Outubro de 1988.

[WINS-84] Winston P.H., Horn B.K., *LISP*, Addison-Wesley Publ. Co., 1984.

CAPÍTULO 4 - O SISTEMA ASTUTO

ASTUTO (Arquitetura para a Resolução de Problemas através do Uso de Técnicas de IA e Oportunismo) traz minha proposta de ferramenta para o desenvolvimento de sistemas Blackboard, que procura enfatizar características relacionais de sua estrutura. Nesta abordagem, procuro retomar alguns aspectos essenciais do modelo, introduzido por Newell [NEW-62] e condensado em [ENGE-88h].

4.1 INTRODUÇÃO

O objetivo de um sistema Blackboard é obter a(s) solução(ões) para um determinado problema através da expansão de "ilhas de solução" contidas no blackboard propriamente dito.

A visão que as fontes de conhecimento têm do blackboard é de particular importância para que possam intervir produtivamente no processo de resolução do problema. Elas devem ter acesso às ilhas de solução e aos detalhes nelas contidos. Para tanto, é desejável que haja formas de representação que encapsulem detalhes em entidades de maior poder semântico. Além disso, é importante que conheçam a forma como as ilhas estão (ou podem estar) relacionadas entre si, visando a resolução global para o problema. O modelamento do blackboard deve ser flexível e semanticamente representativo o bastante para prover estas características ao projeto.

4.2 O MODELO ENTIDADE-RELACIONAMENTO

O Modelo Entidade-Relacionamento (MER), desenvolvido por Chen [CHEN-76] surgiu para dar maior flexibilidade e contornar problemas semânticos na modelagem de bancos de dados, até então por demais presa à implementação física. Possibilitou, assim, a independência entre projeto abstrato e implementação.

Um diagrama do Modelo Entidade-Relacionamento é constituído basicamente por:

- a) *entidades*, que representam conceitos reais ou abstratos sobre os quais desejamos armazenar dados;
- b) *relacionamentos*, que são associações que indicam como as entidades estão relacionadas.

Além disso, podem ser associados atributos às entidades e aos relacionamentos. Esses atributos denotam qualidades ou quantidades, e auxiliam na descrição de entidades e relacionamentos. Podem ainda haver cardinalidades associadas aos relacionamentos. Elas fornecem uma indicação numérica de quantas entidades de um tipo podem estar relacionadas a entidades de um outro tipo.

Desde a concepção inicial do Modelo Entidade-Relacionamento, vários pesquisadores vêm se preocupando em dar-lhe maior poder semântico, quer acrescentando-lhe facilidades para especificação de restrições de integridade, quer possibilitando-lhe representar objetos em vários níveis de abstração. Com essas modificações, o uso do MER tem se ampliado para a modelagem de sistemas em geral. Em [WAGN-87] encontramos uma proposta de padronização para estas modificações, visando facilitar seu uso integrado.

Na modelagem de blackboards, o nível de confiança das hipóteses e as expectativas de ocorrência de eventos da aplicação podem ser representados tanto em termos dos objetos (entidades) quanto de seus relacionamentos, através de atributos, por exemplo.

Pode-se também representar facilmente medidas de proximidade entre entidades.

A seguir, são mostrados alguns exemplos de uso do Modelo Entidade-Relacionamento, que ilustram algumas das mais importantes contribuições que este modelo recebeu. Nas figuras, não são representados todos os atributos e as cardinalidades associadas por motivos de clareza.

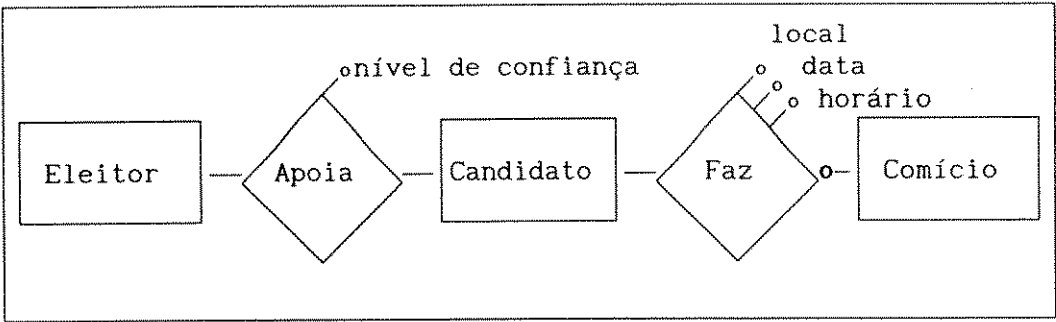


Figura 4.1 - Exemplo de Modelagem

Na Figura 4.1, um *eleitor* apoia um *candidato* com um determinado nível de confiança. Poderia também, por exemplo, ser representado seu apoio a partir de uma certa data. Existe ainda a expectativa de um *comício* ser realizado pelo candidato em determinado horário. O círculo escuro na figura indica que *comício* é uma entidade fraca em relação ao relacionamento, ou seja, sem que o *candidato* o fizesse, não haveria *comício*. Essa dependência de existência não é representada em função da entidade, pois *comício* poderia estar ligado a outras entidades.

O uso de agregação e generalização, proposto em [SMIT-77] torna possível a visualização do sistemas em diferentes níveis de abstração, bem como a unificação com o modelo de redes semânticas [SHAP-71, SCHU-79] e hierarquias IS-A [CHAR-85], por exemplo.

Na Figura 4.2, temos um exemplo de agregação. A entidade *fornecimento* é um agregado formado pelo relacionamento entre as entidades *fornecedor* e *peças*.

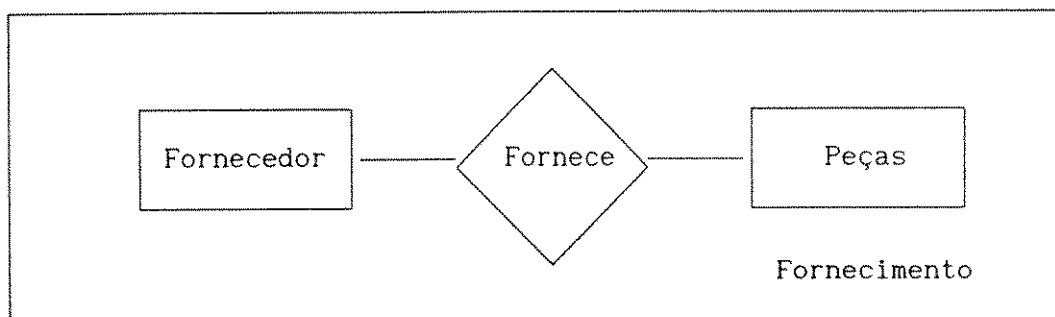


Figura 4.2 - Exemplo de Agregação

A generalização é um dos mecanismos mais importantes para a conceituação do mundo real. Podemos associar características às generalizações, dando-lhes maior poder semântico. Elas podem ter cobertura total ou parcial, conforme representem totalmente ou parcialmente a entidade de nível superior. Podem ainda ser exclusivas, se o relacionamento se limitar a uma única entidade do nível inferior, ou sobrepostas ("overlapping"), caso contrário. Na Figura 4.3 ilustramos estes casos.

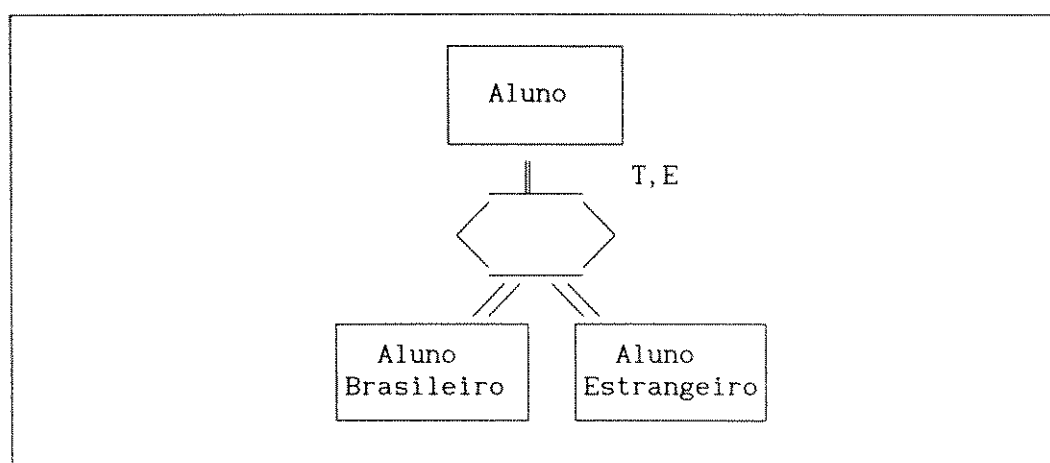


Figura 4.3a - Exemplo de Generalização

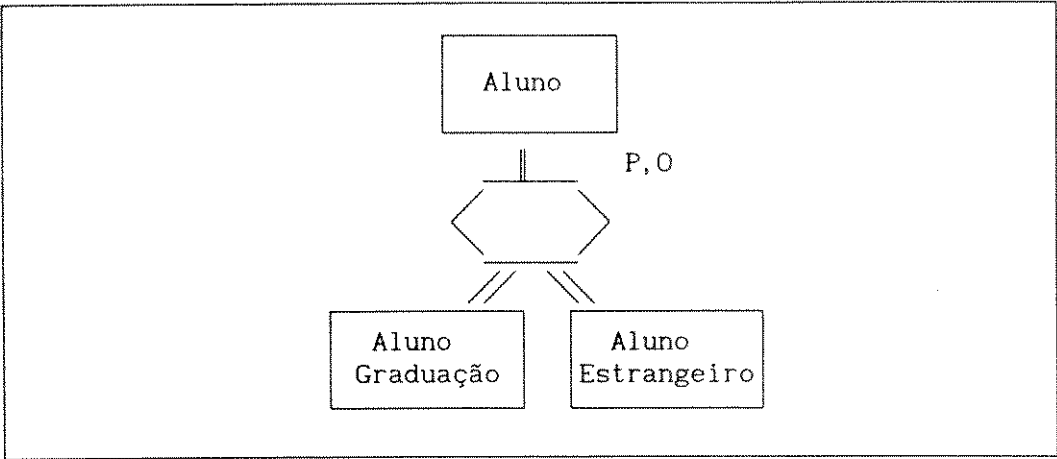


Figura 4.3b - Outro Exemplo de Generalização

O uso de generalizações pode ser estendido para relacionamentos, como é mostrado na Figura 4.4. Considera-se que seja possível praticar esportes profissionalmente, recreativamente, ou ambos.

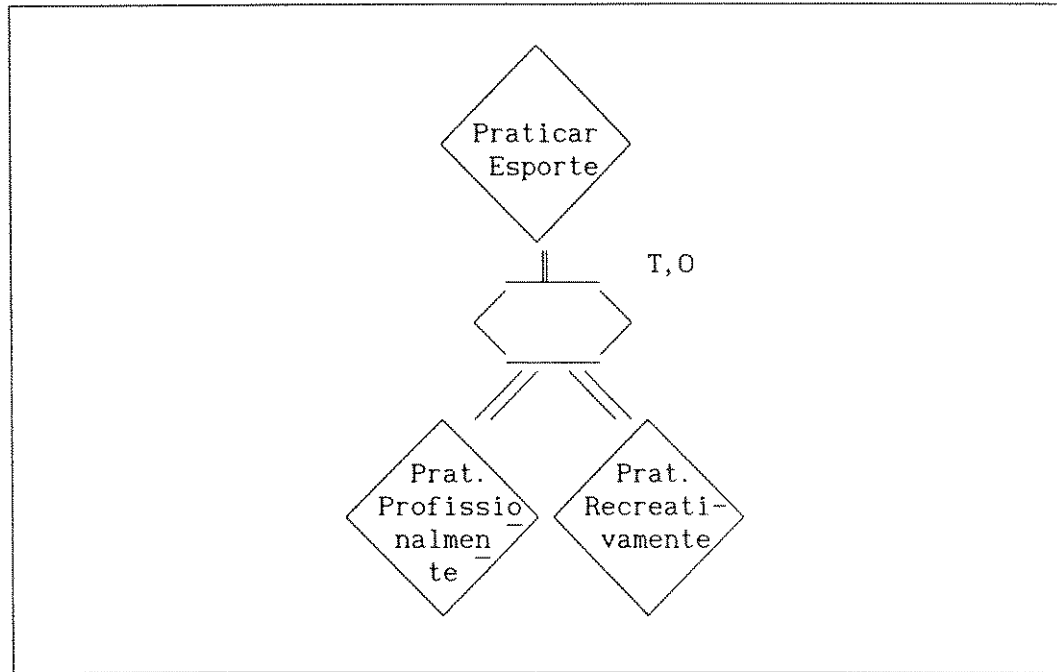


Figura 4.4 - Generalização de Relacionamentos

Uma outra extensão ao MER, o conceito de transparência parcial [WAGN-87] pode permitir um aumento de flexibilidade na modelagem de sistema Blackboard, auxiliando também na eliminação de redundâncias. Através da transparência parcial, permite-se que níveis diferentes de abstração não sejam totalmente transparentes entre si. Sempre que necessário, uma determinada abstração pode ter conhecimento de elementos de outra.

A utilização do Modelo Entidade-Relacionamento em sistemas Blackboard traz grande poder de representação, e a flexibilidade desejada em seu modelamento.

4.3 ABORDAGEM QUANTO AO COMPORTAMENTO

O uso do Modelo Entidade-Relacionamento, apesar de bastante útil no modelamento de sistemas Blackboard, não é suficiente para nos aproximarmos do modelo proposto inicialmente. A analogia do modelo Blackboard com a montagem do quebra-cabeças, presente em [ENGE-88a], nos traz muito de sua essência e do oportunismo intrínseco. As pessoas devem ter visão global e detalhada do estado da solução para que melhor possam contribuir. Além disso, alguns acontecimentos podem alterar significativamente sua visão do problema e caminhos de raciocínio. Isso pode ocorrer, por exemplo, quando duas ou mais porções do quebra-cabeças (ilhas de solução) se unem em uma única. É interessante que as fontes de conhecimento possam ser alertadas sobre fatos relevantes, para que possam se adequar mais rapidamente ao estado atual do problema. Um mecanismo de controle dirigido por eventos pode viabilizar a reprodução desse tipo de comportamento.

Nos ambientes computacionais seriais, o módulo de controle deve também gerenciar a execução das fontes de conhecimento. A convivência de um componente de controle em um sistema Blackboard tem se mostrado bastante delicada, infringindo até mesmo a sua essência, como discutido em [NII -88c]. E ainda, em um sistema de

processamento paralelo, a execução concorrente das fontes de conhecimento não provê muito ganho no tempo de processamento se o controle permanecer centralizado. Por outro lado, descentralizando o controle ou removendo-o completamente, cria-se um ambiente computacional no qual é muito difícil controlar o comportamento de resolução de problemas e obter uma solução razoável [NII -88c].

Desta forma, a opção feita foi por um comportamento do sistema dirigido por eventos, onde o módulo de controle atua somente como disciplinador, e não como um escalonador para a atuação das fontes de conhecimento. Na Figura 4.5, observamos o esquema conceitual do funcionamento de sistemas desenvolvidos a partir de ASTUTO. Nele, as fontes de conhecimento atuam através de primitivas de vários tipos. O tipo de primitivas utilizado determina a estratégia de resolução de problemas empregada a cada passo. As estratégias podem ser, por exemplo, top-down, bottom-up, e inside-outside. As primitivas operam sobre entidades e relacionamentos presentes no blackboard a nível físico. Suas ações geram eventos e fornecem base para o módulo de controle supervisionar a execução das fontes de conhecimento.

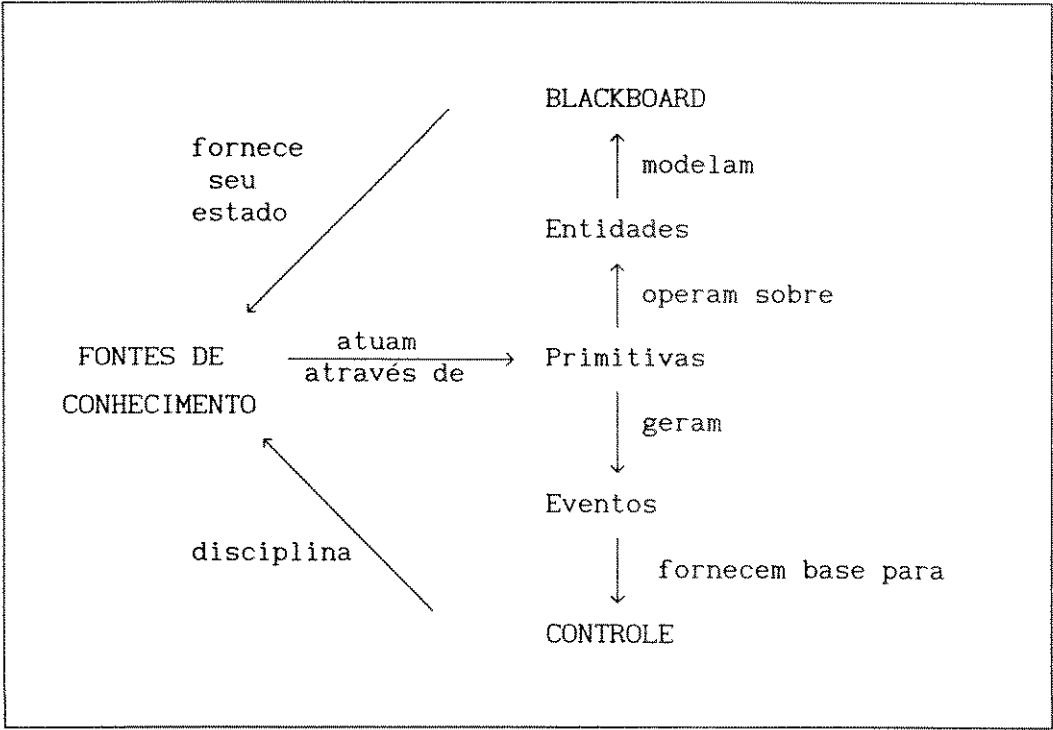


Figura 4.5 - Esquema Conceitual do Sistema ASTUTO

4.4 ESPECIFICAÇÃO DO SISTEMA

O objetivo do sistema ASTUTO é se constituir em uma ferramenta através da qual se possa desenvolver e experimentar técnicas e teorias no domínio da aplicação. Pretende-se propiciar a obtenção de uma melhor compreensão do domínio da aplicação através de sua modelagem, e facilitar o entendimento dos princípios de resolução de problemas nele aplicáveis. Algumas características importantes para o sistema dizem respeito à versatilidade quanto ao domínio da aplicação, modularidade e facilidade de manutenção e extensão.

Para atingir esses quesitos, utilizamos recursos da metodologia SADT- Structured Analysis and Design Technique, apresentada em [SOFT-76]. Trata-se de uma metodologia bastante difundida. SADT é a metodologia mais citada em [LEIT-87], que envolve mais de cento e vinte referências. Em [CERI-86], além de ressaltar a larga utilização dessa metodologia, ressalta a importância da modelagem associada a uma representação gráfica. Em [YADA-89], fala-se da facilidade do aprendizado de SADT, quando comparada a outras "técnicas" de análise de requisitos. Já em [BLAN-83], critica-se a necessidade de um esforço considerável para a aquisição do conhecimento SADT. O SADT, apesar de sua popularidade, possui vários pontos fracos, como apresentado em [TOLE-89]. Falta, por exemplo, um suporte para experimentação (elaboração de protótipos), que permita a validação das especificações antes do avanço para outras etapas do desenvolvimento [LEIT-87].

Não foram seguidos todos os passos da metodologia; em particular, o ciclo autor-comentarista, realizado pela suposta equipe de projeto. Foram seguidos somente aqueles que permitissem ou auxiliassem a exprimir, entender, manipular e verificar opções para a especificação do sistema e o desenvolvimento do protótipo. Tabelas auxiliares foram incorporadas a esta especificação para facilitar o entendimento do projeto.

Como opção de projeto, foi bastante utilizado o princípio de ocultamento de informação, bastante discutido na literatura (por exemplo, em [PARN-72, PRES-87]), e apresentado em [CORK-87] como recurso importante para se obter, por exemplo, características de flexibilidade em sistemas Blackboard.

4.5 O BLACKBOARD EM ASTUTO

No Capítulo 3, bem como em [NAKA-91d], menciona-se o fácil mapeamento entre um modelo baseado em relações, como é o Modelo Entidade-Relacionamento e a linguagem PROLOG. Isto levou à implementação do sistema e, em particular, à representação do blackboard, através das estruturas dessa linguagem. Assim, as entidades e os relacionamentos são representados por fatos PROLOG. Esses fatos compõem o blackboard e permanecem na base de dados manipulada pelo interpretador.

Entidades para representar um restaurante e um clube poderiam, por exemplo, ter a forma:

estabelecimento(roma, restaurante, (rua_manaus, 935), (11, 22), 40).

estabelecimento(regatas, clube, (av_parana, 462), (8, 20), 90).

, onde "estabelecimento" seria o nome geral para todos os estabelecimentos; "restaurante" e "clube" representariam o tipo dos estabelecimentos, estando também presentes seus nomes, endereços, horários de funcionamento e o tempo esperado de permanência neles.

Abaixo, é dado um exemplo de relacionamento que mostra o trajeto entre o clube e o restaurante. Deixa-se o clube às 11:30, toma-se a avenida Paraná, segue-se pela rua Guanabara, e chega-se ao restaurante pela rua Manaus. O tempo de percurso é de dez minutos, e o tempo de permanência no destino é de quarenta minutos. A Figura 4.6 ilustra este relacionamento.

ir(regatas, roma, 11.30, 10, 40, [av_parana, rua_guanabara, rua_manaus]).

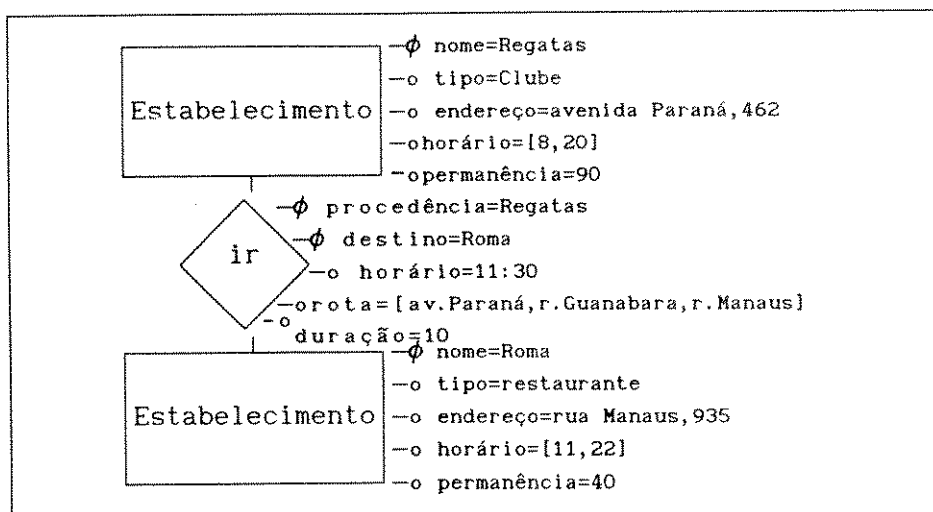


Figura 4.6 - Ilustração de Relacionamento

Os fatos (entidades e relacionamentos) presentes no blackboard são alterados pela ação das Fontes de Conhecimento, implementadas por regras Prolog.

4.6 OPERAÇÃO SOBRE O BLACKBOARD

Para solucionar o problema, as Fontes de Conhecimento agem modificando o blackboard. O procedimento das fontes é determinado por suas primitivas de ação que, por sua vez, seguem uma estratégia de resolução de problemas.

Desta forma, poderíamos ter Fontes de Conhecimento implementando primitivas top-down, de modo a:

a) gerar soluções alternativas.

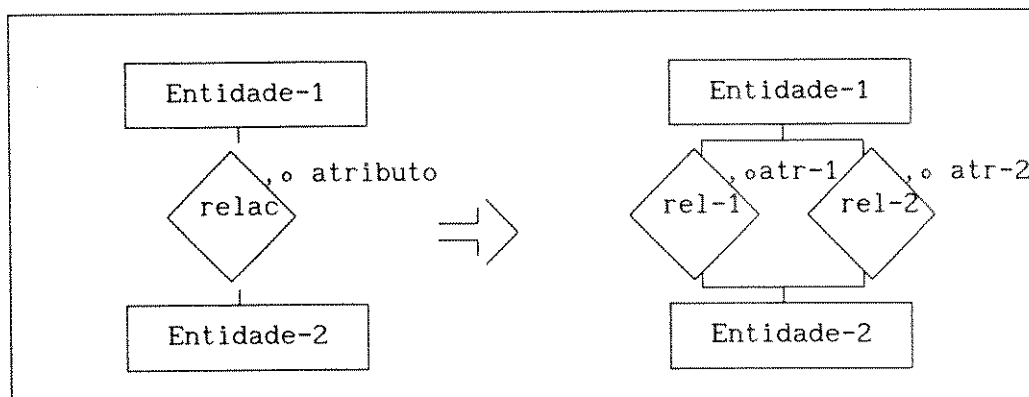


Figura 4.7 - Geração de Soluções Alternativas

Pode haver mais de um caminho de resolução que nos leve a uma solução satisfatória para o problema. Muitas vezes é desejável manter as soluções alternativas, como ilustrado na Figura 4.7.

b) classificar entidades.

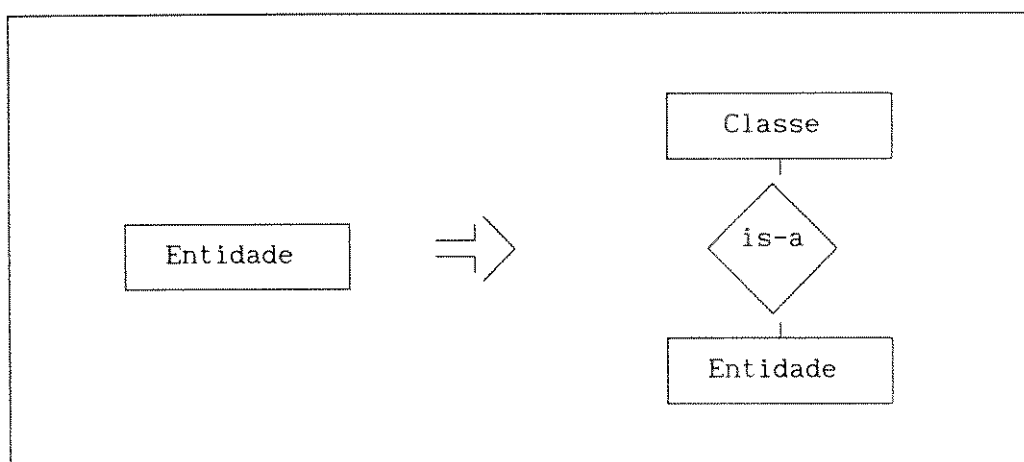


Figura 4.8 - Classificação de Entidades

Através da classificação de entidades, pode-se passar mais facilmente de um nível de abstração para outro, facilitando o raciocínio e a tomada de decisões (Figura 4.8)

c) dividir relacionamentos.

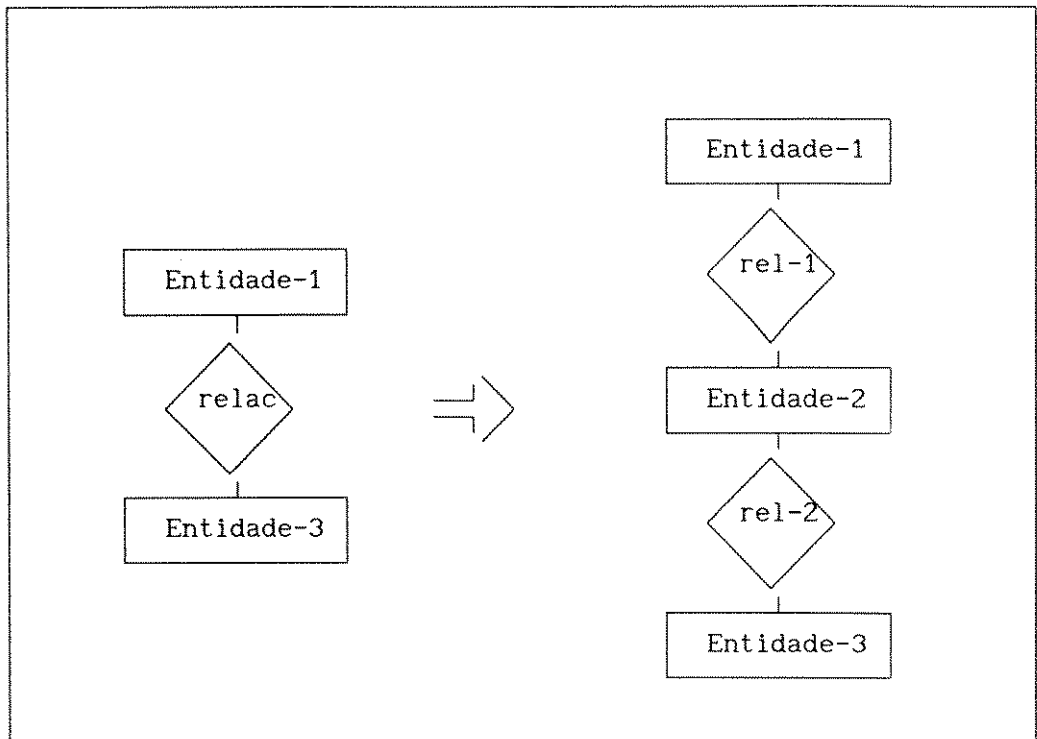


Figura 4.9 - Divisão de Relacionamento

Podem existir entidades estreitamente relacionadas com outras. Muitas vezes, a representação dessa proximidade é desejável, e isto pode ser obtido através de divisão de relacionamentos (Figura 4.9).

Poderíamos ter também Fontes de Conhecimento implementando primitivas bottom-up, de modo a:

a) gerar entidades.

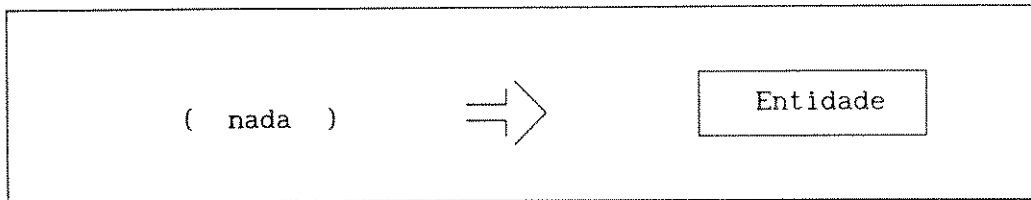


Figura 4.10 - Geração de Entidade

Novas entidades passam a existir a partir da ocorrência de eventos externos, por exemplo. Primitivas de geração de entidades (Figura 4.10) inicializam sua existência no âmbito do sistema.

b) gerar relacionamentos.

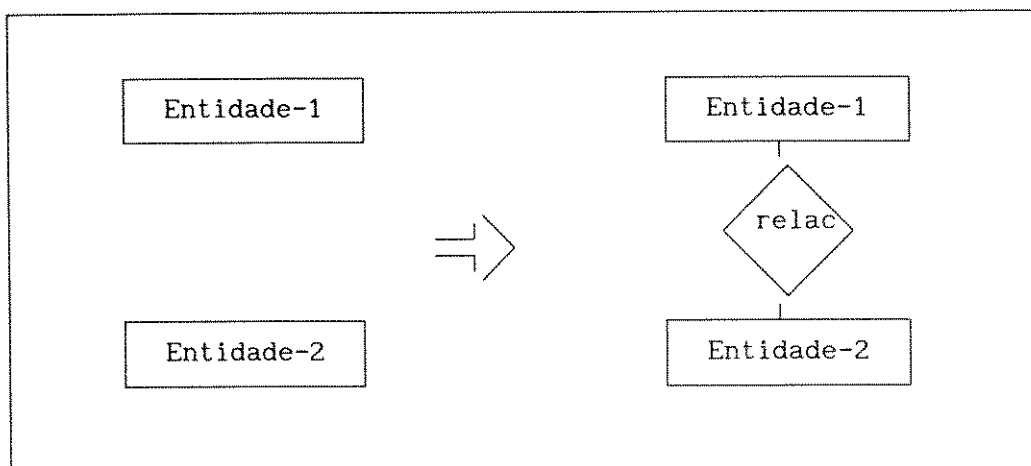


Figura 4.11 - Geração de Relacionamento

A geração de relacionamentos (Figura 4.11), além de contribuir para a construção da solução, é também uma maneira muito eficaz de melhorar o raciocínio das Fontes de Conhecimento, fornecendo a elas uma visão melhor do estado da solução.

c) gerar hierarquias.

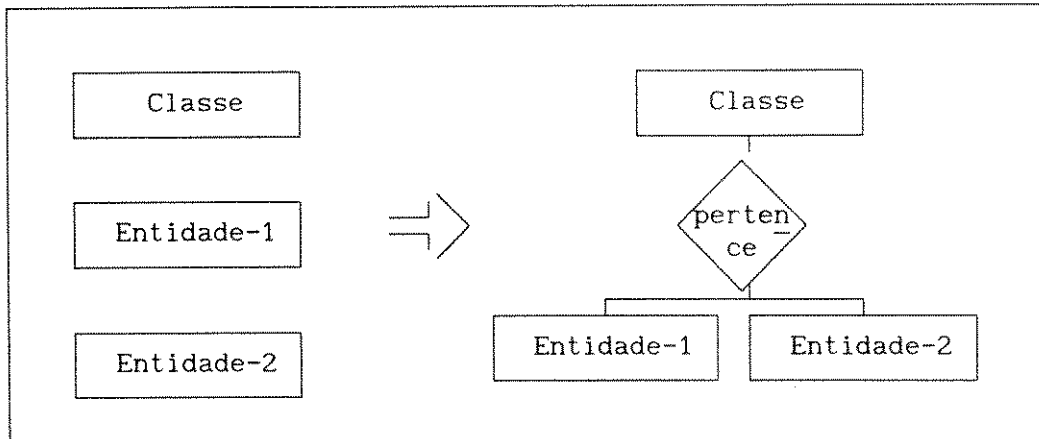


Figura 4.12 - Geração de Hierarquia

A geração de hierarquias (Figura 4.12) permite que o sistema se abstraia de detalhes quando raciocina em níveis de abstração mais altos.

4.7 ESPECIFICAÇÃO DO SISTEMA

Nesta seção, é apresentada a especificação lógica de ASTUTO, composta pela definição de seus módulos em SADT, das tabelas de descrição e inter-relação de seus dados e fluxos, de sua arquitetura de controle e de sua definição em PDL.

4.7.1 DEFINIÇÃO DOS MÓDULOS

No nível mais alto, ASTUTO é composto por três módulos executados sequencialmente, como mostrado na Figura 4.13.

O módulo *Inicializar Sistema* tem por objetivo adequar a representação da aplicação à representação interna. Nele, são

definidas as estruturas de dados. São também estabelecidos os critérios gerais de funcionamento, dadas as peculiaridades do domínio da aplicação e as preferências e objetivos do usuário.

O módulo *Realizar Resolução Problema* é responsável pela obtenção da(s) solução(ões) para o problema proposto através do paradigma blackboard. Essa(s) solução(ões) é(são) então apresentada(s) ao usuário pelo módulo *Informar Resultado Processamento*.

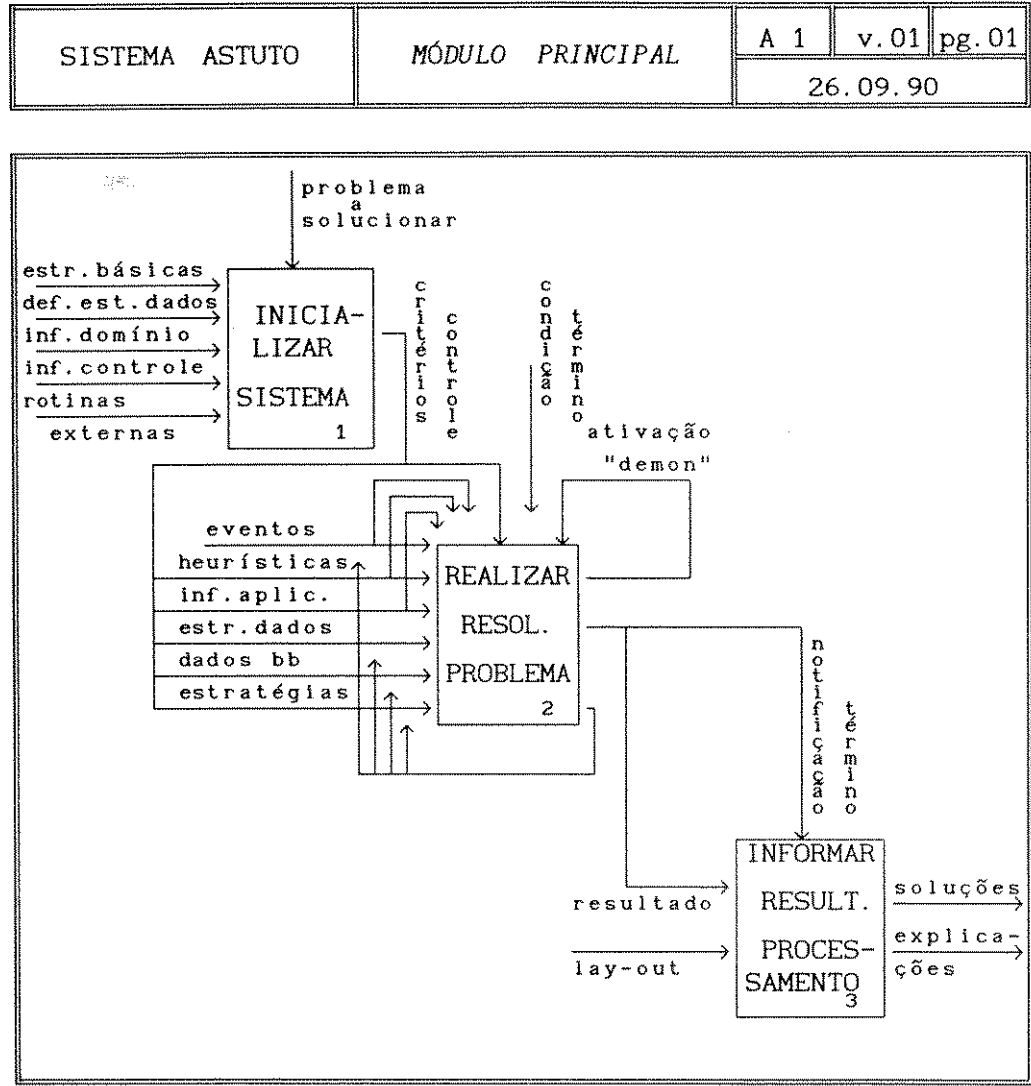


Figura 4.13 - SADT de Primeiro Nível - MÓDULO PRINCIPAL

MÓDULO INICIALIZAR SISTEMA

O módulo *Inicializar Sistema* (Figura 4.14) realiza o pré-processamento, gerando e inicializando as estruturas básicas do sistema, além de tratar informações particulares da aplicação e do usuário.

Dado um problema, são definidos os dados e as estruturas de dados básicas para sua resolução. Isso é feito pelo módulo *Definir Estruturas Básicas*. Algumas das informações particulares da aplicação poderão já fazer parte do blackboard, como é o caso de expectativas. Outras podem referir-se a verdades do domínio da aplicação, regras de funcionamento, ou restrições. Esse tipo de informação pode ser útil para o direcionamento das estratégias de resolução, e são tratadas pelo módulo *Tratar Particularidades Domínio*.

O módulo *Estabelecer Critério Controle* é responsável por analisar informações estratégicas, quer sejam particulares da aplicação, quer sejam estratégias básicas de funcionamento. Sua função se limita a compatibilizar a adoção do paradigma Blackboard com sua implementação em máquinas seriais.

Em todos esses módulos, as informações tratadas podem ser externas ao sistema, tornando-o aberto a fatos e regras de outros sistemas que também utilizem a linguagem PROLOG.

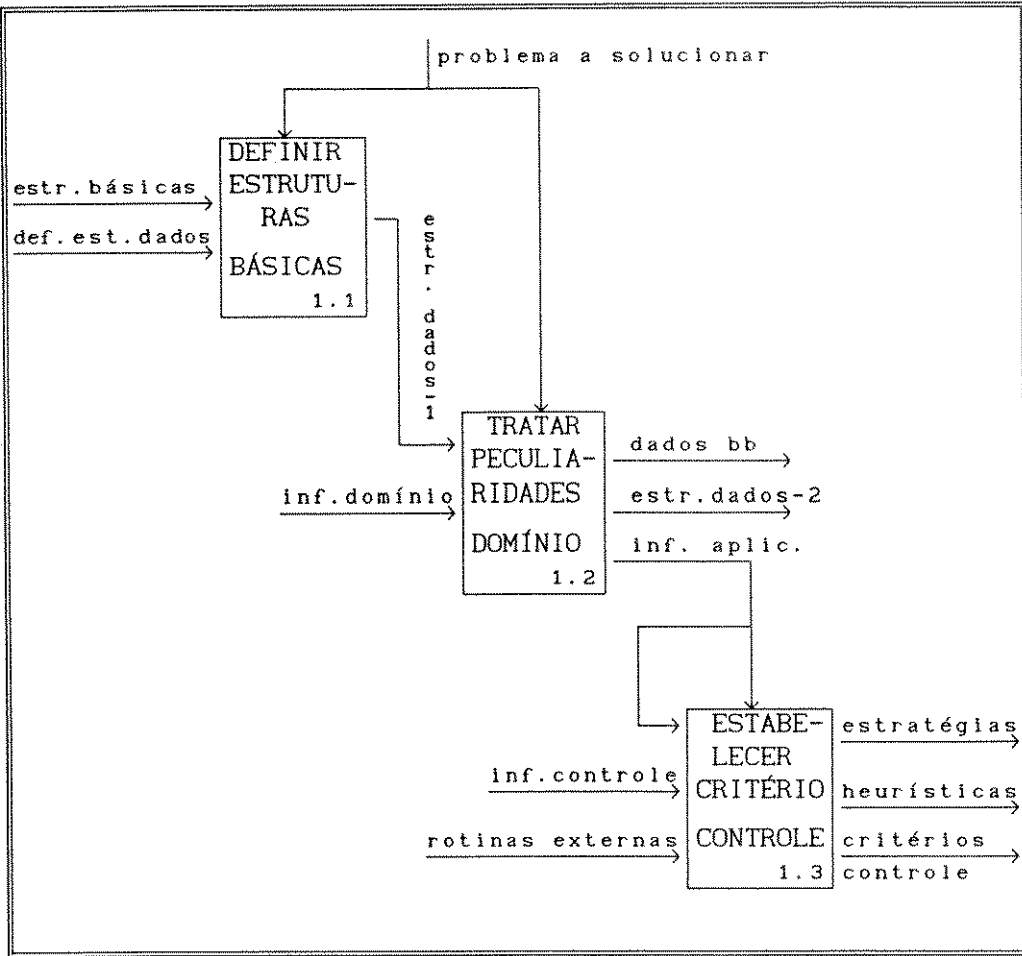


Figura 4.14 - SADT de Segundo Nível - INICIALIZAR SISTEMA

MÓDULO REALIZAR RESOLUÇÃO PROBLEMA

O objetivo deste módulo (Figura 4.15) é obter uma solução para o problema através da expansão de "ilhas de solução". Essas ilhas são compostas de dados e estão armazenadas no blackboard.

O módulo *Analisar Estado Problema* identifica estados de término, isto é, quando a solução foi alcançada ou uma solução final não parece factível. Nestes casos, notifica o sub-sistema *INFORMAR RESULTADO PROCESSAMENTO*. Quando não for identificado

estado de término, produz informações estratégicas a partir do estado da solução, que possam auxiliar no processo de resolução.

Podem ocorrer eventos relevantes para a resolução do problema, como a chegada de um dado que estava sendo esperado. Várias fontes de conhecimento habilitam-se para agir. O módulo *Disciplinar Fontes Executar* intervém no processo, de acordo com as prioridades de controle, designando uma fonte de conhecimento para ser executada. Essa decisão conterà também alguns critérios de execução que podem permitir, por exemplo, que uma fonte ative outra fonte, dentro do módulo *Executar Fonte*.

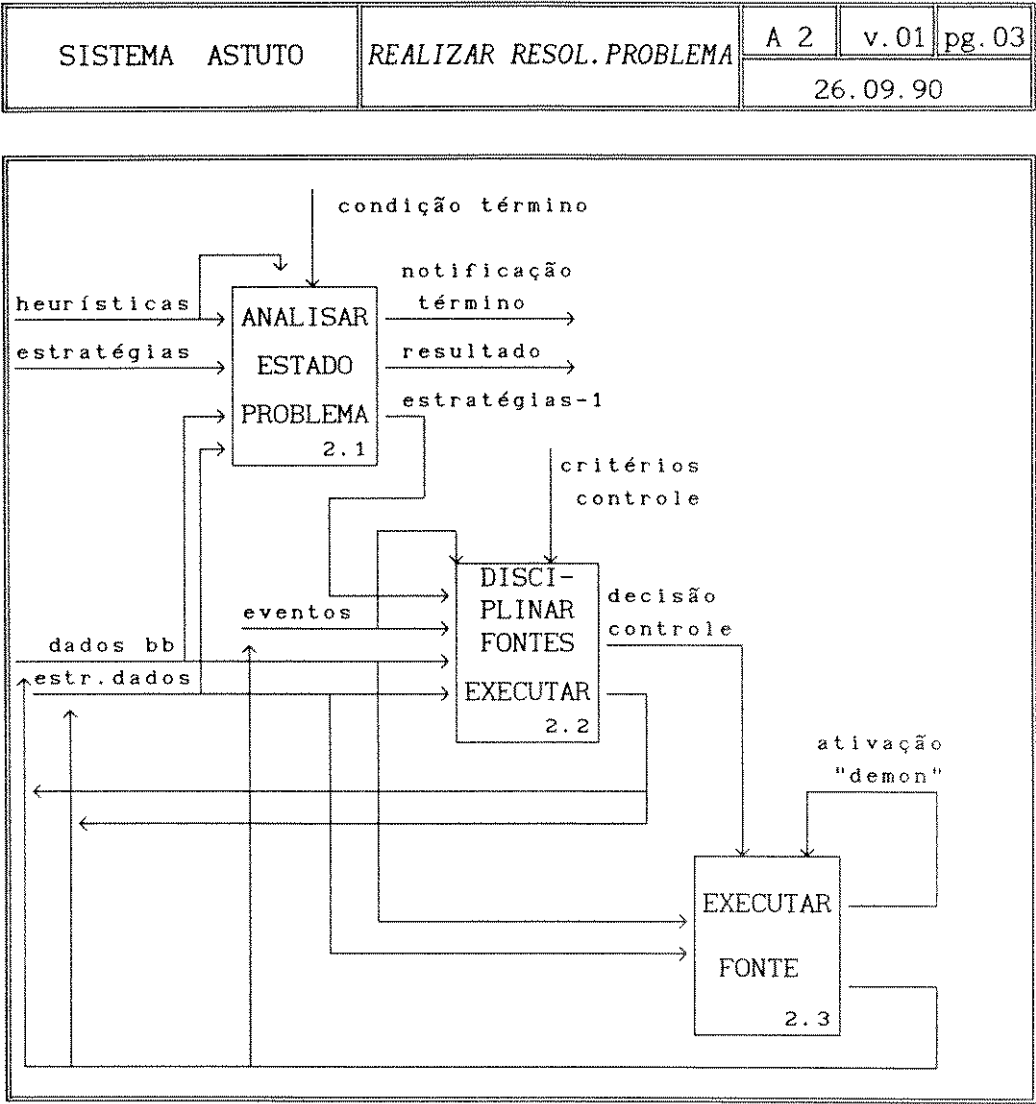


Figura 4.15 - SADT de Segundo Nível - REALIZAR RESOLUÇÃO PROBLEMA

MÓDULO INFORMAR RESULTADO PROCESSAMENTO

O módulo *Informar Resultado Processamento*, mostrado na Figura 4.16 apresenta a(s) solução(ões) ao usuário.

O módulo *Analisar Resultado* verifica a(s) solução(ões) encontrada(s) e monta-a(s) a partir das estruturas de dados processadas. No módulo *Informar Usuário*, essas informações são rearranjadas e impressas.

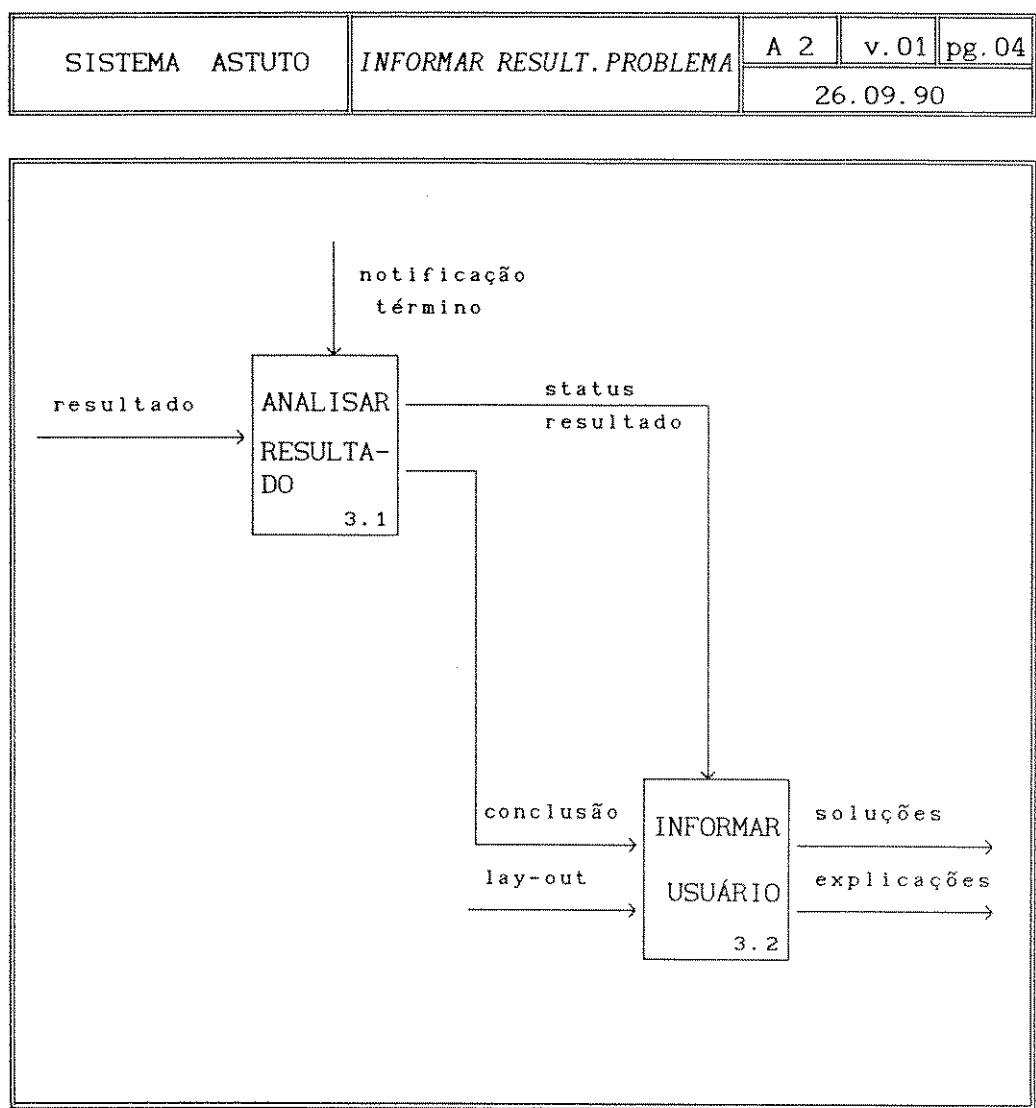


Figura 4.16-SADT de Segundo Nível-INFORMAR RESULTADO PROCESSAMENTO

4.7.2 TABELAS DE DESCRIÇÃO DE DADOS

Nas Figuras 4.17 a 4.20 são apresentadas tabelas e relações existentes entre a informação de controle, a entrada de dados e a saída produzida. Existem também textos associados para facilitar o entendimento das tabelas.

São apresentadas também a descrição dos fluxos de dados e de controle presentes nos diagramas. Essas descrições encontram-se nas Figuras 4.21 a 4.32 e contêm também as estruturas de dados neles contidas.

SISTEMA ASTUTO	RELAÇÃO CONTROLE-ENTRADA-SAÍDA	R 1	v. 01	pg. 05
		26. 09. 90		

INICIALIZAR SISTEMA		
CONTROLE	ENTRADA	SAÍDA
problema a solucionar	estruturas básicas	estruturas dados
problema a solucionar	definição estruturas de dados	informações aplicação
informações aplicação	estruturas dados	dados blackboard
	informações domínio	heurísticas
	inf. controle	critérios controle
	rotinas externas	
A definição das estruturas de dados que serão manipuladas é feita a partir das estruturas genéricas básicas que o sistema manipula independentemente do problema e do domínio da aplicação. Existem informações do domínio e da aplicação que se referem a essas estruturas. Essas informações são incorporadas a elas e ao blackboard em particular, na forma de entidades e relacionamentos. Os critérios básicos de controle são determinados a partir de informações gerais e específicas da aplicação. Rotinas externas podem ser incorporadas para o tratamento de situações específicas ou melhorar o desempenho do sistema.		

Figura 4.17 - Relação Controle- Entrada- Saída - 01

SISTEMA ASTUTO	RELAÇÃO CONTROLE-ENTRADA-SAÍDA	R 1	v. 01	pg. 06
		26. 09. 90		

REALIZAR RESOLUÇÃO PROBLEMA		
CONTROLE	ENTRADA	SAÍDA
condição término	———	notificação término
condição término	estruturas dados dados blackboard	resultado
heurísticas	heurísticas inf.aplic. estruturas dados dados blackboard	estratégias
eventos	eventos	estruturas dados dados blackboard
eventos	eventos estratégias	decisão controle
critérios controle	heurísticas estruturas dados dados blackboard	decisão controle
decisão controle	———	eventos estruturas dados dados blackboard [ativação "demon"]
ativação "demon"	———	eventos estruturas dados dados blackboard [ativação "demon"]

Figura 4.18 - Relação Controle- Entrada- Saída - 02

SISTEMA ASTUTO	RELAÇÃO CONTROLE-ENTRADA-SAÍDA	R 1	v. 01	pg. 07
		26. 09. 90		

REALIZAR RESOLUÇÃO PROBLEMA

Quando uma condição de término é satisfeita, é emitida uma notificação. Além disso, é elaborado o resultado do processamento a partir das estruturas de dados e do blackboard em particular.

Podem ser geradas estratégias de controle em função de heurísticas e do estado do problema. Essas informações fornecem também subsídios para disciplinar as fontes de conhecimento executáveis em um dado momento. Esse tipo de decisão é associado aos critérios de controle gerado pelo módulo *Estabelecer Mecanismo Controle*.

Todos os eventos são tratados, e quando forem relevantes podem implicar na geração de decisões de controle. As decisões de controle são também geradas a partir de heurísticas e do estado do problema. Essas decisões levam à ativação de fontes de conhecimento que atuarão através de primitivas (eventos) sobre as entidades e relacionamentos fisicamente presentes no blackboard. De acordo com as decisões de controle, as fontes de conhecimento podem ativar-se mutuamente, sem que seja necessária a participação de outras formas de controle.

Figura 4.19 - Relação Controle- Entrada- Saída - 03

SISTEMA ASTUTO	RELAÇÃO CONTROLE-ENTRADA-SAÍDA	R 1	v. 01	pg. 08
		26. 09. 90		

INFORMAR RESULTADO PROBLEMA		
CONTROLE	ENTRADA	SAÍDA
notificação término	resultado	status resultado conclusão
status resultado	conclusão lay-out	soluções explicações
Dada a notificação de término, é necessário distinguir se o sistema obteve ou não solução(ões) para o problema. A este parâmetro está associada sua conclusão. A apresentação da(s) solução(ões) e explicações ao usuário segue um formato pré-estabelecido.		

Figura 4.20 - Relação Controle- Entrada- Saída - 04

SISTEMA ASTUTO	DESCRIÇÃO FLUXO DE DADOS/CONTROLE	F 1	v. 01	pg. 09
		26.09.90		

Nome do fluxo: ativação "demon"

Alias: Pag.Ref.: 01,03

Descrição: Ativação de uma fonte de conhecimento por outra fonte de conhecimento.

Estruturas contidas:

- fonte de conhecimento
- critérios de liberdade (estabelecem, por exemplo, se as fontes podem ou não ativar-se mutuamente).

Nome do fluxo: conclusão

Alias: Pag.Ref.: 04

Descrição: Resultado obtido pelo sistema.

Estruturas contidas:

- blackboard
- lista de explicações

Figura 4.21 - Descrição dos Fluxos de Dados e de Controle - 01

SISTEMA ASTUTO	DESCRIÇÃO FLUXO DE DADOS/CONTROLE	F 1	v. 01	pg. 10
		26.09.90		

Nome do fluxo: condição término Alias: Pag.Ref.: 01,03 Descrição: Critério de final de programa. Pode significar que uma ou mais soluções foram encontradas, ou que parece inviável obter uma solução. Estruturas contidas: (função ou regra que funcione como critério)
Nome do fluxo: dados blackboard Alias: dados bb Pag.Ref.: 01,02,03 Descrição: Trata-se do blackboard propriamente dito, com todos os dados nele contidos. Estruturas contidas: entidades relacionamentos

Figura 4.22 - Descrição dos Fluxos de Dados e de Controle - 02

SISTEMA ASTUTO	DESCRIÇÃO FLUXO DE DADOS/CONTROLE	F 1	v. 01	pg. 11
		26.09.90		

Nome do fluxo: decisão controle
Alias: *Pag.Ref.:* 03
Descrição: Fonte de conhecimento que será executada.

Estruturas contidas:
 critérios de liberdade

Nome do fluxo: definição estruturas de dados
Alias: def.est.dados *Pag.Ref.:* 01,02
Descrição: Definição das estruturas de dados particulares do domínio. Serão mapeadas com *estruturas básicas* pelo módulo 1.1 - Definir Estruturas Básicas.
Estruturas contidas:
 entidades
 relacionamentos

Figura 4.23 - Descrição dos Fluxos de Dados e de Controle - 03

SISTEMA ASTUTO	DESCRÇÃO FLUXO DE DADOS/CONTROLE	F 1	v. 01	pg. 12
		26.09.90		

Nome do fluxo: estratégias

Alias: estratégias-1 Pag.Ref.: 01,03

Descrição: Informações que fornecem subsídios para disciplinar as fontes de conhecimento executáveis em um dado instante.

Estruturas contidas:

(algoritmos)

(heurísticas)

Nome do fluxo: estruturas básicas

Alias: estr.básicas Pag.Ref.: 01,02

Descrição: Definição das estruturas de dados básicas.

Estruturas contidas:

blackboard	lista de fontes de conhecimento "ready"
lista de eventos	
lista de expectativas	lista de explicações

Figura 4.24 - Descrição dos Fluxos de Dados e de Controle - 04

SISTEMA ASTUTO	DESCRÇÃO FLUXO DE DADOS/CONTROLE	F 1	v. 01	pg. 14
		26.09.90		

Nome do fluxo: explicações

Alias: Pag.Ref.: 01,04

Descrição: explicações fornecidas ao usuário

Estruturas contidas:
(formato de apresentação de explicação)

Nome do fluxo: heurísticas

Alias: Pag.Ref.: 01,02,03

Descrição: Informações que auxiliam no estabelecimen-
to das estratégias de controle.

Estruturas contidas:
(podem estar, por exemplo, na forma de algoritmos
ou regras).

Figura 4.26 - Descrição dos Fluxos de Dados e de Controle - 06

SISTEMA ASTUTO	DESCRIÇÃO FLUXO DE DADOS/CONTROLE	F 1	v. 01	pg. 15
		26.09.90		

Nome do fluxo: informações aplicação
Alias: inf.aplic. *Pag.Ref.:* 01,02
Descrição: Informações associadas à aplicação, que poderão contribuir para estabelecer critérios para disciplinar a atuação das fontes de conhecimento.
Estruturas contidas:
 (algoritmos)
 (heurísticas)

Nome do fluxo: informações controle
Alias: inf.controle *Pag.Ref.:* 01,02
Descrição: Critérios básicos de disciplina para as fontes de conhecimento.
Estruturas contidas:
 (algoritmos)
 (heurísticas)

Figura 4.27 - Descrição dos Fluxos de Dados e de Controle - 07

SISTEMA ASTUTO	DESCRIÇÃO FLUXO DE DADOS/CONTROLE	F 1	v. 01	pg. 16
		26.09.90		

Nome do fluxo: informações domínio

Alias: inf.domínio Pag.Ref.: 01,02

Descrição: Informações referentes à aplicação.

Estruturas contidas:

- eventos (heurísticas)
- expectativas
- fontes de conhecimento "ready"

Nome do fluxo: lay-out

Alias: Pag.Ref.: 04

Descrição: Formato de apresentação de resultados ao usuário.

Estruturas contidas:

(formato de lay-out)

Figura 4.28 - Descrição dos Fluxos de Dados e de Controle - 08

SISTEMA ASTUTO	DESCRIÇÃO FLUXO DE DADOS/CONTROLE	F 1	v. 01	pg. 17
		26.09.90		

Nome do fluxo: mecanismo controle

Alias: Pag.Ref.: 01,02,03

Descrição: Informações para disciplinar as fontes de conhecimento executáveis em um dado momento.

Estruturas contidas:
(pode estar, por exemplo, na forma de algoritmos ou regras).

Nome do fluxo: notificação término

Alias: Pag.Ref.: 01,03,04

Descrição: Aviso de final de processamento.

Estruturas contidas:
informe de final

Figura 4.29 - Descrição dos Fluxos de Dados e de Controle - 09

SISTEMA ASTUTO	DESCRIÇÃO FLUXO DE DADOS/CONTROLE	F 1	v. 01	pg. 18
		26.09.90		

Nome do fluxo: problema a solucionar

Alias: Pag.Ref.: 01,02

Descrição: Notificação de controle, que informa a presença de um problema e dispara a execução do sistema.

Estruturas contidas:
 identificação do problema a solucionar

Nome do fluxo: resultado

Alias: Pag.Ref.: 01,03,04

Descrição: Seleção das informações necessárias para fornecer a conclusão ao usuário.

Estruturas contidas:
 blackboard
 lista de explicações

Figura 4.30 - Descrição dos Fluxos de Dados e de Controle - 10

SISTEMA ASTUTO	DESCRIÇÃO FLUXO DE DADOS/CONTROLE	F 1	v. 01	pg. 19
		26.09.90		

Nome do fluxo: rotinas externas
Alias: *Pag.Ref.:* 02
Descrição: Algoritmos externos (ex. estratégias de busca para tratar situações específicas ou melhorar o desempenho do sistema).
Estruturas contidas:
(algoritmos)
(heurísticas)

Nome do fluxo: soluções
Alias: *Pag.Ref.:* 01,04
Descrição: Soluções obtidas e apresentadas ao usuário.

Estruturas contidas:
(formato de apresentação de solução)

Figura 4.31 - Descrição dos Fluxos de Dados e de Controle - 11

SISTEMA ASTUTO	DESCRIÇÃO FLUXO DE DADOS/CONTROLE	F 1	v. 01	pg. 20
		26.09.90		

Nome do fluxo: status resultado

Alias: Pag.Ref.: 04

Descrição: Distingue se o sistema alcançou ou não soluções para o problema, e em caso positivo, a quantidade delas.

Estruturas contidas:

status

Nome do fluxo:

Alias: Pag.Ref.:

Descrição:

Estruturas contidas:

Figura 4.32 - Descrição dos Fluxos de Dados e de Controle - 12

4.7.3 ARQUITETURA DE CONTROLE DO SISTEMA

É apresentada aqui a arquitetura de controle do sistema ASTUTO e seu projeto procedural, especificado através de Program Design Language (PDL). Segundo [CAIN-75], que introduziu a linguagem por vezes descrita como PDL, uma linguagem de projeto de programa (PDL) é um dialeto que usa o vocabulário de uma linguagem (como o inglês) e a sintaxe de uma outra (uma linguagem de programação estruturada) [PRES-87].

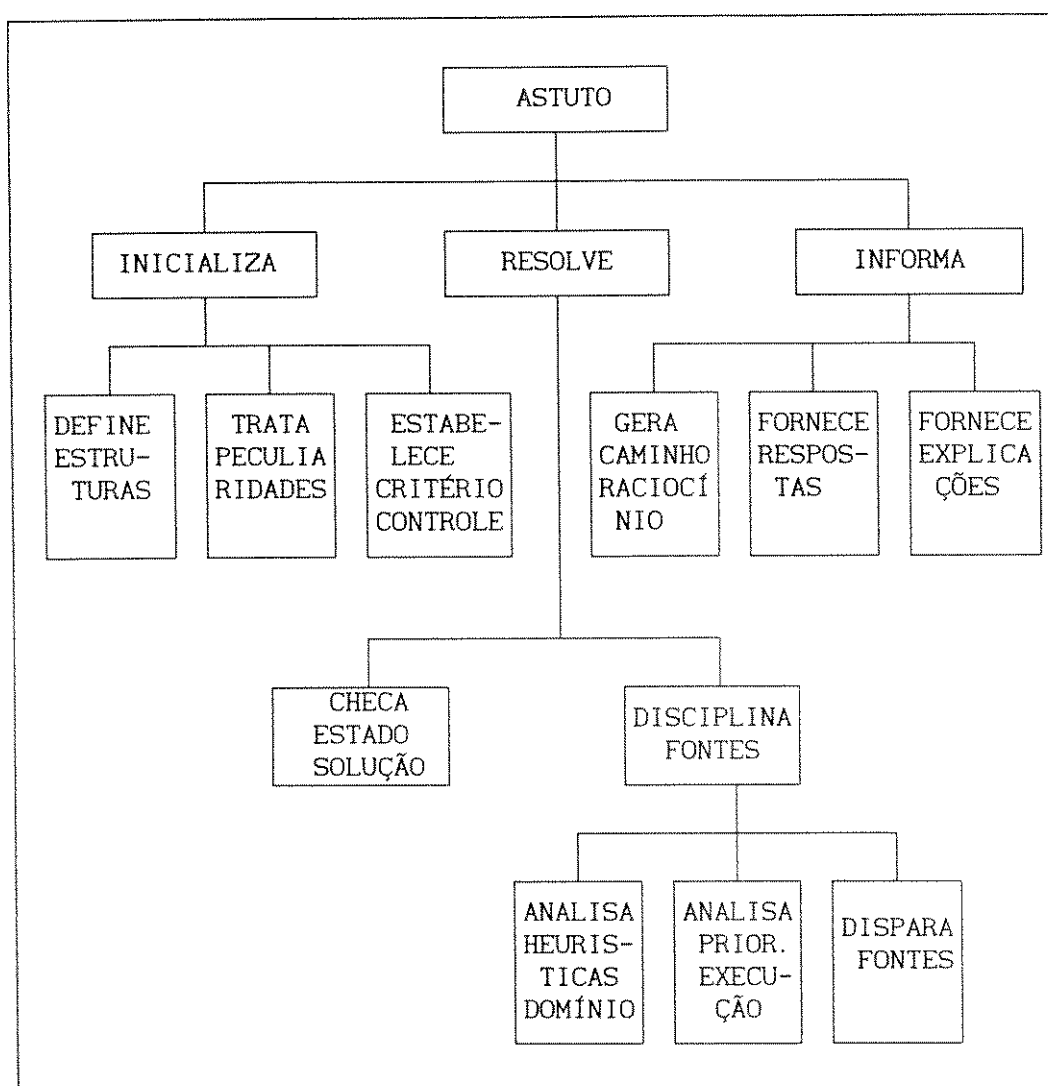


Figura 4.33 - Arquitetura de Controle do Sistema ASTUTO

Na Figura 4.33 é mostrada a arquitetura hierárquica de controle de ASTUTO.

4.7.4 ASTUTO EM PDL

A seguir, os módulos são apresentados em PDL.

```
PROGRAM astuto;
```

```
%-----  
%|              DEFINIÇÃO DE TIPOS              |  
%-----
```

```
% Definição de par atributo-valor
```

```
TYPE atr-val IS STRUCTURE
```

```
    atributo: STRING;
```

```
    valor: STRING;
```

```
END atr-val TYPE
```

```
% Definição de lista de pares atributo-valor
```

```
TYPE pair-list IS LIST OF
```

```
    atr-val
```

```
END pair-list TYPE
```

```
% Definição de intervalo de tempo
```

```
TYPE intervalo IS STRUCTURE
```

```
    horário-inicial: SCALAR;
```

```
    horário-final: SCALAR;
```

```
END intervalo TYPE
```

% Definição de entidade do blackboard

```
TYPE entidade-blackboard IS LIST OF
    pair-list
END entidade-blackboard TYPE
```

% Definição de relacionamento do blackboard

```
TYPE relacionamento-blackboard IS STRUCTURE
    entidade1 IS INSTANCE OF entidade-blackboard;
    entidade2 IS INSTANCE OF entidade-blackboard;
    parametros IS INSTANCE OF pair-list;
END relacionamento-blackboard TYPE
```

% Definição de evento

```
TYPE evento-blackboard IS STRUCTURE
    tipo-evento: STRING;
    parâmetros IS INSTANCE OF pair-list;
END evento-blackboard TYPE
```

% Definição de fonte de conhecimento

```
TYPE fonte-conhecimento IS STRUCTURE
    tipo-estratégia: STRING;
    nome-fonte: STRING;
    parâmetros IS INSTANCE OF pair-list;
END fonte-conhecimento TYPE
```

% Definição de expectativa

```
TYPE expectativa IS STRUCTURE
    entidade IS INSTANCE OF entidade-blackboard;
    certeza: SCALAR;
    horário IS INSTANCE OF intervalo;
END expectativa TYPE
```

% Definição de explicação

TYPE explicação IS LIST OF

acontecimento: STRING;

comentário: STRING;

END explicação TYPE

% Definição da estrutura de um blackboard

TYPE blackboard IS LIST OF

entidade-blackboard;

relacionamento-blackboard;

END blackboard TYPE

% Definição de lista de eventos

TYPE lista-eventos IS LIST OF

evento-blackboard

% Definição de lista de fontes de conhecimento (aptas a executar)

TYPE lista-fontes-ready IS LIST OF

fonte-conhecimento

% Definição de lista de expectativas

TYPE lista-expectativas IS LIST OF

expectativa

% Definição de lista de explicações

TYPE lista-explicações IS LIST OF

explicação

```
%-----  
%:          CHAMADA DOS PROCEDIMENTOS          :  
%-----
```

```
CALL inicializa;  
CALL resolve;  
CALL informa;
```

END astuto

```
%-----  
%:          PROCEDIMENTOS          :  
%-----
```

```
% Procedimento que realiza o pré-processamento do sistema, gerando  
% suas estruturas básicas e tratando informações particulares da  
% aplicação
```

PROCEDURE inicializa;

```
CALL define-estruturas;  
CALL trata-peculiaridades;  
CALL estabelece-critério-controle;
```

END inicializa

```
% Procedimento que dá início às estruturas básicas do sistema
```

PROCEDURE define-estruturas;

```
inicializa blackboard;  
inicializa lista de eventos;  
inicializa lista de fontes "ready"  
inicializa lista de expectativas;  
inicializa lista de explicações;
```

END define-estruturas

% Procedimento que trata as peculiaridades do domínio da aplicação

PROCEDURE trata-peculiaridades;

adequa blackboard ao domínio;
adequa lista de eventos ao domínio;
adequa lista de fontes "ready" ao domínio;
adequa lista de expectativas ao domínio;
adequa lista de explicações ao domínio;

END trata-peculiaridades

% Procedimento que estabelece o mecanismo de controle

PROCEDURE estabelece-critério-controle;

analisa critérios gerais para sistemas blackboard;
analisa heurísticas domínio;
gera critérios resolução;

END estabelece-critério-controle

% Procedimento para obtenção da solução para o problema

PROCEDURE resolve;

TYPE status: BOOLEAN;

status=false;
DO WHILE NOT status
 CALL checa-estado-solução;
 CALL disciplina-fontes;
ENDDO

END resolve

% Procedimento que checa estados de término

PROCEDURE checa-estado-solução;

analisa condições término;
analisa estado problema;
IF estado satisfaz condições término
 THEN status=true;
ENDIF

END checa-estado-solução

% Procedimento disciplinador da ativação das fontes de
% conhecimento

PROCEDURE disciplina-fontes;

TYPE prioridade: STRING;

CALL analisa-heurísticas-domínio;
CALL analisa-prioridades-resolução;
CALL dispara-fontes WITH prioridade;

END disciplina-fontes

% Procedimento que dá informações particulares do domínio da
% aplicação para disciplinar as fontes de conhecimento

PROCEDURE analisa-heurísticas-domínio;

analisa estado solução;
analisa heurísticas resolução;
gera(prioridade);

END analisa-heurísticas-domínio

% Procedimento que verifica se a prioridade de execução das fontes
% é adequada

PROCEDURE analisa-prioridades-execução;

TYPE incompatível: BOOLEAN;

 verifica(prioridade);
 determina se prioridade é compatível e desejável;
 IF incompatível
 THEN estabelece nova prioridade;
 ENDIF

END analisa-prioridades-execução

% Procedimento que dispara as fontes de conhecimento para serem
% executadas

PROCEDURE dispara-fontes(prioridade);

CASE OF prioridade:
 WHEN prioridade é uma fonte
 dispara fonte;
 WHEN prioridade é uma estratégia
 escolhe uma fonte com essa estratégia;
 dispara fonte;
 WHEN prioridade é um evento
 trata evento;
 WHEN prioridade é uma expectativa
 trata expectativa;
 DEFAULT none;
ENDCASE

END dispara-fontes

REFERÊNCIAS

- [BLAN-83] Blank J., Krijger M.J., *Software Engineering: Methods and Techniques*, Wiley-Interscience Publishing, 1983.
- [CAIN-75] Caine S., Gordon K., *PDL - A Tool for Software Design*, Proceedings of the National Computer Conference, AFIPS Press, 1975, pp. 271-276.
- [CERI-86] Ceri S., *Requirements Collection and Analysis in Information Systems Design*, Elsevier Science Pub., North-Holland, 1986, pp. 205-214.
- [CHEN-76] Chen P.P., *The Entity-Relationship Model: Toward a Unified View of Data*, ACM Transactions on Database Systems, vol.01, no.01, 1976.
- [CORK-87] Corkill D., Gallagher K., Johnson P., *Achieving Flexibility, Efficiency, and Generality in Blackboard Architectures*, Proceedings of the Sixth National Conference on Artificial Intelligence, 1987, pp. 18-23.
- [ENGE-88a] Engelmores R.S., Morgan A.J., *Introduction*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 01-22.
- [ENGE-88h] *Blackboard Systems*, editado por Robert Engelmores e Tony Morgan, Addison-Wesley Publ. Co., 1988.
- [LEIT-87] Leite J.C., *A Survey on Requirements Analysis*, Tech. Report, University of California at Irvine, 1987.
- [NAKA-91d] Nakamiti G.S., *ASTUTO: Modelling Blackboards Through an Entity-Relationship Approach*, submetido para publicação.

[NEWE-62] Newell A., *Some Problems of the Basic Organization in Problem-Solving Programs*, Proceedings of the Second Conference on Self-Organizing Systems, editado por Yovits, M.C., Jacobi, G.T, e Goldstein, G.D., Spartan Books, 1962, pp. 393-323.

[NII -88c] Nii H.P., Aiello N., Rice J., *Frameworks for Concurrent Problem Solving: A Report on CAGE and POLIGON*, Blackboard Systems, Addison-Wesley Publ. Co., 1988, pp. 475-501.

[PARN-72] Parnas D.L., *On Criteria to be Used in Decomposing Systems into Modules*, CACM, vol.15, no.12, 1972, pp. 1053-1058.

[PRES-87] Pressman R.S., *Software Engineering - A Practitioner's Approach*, McGraw-Hill Book Co., 1987.

[SCHU-79] Schubert L.K., Goebel R.G., Cercone N.J. *The Structure and Organization of a Semantic Net for Comprehension and Inference*, Associative Networks - Representation and Use of Knowledge by Computers, Academic Press Inc., 1979, pp. 121-175.

[SHAP-71] Shapiro S.C., *A Net Structure for Semantic Information Storage, Deduction and Retrieval*, Advance Papers of the Second International Joint Conference on Artificial Intelligence, 1971, pp. 512-523.

[SMIT-77] Smith J.M., Smith D.C., *Database Abstractions: Aggregation and Generalization*, ACM Transactions on Database Systems, vol.02, no.02, 1977.

[SOFT-76] SofTech Inc., *An Introduction to SADT*, Manual, 1976.

[TOLE-89] Toledo C.M., *ANA-RE - um método para ANÁLise e especificação de REquisitos*, Tese de Mestrado, Unicamp, 1989.

[WAGN-87] Wagner C.F., Medeiros C.B., *Um Modelo Binário Estendido para Entidade-Relacionamento*, Anais do XIV Seminário Integrado de Software e Hardware, Salvador, 1987, pp. 164-174.

[YADA-89] Yadav S.B. et alia., *Comparison of Analysis Techniques for Information Requirements Determination*, CACM, vol. 31, no. 9, 1989, pp. 1090-1097.

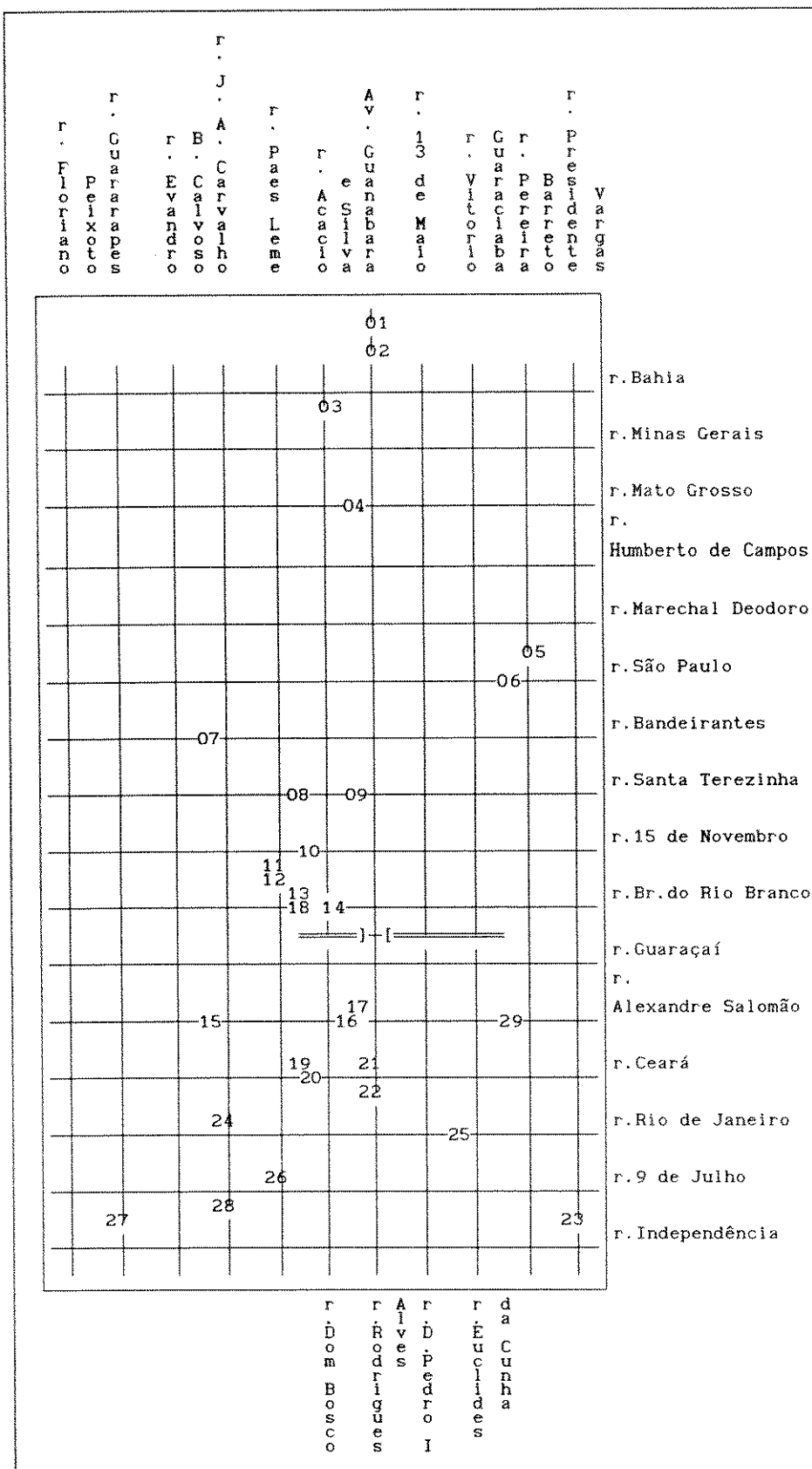
CAPÍTULO 5 - EXEMPLO DE APLICAÇÃO

5.1 INTRODUÇÃO

Será apresentado aqui um exemplo de aplicação do sistema ASTUTO em um problema de planejamento multi-tarefas, semelhante àquele proposto em [HAYE-85].

5.2 O PROBLEMA

O problema consiste em determinar a rota a ser percorrida em uma determinada cidade, respeitando restrições e observando prioridades. O mapa da área de interesse é apresentado na Figura 5.1.



ESTABELECIMENTOS:

01-Restaurante Pé de Cedro	02-Soberana Mecanização Agrícola
03-Tagros Pastoril Agrícola	04-Padaria Cinderela
05-Vídeo Center	06-Supermercado Estrela
07-Supermercado Mini-Box	08-Banco do Brasil
09-Restaurante Sabor e Arte	10-Restaurante Pinguim
11-Lotérica Apolo	12-Banespa
13-Barão da Sorte Loterias	14-Livraria Copacabana
15-Farmácia Pedrogás	16-Floricultura Andraflora
17-Farmácia Moretti	18-Sul América Seguros
19-Peixaria Bandeirantes	20-Eletrônica Gino
21-Correios	22-Livraria Colacino
23-Açougue Dois Irmãos	24-Livraria Regional
25-Auto-Elétrica Estrela	26-Casa do Lavrador
27-Açougue Bom Bife	28-Distribuidora de Bebidas
29-Estação Rodoviária	

Figura 5.1b - Lista dos Principais Estabelecimentos da Cidade

Você chega à estação rodoviária às oito horas e deseja visitar muitos estabelecimentos do comércio local até às catorze horas, horário em que deve pegar seu carro na auto-elétrica e retornar à sua casa, no campo. É muito importante ir ao banco pagar as suas contas, à farmácia, ao supermercado, ao açougue e à padaria abastecer-se. Você gostaria de ir a uma loja de suplementos agrícolas para comprar ração para seus animais e à corretora de seguros. Gostaria também de ir ao correio, à livraria e à locadora de vídeo. Se houver tempo, ir à distribuidora de bebidas verificar se chegou a sua cerveja predileta, à peixaria, à eletrônica, à floricultura e a uma casa lotérica. Além disso, entre 11:30 e 13:30 horas, você gostaria de almoçar em algum de seus restaurantes favoritos. Você associou um intervalo de tempo de permanência a cada tipo de estabelecimento. Desta forma, você espera ter que permanecer vinte minutos na corretora de seguros caso vá renovar o seguro de sua plantação. Você também faz idéia do tempo de percurso entre os estabelecimentos, preferindo então que seu trajeto possa ser cumprido por setores da cidade, a fim de minimizá-lo.

O sistema deve desenvolver um plano que:

- a) especifique quais estabelecimentos visitar, a ordem de percurso, e as rotas entre estabelecimentos sucessivos;
- b) contemple o maior número de estabelecimentos, priorizando os mais importantes e desejáveis, respeite as restrições de tempo, e proporcione rotas eficientes.

5.3 AS FONTES DE CONHECIMENTO

A formação incremental da(s) resposta(s) ao problema é realizada através da ação de primitivas, na forma descrita na Seção 3.4.

A seguir, são mostrados exemplos de primitivas implementadas em ASTUTO. Os exemplos mostram a lógica das fontes de conhecimento. Os detalhes de implementação podem ser obtidos em [NAKA-91b]. Os exemplos podem ser compreendidos sem que sejam necessários maiores conhecimentos da linguagem PROLOG. Àqueles interessados em aprofundar-se na linguagem, sugere-se consultar um texto como [CLOC-84].

Exemplos de primitivas implementadas:

- a) primitivas que geram percursos alternativos.

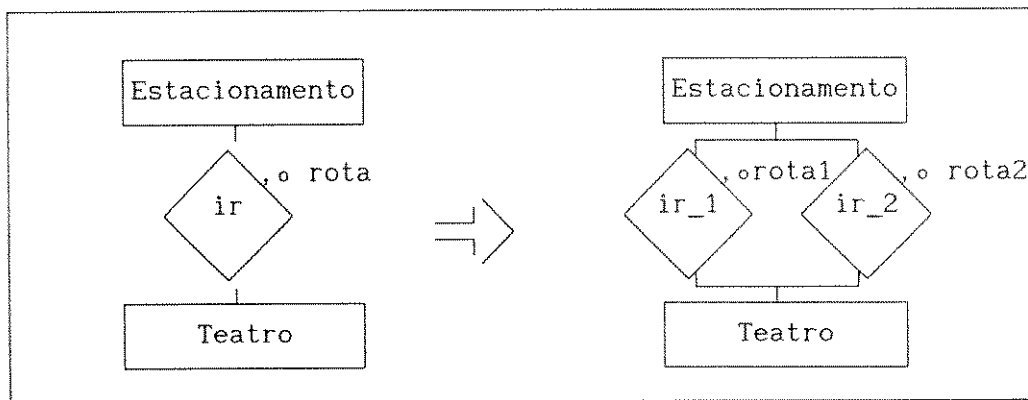


Figura 5.2 - Exemplo de Percurso Alternativo

Existem muitos caminhos diferentes para ir de um ponto a outro numa cidade. Mudando o trajeto, pode-se conseguir visitar outros estabelecimentos próximos ao novo trajeto. Dependendo da estratégia de resolução, pode ser desejável manter ambos os percursos.

Exemplo:

```
kst_sol_alternativa(Estab1,Estab2,Rota1) :-
    eh_caminho(Estab1,Estab2,Rota2,Tempo2),
    Rota2 \= Rota1,
    ins_blackboard(ir(Estab1,Estab2,Hor,Tempo2,Perm,Rota2)),
    ins_l_eventos(sol_alt,[ir,Estab1,Estab2])).
```

Nesse exemplo, ilustrado pela Figura 5.2, *ir(Estab1,Estab2,Hor,Tempo2,Perm,Rota1)* é um fato já presente no blackboard, e representa o percurso entre *Estab1* e *Estab2*; *eh_caminho* é um predicado que retorna um caminho entre dois estabelecimentos, além do tempo de percurso através dele. Sendo diferentes as rotas, insere-se o percurso alternativo no blackboard através do predicado *ins_blackboard* e isso é registrado na lista de eventos através do predicado *ins_l_eventos*.

b) primitivas que classificam os estabelecimentos.

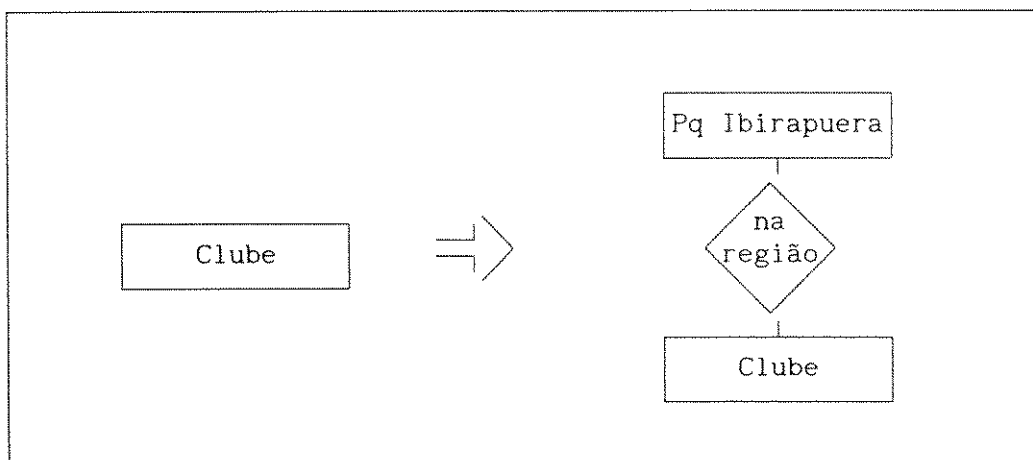


Figura 5.3 - Exemplo de Classificação de Estabelecimento

Classificar os estabelecimentos de acordo com sua localização pode, por exemplo, facilitar o trajeto na região. Isso viabiliza o planejamento das rotas por setores da cidade.

Exemplo:

```
kst_classificação(região, Estabelecimento) :-  
    na_região(Estabelecimento, Região),  
    válida(Região),  
    ins_blackboard(na_região, [Estabelecimento, Região]),  
    ins_l_eventos(ger_classif, [na_região, Estabelecimento, Região]).
```

Nesse exemplo, ilustrado pela Figura 5.3, *na_região* é um predicado utilizado para determinar a região à qual o estabelecimento pertence; *válida* é um predicado que determina se a região é de interesse e se já não foi classificado. Os predicados *ins_blackboard* e *ins_l_eventos* introduzem o relacionamento obtido no blackboard e na lista de eventos.

c) primitivas que dividem relacionamentos.

Alguns estabelecimentos devem ser visitados em sequência, por sua proximidade física, por exemplo. Além disso, se existir um relacionamento *ir* com *horário=meio_dia*, pode ser desejável ir a um restaurante antes de continuar o percurso.

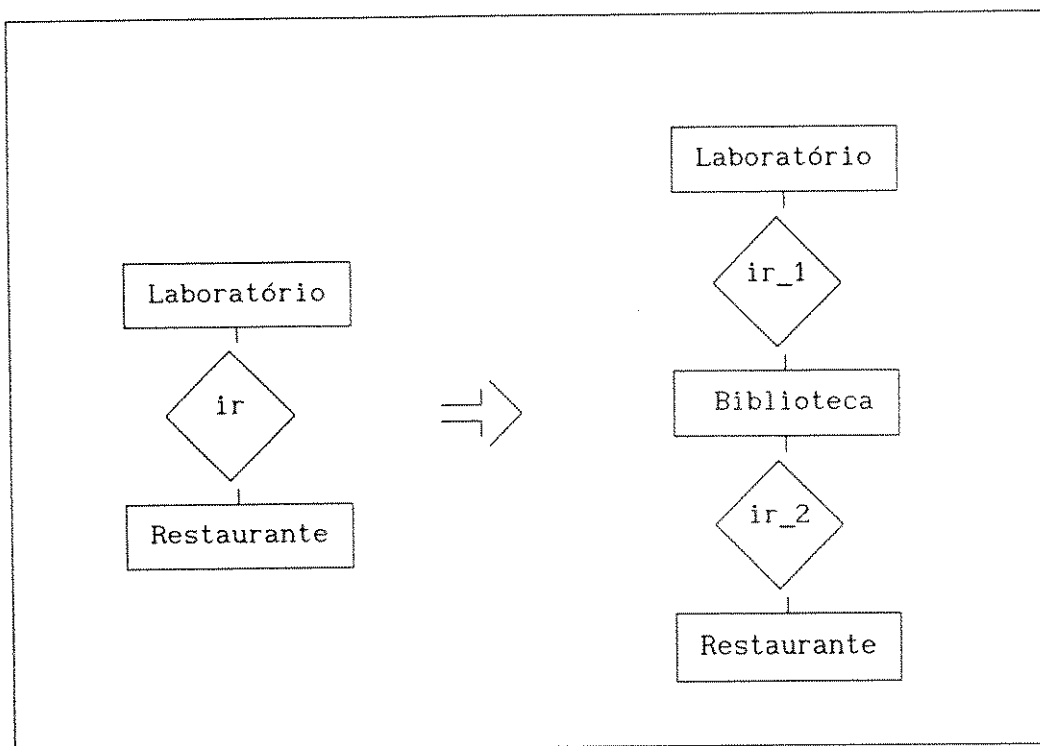


Figura 5.4 - Exemplo de Divisão de Relacionamento

Exemplo:

```

kst_div_relacionamento(Estab, Est1, Est2) :-
    ir(Est1, Est2, Hor, Tempo, Dur, Caminho),
    estabelecimento(Estab, _, _, _, Duracao),
    eh_caminho(Est1, Estab, Caminho1, Tempo1),
    eh_caminho(Estab, Est2, Caminho2, Tempo2),
    retira_bb(ir(Est1, Est2, Hor, Tempo, Dur, Caminho)),
    ins_blackboard(ir(Est1, Estab, Hor, Tempo1, Duracao, Caminho1)),
    soma_tempos([Hor, Tempo1, Duracao], Novo_Hor),
    ins_blackboard(ir(Estab, Est2, Novo_Hor, Tempo2, Dur, Caminho2)),
    ins_l_eventos(div_relacion, [ir, Est1, Estab, Est2]).

```

Fatos presentes no blackboard irão unificar-se com `ir(Est1, Est2, Hor, Tempo, Dur, Caminho)`, e `estabelecimento(Estab, _, _, _, Duracao)`, obtendo informações sobre o relacionamento existente e o estabelecimento que vai ser inserido. O predicado `eh_caminho`

determina caminhos entre os estabelecimentos, e eles são inseridos no blackboard, após estabelecidos seus atributos corretos (o predicado *retira_bb* remove do blackboard o relacionamento anterior). Este procedimento é registrado na lista de eventos. O processo de divisão de relacionamento é ilustrado na Figura 5.4.

d) primitivas que geram estabelecimentos.

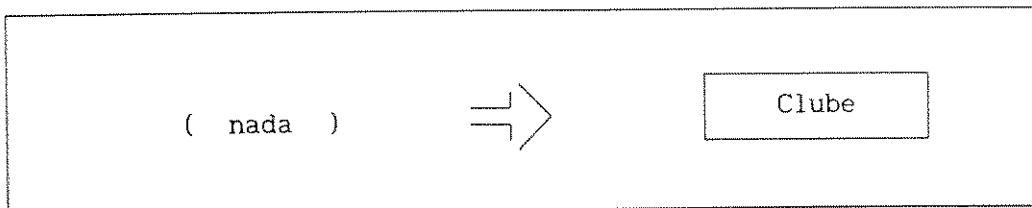


Figura 5.5 - Exemplo de Geração de Estabelecimento

Às vezes pode ser desejável inserir um novo estabelecimento no âmbito do sistema (por exemplo: caso ao fim do percurso ainda haja tempo, pode-se inserir um novo estabelecimento onde praticar esportes).

Exemplo:

```
ksb_ger_estabelecimento(Nome,Tipo,End,Hor,Perm) :-
    not(estabelecimento(Nome,_,_,_,_)),
    ins_blackboard(estabelecimento,[Nome,Tipo,End,Hor,Perm]),
    ins_l_eventos(ger_estabelecimento,[Nome]).
```

A geração de estabelecimentos (Figura 5.5) dá-se com sua inserção no blackboard. Inicialmente, é verificado se o estabelecimento já não fazia parte dele. A geração do novo estabelecimento é registrada na lista de eventos. No protótipo implementado, não existe um módulo que configure os tipos de entidades existentes em cada aplicação. As entidades são diretamente inseridas no blackboard, e as restrições existentes para a sua manipulação, bem como as restrições de memória são dadas pelas limitações impostas pelo equipamento e pelo compilador

utilizado.

e) primitivas que geram relacionamentos.

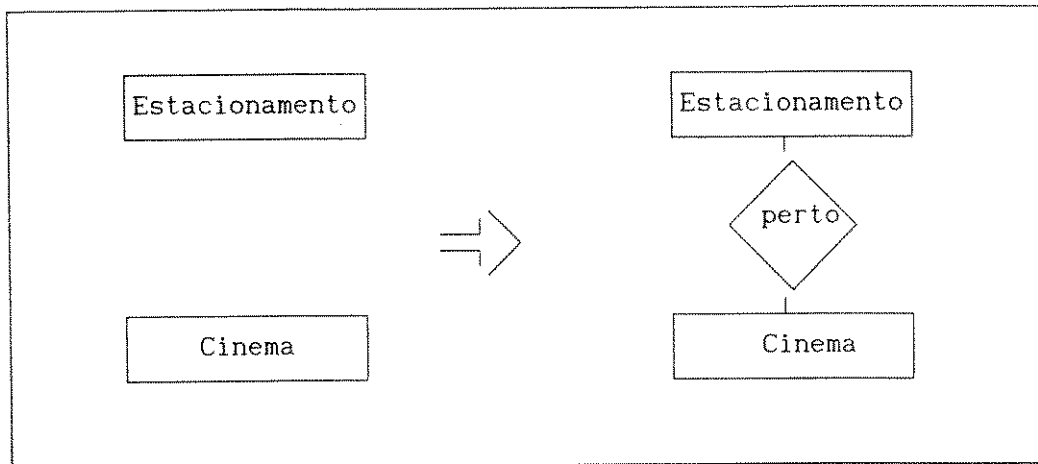


Figura 5.6 - Exemplo de Geração de Relacionamento

A geração de relacionamentos como o da Figura 5.6 constitui-se na ação mais importante e eficaz para a solução do problema, onde o MER é usado efetivamente para a construção da solução.

Exemplo:

```
ksb_ger_relacionamento(ir,Estab1,Estab2) :-  
    eh_caminho(Estab1,Estab2,Caminho,Tempo),  
    melhor_horario_vago(Horario,Tempo),  
    ins_blackboard(ir,[Estab1,Estab2,Horario,Tempo,Caminho]),  
    ins_l_eventos(ger_relacionamento,[ir,Estab1,Estab2]).
```

Após o estabelecimento de um caminho unindo *Estab1* e *Estab2*, *melhor_horario_vago* determina um horário viável para que este percurso seja realizado. O percurso é inserido no blackboard por *ins_blackboard*, e isso é registrado na lista de eventos por *ins_l_eventos*.

f) primitivas que geram hierarquias.

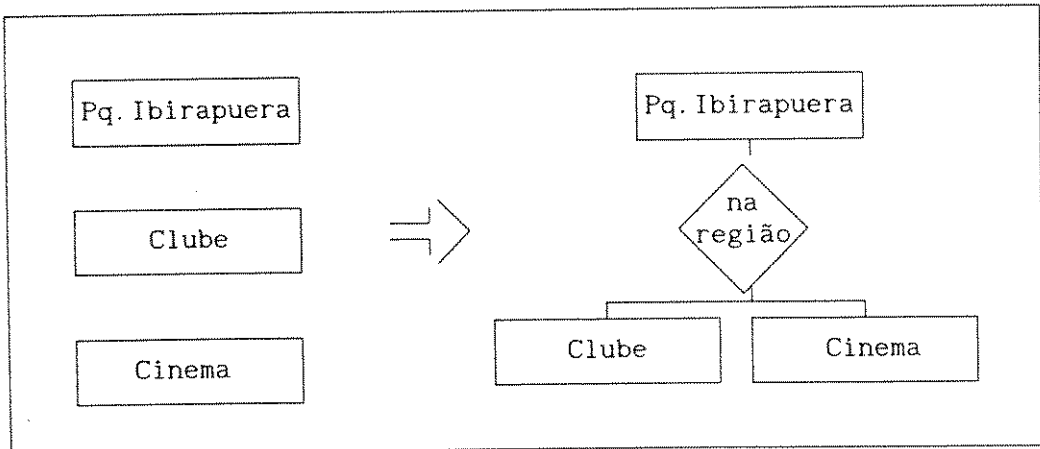


Figura 5.7 - Exemplo de Geração de Hierarquia

O uso de hierarquias (Figura 5.7) viabiliza a utilização de estratégias de resolução de diferentes níveis. Este tipo de fonte de conhecimento pode ser empregado de forma similar às fontes de classificação, agrupando e privilegiando relacionamentos entre entidades afins.

Exemplo:

```
ksb_ger_hierarquia(_,_,[]).
```

```
ksb_ger_hierarquia(na_região,Região,[Estab|Resto]) :-
    ksb_ger_hierarquia(na_região,Região,Resto),
    não_classificado(na_região,Estab),
    ins_blackboard(hierarquia,[Estab,Região]),
    ins_l_eventos(ger_hierarquia(na_região,[Estab,região])).
```

Os estabelecimentos presentes em uma lista pertencem a uma determinada região. Cada estabelecimento ainda não classificado por região (verificado pelo predicado *nao_classificado*) passa a fazer parte de uma hierarquia *na_região*. Esse evento é registrado na lista de eventos.

5.4 CONHECIMENTO SOBRE O DOMÍNIO

A inserção de conhecimento sobre o domínio da aplicação visa dar subsídios ao sistema para que possa efetivamente resolver o problema proposto. Existe conhecimento que contribui também para restringir o universo de soluções válidas, de caminhos desejáveis, e para aumentar a eficiência do mesmo.

O conhecimento, que pode ser implementado na forma de heurísticas, pode ser utilizado na resolução de outros problemas do domínio, facilitando o trabalho em projetos subsequentes. Para o problema das rotas, foram implementadas algumas heurísticas que são utilizadas diretamente nas fontes de conhecimento. As heurísticas implementadas referem-se à otimização de percurso entre estabelecimentos sucessivos. Seu objetivo é exemplificar como podem ser empregadas.

a) Expectativa de tempo de percurso a partir de um dado estabelecimento.

É possível estimar o tempo necessário para o deslocamento a partir de (ou para) um determinado ponto da cidade. Isso pode ser realizado conhecendo-se a posição relativa da região a que pertence. Caso sejam conhecidas tanto a região do estabelecimento origem quanto a do destino, é possível fornecer estimativas mais precisas. Esse tipo de estimativa pode ser muito útil, por exemplo, para restringir o universo de opções quando dispusermos de pouco tempo entre a saída de um estabelecimento e a ida a um outro.

Exemplo:

```
otimiza_percurso(Procedência, Destino, Destino_2, Tipo) :-  
    expectativa(Procedência, Destino, Expectativa),  
    estabelecimento(Destino_2, Tipo, _, _, _),  
    Destino_2 \= Destino,  
    expectativa(Procedência, Destino_2, Expectativa_2),  
    Expectativa_2 < Expectativa,
```

otimiza_percurso(Procedência, Destino_2, Melhor_destino, Tipo).

otimiza_percurso(_, D, D, _).

expectativa(Estabelecimento1, Estabelecimento2, Expectativa) :-

na_região(Establ1, Região1),

na_região(Estab2, Região2),

expectativa_região(Região1, Região2, Expectativa).

No exemplo, *otimiza_percurso* é o predicado que irá indicar um estabelecimento mais próximo (ou de uma região mais próxima). Através do predicado *expectativa*, obtém-se uma expectativa de tempo de percurso entre estabelecimentos, de acordo com as regiões da cidade em que se encontrem. Caso haja algum estabelecimento do tipo desejado em região próxima ao estabelecimento de origem, este será escolhido como destino.

b) Otimização de caminhos.

O estabelecimento de caminhos pode ser otimizado inserindo-se informações sobre a geografia da cidade, e conhecimento sobre como se dirigir de um ponto a outro. Na Figura 5.8, os caminhos "naturais" entre os estabelecimentos indicados por (1) e (2) são aqueles contidos no quadrilátero imaginário pontilhado. É o caso, por exemplo, do caminho (a). Os caminhos externos ao quadrilátero, como o caminho (b), não são considerados para estabelecimentos sucessivos.

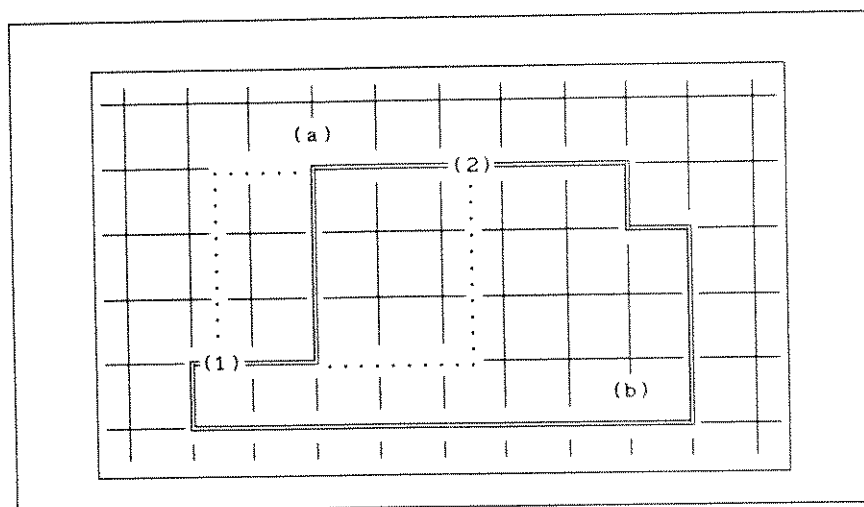


Figura 5.8 - Exemplos de Caminhos

c) Inserção de estabelecimentos mais prioritários

Quando mais de um tipo de estabelecimento pode ser inserido em um determinado ponto do trajeto, insere-se o mais prioritário. Isto aumenta a qualidade da resposta obtida e obedece a proposição do problema.

5.5 GERAÇÃO DE APLICAÇÕES

ASTUTO pode ser utilizado em várias aplicações, pela generalidade do paradigma a que pertence.

O núcleo básico de ASTUTO, utilizado independentemente da aplicação, contém os procedimentos básicos para inicializar e configurar um sistema Blackboard, além das rotinas básicas de controle.

O módulo de controle prioriza o tratamento de eventos relevantes, como a geração de novos relacionamentos, e de expectativas com alto grau de confiabilidade.

Para reconfigurar o sistema, é necessário implementar as fontes de conhecimento voltadas à aplicação.

Abaixo, é mostrado um exemplo de fonte de conhecimento para a geração de relacionamentos. Essa fonte de conhecimento pertence à implementação realizada para o problema das rotas, e está contida em [NAKA-91b], onde encontram-se também alguns detalhes técnicos, como a descrição do ambiente de execução e os nomes utilizados para os arquivos.

```
ksb_geração_relacionamento(ir,Estab1,Estab2) :-  
    estabelecimento(Estab2,Tipo,Endereco,Horario,Perm),  
    insere_ent_bb(Estab1,estab2,Tipo,Endereco,Horario,Perm),  
    assert(lista_eventos(ger_rel,  
        ('Geração de relacionamento -> ir : de ',Estab1,  
        ' para ',Estab2),[Estab1,Estab2])).
```

O predicado *insere_ent_bb* insere a relação de *ir* ao estabelecimento *Estab2*, a partir do estabelecimento *Estab1*. Isso é registrado na lista de eventos.

A implementação de heurísticas pode aumentar muito o desempenho do sistema e a qualidade da solução obtida. As heurísticas podem ser ativadas diretamente das fontes de conhecimento, caso a aplicação sugira que devam ser executadas em sequência.

Para ilustrar a execução de ASTUTO, é apresentada abaixo a listagem dos resultados obtidos pelo sistema para o problema das rotas:

I	I
I	I
I	I
I	I
I	I
I	I
I	I
I	I
I	I
I	I
I	I
I	I
I	I

TEMPO INICIAL : 10:00 hs - Chegada na Estacao Rodoviaria

TEMPO FINAL : aprox. 14:30 hs - Retorno a Estacao Rodoviaria

A partir da Estação Rodoviaria :

--> Pelo caminho:

[(Indo pela rua , rua_alexandre_salomao , ate a esquina com a rua , rua_presidente_vargas , , seguindo-a ate , dois_irmaos)] chega-se ao destino as 10.06hs.

--> Saindo de dois_irmaos:

[(Pela rua , rua_presidente_vargas , , tomando , rua_ceara , ate , rua_rodrigues_alves , , seguindo-a ate , correio)] as 10.23hs.

--> Então, as 10.33hs:

[(Pela rua , rua_rodrigues_alves , seguindo por , rua_alexandre_salomao , , tomando , rua_guararapes , e chegando em , sul_america , por , av_br_rio_branco)]

--> Pelo caminho:

[(Indo pela rua , av_br_rio_branco , ate a esquina com a rua , rua_j_a_carvalho , , seguindo-a ate , kaiser_e_cia)] chega-se ao destino as 11.11hs.

--> Saindo de kaiser_e_cia:

[(Indo pela rua , rua_j_a_carvalho , ate a esquina com a rua , rua_15_de_novembro , , seguindo-a ate , pinguim)] as 11.3hs.

--> Então, as 12.1hs:

[(Pela rua , rua_15_de_novembro , , tomando , rua_paes_leme , ate , rua_santa_terezinha , , seguindo-a ate , banco_do_brasil)]

--> Pelo caminho:

[(Pela rua , rua_santa_terezinha , , tomando , rua_acacio_silva , ate , rua_ceara , , seguindo-a ate , eletronica_gino)] chega-se ao destino as 12.37hs.

--> Saindo de eletronica_gino:

[(Pela rua , rua_ceara , , tomando , rua_presidente_vargas , ate , rua_alexandre_salomao , , seguindo-a ate , moretti)] as 13.03hs.

--> Entao, as 13.2hs:

[(Pela rua , rua_alexandre_salomao , , tomando , rua_rodrigues_alves , ate , rua_rio_de_janeiro , , seguindo-a ate , auto_estrela)]

--> Pelo caminho:

[(Pela rua , rua_rio_de_janeiro , , tomando , rua_rodrigues_alves , ate , rua_alexandre_salomao , , seguindo-a ate , andraflora)] chega-se ao destino as 13.48hs.

Classificacao por Regiao: , pe_de_cedro , em ,
norte
Classificacao por Regiao: , estacao_rodoviaria_ch
egada , em , leste
Classificacao por Regiao: , pinguim , em , matr
iz
Classificacao por Regiao: , sabor_e_arte , em ,
leste

OTIMIZACAO DE PERCURSO

Otimizacao de percurso : , pinguim , em lugar d
e , pe_de_cedro

GERACAO DE CLASSIFICACAO

Classificacao por Regiao: , apolo , em , matriz

TRATAMENTO DE EXPECTATIVA

EVENTO TRATADO : Expectativa , almocar_restaurante

GERACAO DE CLASSIFICACAO

Classificacao por Regiao: , auto_estrela , em ,
centro

TRATAMENTO DE EXPECTATIVA

EVENTO TRATADO : Expectativa , buscar_carro_oficina

GERACAO DE CLASSIFICACAO

Classificacao por Regiao: , estacao_rodoviaria ,
em , leste

TRATAMENTO DE EXPECTATIVA

EVENTO TRATADO : Expectativa , ir_a_rodoviaria

GERACAO DE CLASSIFICACAO

Classificacao por Regiao: , bom_bife , em , sul
Classificacao por Regiao: , dois_irmaos , em ,
sul

OTIMIZACAO DE PERCURSO

Otimizacao de percurso : , dois_irmaos , em lug
ar de , bom_bife

GERACAO DE ENTIDADE

EVENTO TRATADO : Geracao da entidade: , andraflora

GERACAO DE CLASSIFICACAO

, matriz Classificacao por Regiao: , banco_do_brasil , em
 iz Classificacao por Regiao: , banespa , em , matr
 GERACAO DE ENTIDADE
 EVENTO TRATADO : Geracao da entidade: , apolo
 GERACAO DE CLASSIFICACAO
 Classificacao por Regiao: , correio , em , cent
 ro
 GERACAO DE ENTIDADE
 EVENTO TRATADO : Geracao da entidade: , auto_estrela
 GERACAO DE CLASSIFICACAO
 Classificacao por Regiao: , sul_america , em ,
 matriz
 GERACAO DE ENTIDADE
 EVENTO TRATADO : Geracao da entidade: , banco_do_brasil
 GERACAO DE CLASSIFICACAO
 Classificacao por Regiao: , kaiser_e_cia , em ,
 sul
 DIVISAO DE RELACIONAMENTO
 Divisao relacionamento com superposicao de entida
 de -> ir : , sul_america , para , kaiser_e_cia , para , pingui
 m
 GERACAO DE ENTIDADE
 EVENTO TRATADO : Geracao da entidade: , bandeirantes
 GERACAO DE CLASSIFICACAO
 Classificacao por Regiao: , eletronica_gino , em
 , centro
 GERACAO DE ENTIDADE
 EVENTO TRATADO : Geracao da entidade: , banespa
 GERACAO DE CLASSIFICACAO
 Classificacao por Regiao: , moretti , em , cent
 ro
 Classificacao por Regiao: , pedrogas , em , oes
 te

DIVISAO DE RELACIONAMENTO

Divisao relacionamento com superposicao de entida
de -> ir : , eletronica_gino , para , moretti , para , auto_es
trela

GERACAO DE ENTIDADE

EVENTO TRATADO : Geracao da entidade: , barao_da_sorte

GERACAO DE CLASSIFICACAO

Classificacao por Regiao: , andraflora , em , c
entro

GERACAO DE ENTIDADE

EVENTO TRATADO : Geracao da entidade: , bom_bife

GERACAO DE CLASSIFICACAO

Classificacao por Regiao: , colacino , em , cen
tro

Classificacao por Regiao: , copacabana , em , m
atriz

Classificacao por Regiao: , regional , em , cen
tro

DIVISAO DE RELACIONAMENTO

Divisao relacionamento com superposicao de entida
de -> ir : , andraflora , para , colacino , para , estacao_rod
oviaria

GERACAO DE ENTIDADE

EVENTO TRATADO : Geracao da entidade: , casa_do_lavrador

REFERÊNCIAS

[CLOC-84] Clocksin W.F., Mellish C.S., *Programming in Prolog*, Springer-Verlag, 1984.

[HAYE-85] Hayes-Roth B., *A Blackboard Architecture for Control*, Artificial Intelligence, vol.26, no.3, July 1985, pp. 251-321.

[NAKA-91b] Nakamiti G.S., Daltrini, B.M., *Especificação e Implementação do Sistema ASTUTO*, Relatório Técnico RT-DCA 018/91, Departamento de Engenharia de Computação e Automação Industrial, Faculdade de Engenharia Elétrica, Unicamp, Junho de 1991.

CAPÍTULO 6 - CONCLUSÃO

O modelo Blackboard pode ser visto como um modelo genérico de resolução de problemas, encontrando uso em inúmeras áreas de aplicação.

ASTUTO é um sistema que segue a filosofia Blackboard e que propicia o uso desse paradigma para a resolução de problemas em diferentes domínios de aplicação. A abordagem utilizada privilegia a visão que as fontes de conhecimento têm do blackboard e o comportamento do sistema dirigido por eventos, aproximando-se daquilo que entendemos por essência do paradigma.

Essa abordagem dirigida por eventos permite que as fontes de conhecimento sejam alertadas sobre fatos importantes que possam alterar seus caminhos de raciocínio. O tratamento dos eventos dá-se de acordo com sua importância para o processo de resolução. Certos tipos de eventos, que causam alterações significativas no estado da solução do problema, podem causar imediata ativação de fontes de conhecimento para tratá-los. Outros tipos de eventos são armazenados em listas e tratados de acordo com as estratégias vigentes para o processo de resolução. Seu tratamento implica a ativação de fontes de conhecimento que irão também alterar o blackboard.

A modelagem de ASTUTO é bastante abrangente, cobrindo seu esquema conceitual, a definição de seus módulos em SADT, a descrição dos fluxos de dados e de controle e seu inter-relacionamento, e sua descrição em PDL. Foi também implementado um protótipo para o sistema, utilizado na resolução de um problema de planejamento multi-tarefas. Esse protótipo, implementado em ARITY PROLOG, oferece funções de inicialização do núcleo e operação de sistemas Blackboard, como por exemplo inicialização de estruturas de dados: blackboard, lista de

eventos, lista de expectativas e lista de fontes de conhecimento prontas para serem executadas. Oferece também funções de controle, que permitem estabelecer dinamicamente as estratégias de resolução de problemas, suportando inclusive rotinas e heurísticas externas, particulares para cada aplicação. Possui ainda rotinas que permitem a reconfiguração do sistema como um todo, como aquelas que inicializam as fontes de conhecimento particulares de cada aplicação. No protótipo produzido já existem fontes de conhecimento para a manipulação do blackboard, implementando primitivas que operam sobre o modelo de entidades e relacionamentos: primitivas para a geração de entidades, geração de relacionamentos, geração de hierarquias, classificação de entidades, geração de soluções alternativas, e divisão de relacionamentos.

A contribuição desta tese caracteriza-se por vários itens nela tratados, onde se destacam:

a) *Aporte do conhecimento sobre sistemas Blackboard para o contexto local.*

Aspectos conceituais, funcionamento, aplicações e diferentes abordagens para o paradigma Blackboard foram apresentados de forma bastante clara e abrangente, permitindo ao leitor uma visão clara, completa e atualizada sobre o tema.

b) *Modelo de funcionamento.*

Obteve-se um modelo genérico de funcionamento para sistemas Blackboard realmente muito interessante (Figura 4.5). Esse modelo privilegia a visão que as fontes de conhecimento têm do blackboard, e onde sua atuação gera eventos de várias naturezas. Isso, além de diminuir a interferência do módulo de controle sobre o processo de resolução como mero escalonador de processos, permite uma ação mais "oportunistica" das várias fontes de conhecimento.

c) *Utilização do Modelo Entidade-Relacionamento na modelagem do blackboard e do conjunto de ferramentas para o projeto ASTUTO.*

A utilização do Modelo Entidade-Relacionamento [CHEN-76] para a modelagem e implementação do blackboard, substituindo o modelo hierárquico presente nos sistemas até então conhecidos é uma contribuição relevante desta tese. O Modelo Entidade-Relacionamento pode contribuir de forma decisiva para um aumento da flexibilidade da representação interna dos sistemas Blackboard, agregando as inúmeras facilidades de modelagem propostas por diversos pesquisadores desde a sua concepção inicial (como em [SMIT-77] e [WAGN-87]).

d) *A implementação de ASTUTO.*

Outra contribuição da tese é a implementação de um protótipo para o sistema ASTUTO, ilustrando seu funcionamento e sua reconfiguração para ser utilizado em diversos domínios de aplicação. Além disso, a aplicação escolhida foi de encontro aos seus propósitos. O sistema de planejamento multi-tarefas é bastante simples e de fácil entendimento. É, ao mesmo tempo, um sistema que cobre toda uma classe de aplicações, permitindo vislumbrar seu enorme potencial de uso. Assim como para as funções implementadas, a utilização das heurísticas apresentadas possui um escopo muito amplo. As heurísticas de otimização de percurso, por exemplo, podem ser utilizadas para calcular custos mínimos ou soluções numéricas ou conceitualmente "melhores", bastando associar medidas adequadas a cada um dos trechos do percurso.

Além de propor um sistema que traz alterações importantes em relação aos sistemas Blackboard tradicionais, aproximando-se da metáfora Blackboard, propôs-se um conjunto coeso de ferramentas para sua efetiva implementação. Destacam-se o uso do SADT - Structured Analysis and Design Technique [SOFT-76], que representa de forma clara e precisa o modelo de funcionamento através de eventos. A utilização dessa metodologia, através de seus actigramas, vai de encontro ao modelo de primitivas que atuam sobre entidades e relacionamentos proposto em ASTUTO. Além disso,

as estruturas do Modelo Entidade-Relacionamento são implementadas simples e diretamente através das cláusulas PROLOG, a linguagem de programação escolhida. Isso simplifica e evita inconsistências na transformação conceitual do modelo de entidades e relacionamentos para a representação interna da linguagem de programação.

Desta forma, os objetivos iniciais deste trabalho foram alcançados com grande êxito, tanto no plano teórico quanto no prático. Espero que esta tese possa motivar e dar subsídios para novos trabalhos nesta área tão fecunda e promissora.

Como sequência deste trabalho, seria interessssante criar interfaces para mais fácil interação com o usuário, através das quais, leigos na linguagem de programação pudessem produzir suas próprias aplicações, uma vez que para a codificação de fontes de conhecimento e heurísticas, é necessário conhecer a linguagem de programação utilizada. O estabelecimento de bibliotecas de procedimentos e heurísticas para diferentes domínios de aplicação parece poder facilitar trabalhos futuros pela reutilização dos mesmos, caso sejam produzidos e documentados adequadamente. Além disso, ASTUTO poderia ser implementado como sistema-produto, inclusive em equipamentos mais poderosos, já disponíveis também no ambiente universitário.

Um sistema como ASTUTO pode ser utilizado não só para gerar aplicações finais, mas também para a geração de protótipos, visando o refinamento de requisitos de software. Pode também ser utilizado dentro de contextos mais amplos, como na manutenção de sistemas e como centralizador das operações das diversas ferramentas em um Ambiente de Desenvolvimento de Software, ou ainda em ambientes industriais. Sua utilização em tais contextos pode acarretar aumentos significativos na flexibilidade, modularidade e extensibilidade dos sistemas que o abrigarem.

Com os avanços tecnológicos que vêm sendo obtidos, com a possibilidade de uso de computadores mais poderosos, processamento paralelo e distribuído, novas tecnologias de software que aproximem mais a execução ao raciocínio humano, e a aplicação do

conhecimento de outras ciências (como as ciências cognitivas), a utilização de paradigmas como o paradigma Blackboard devem receber cada vez maior atenção pelas perspectivas que abrem para o estabelecimento de novos modelos para a resolução de problemas, para o planejamento, e para as atividades humanas de maneira geral.

REFERÊNCIAS

[CHEN-76] Chen P.P., *The Entity-Relationship Model: Toward a Unified View of Data*, ACM Transactions on Database Systems, vol.01, no.01, 1976.

[SMIT-77] Smith J.R., Smith D.C., *Database Abstractions: Aggregation and Generalization*, ACM Transactions on Database Systems, vol.02, no.02, 1977.

[SOFT-76] Softec Inc., *An Introduction to SADT*, Manual, 1976.

[WAGN-87] Wagner C.F., Medeiros C.B., *Um Modelo Binário Estendido para Entidade-Relacionamento*, Anais do XIV Seminário Integrado de Software e Hardware, Salvador, 1987, pp.164-174.