UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

# Utilização de uma Ferramenta Independente do Domínio para Diagnóstico do Comportamento do Aluno em Atividades de Ensino a Distância

por

VANESSA LINDEMANN

Dissertação submetida à avaliação como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Profa. Dra. Ana Lúcia Cetertich Bazzan
Orientadora

Porto Alegre, abril de 2002

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Lindemann, Vanessa
Utilização de uma Ferramenta Independente do Domínio para Diagnóstico do Comportamento do Aluno em Atividades de Ensino a Distância/ por Vanessa Lindemann. – Porto Alegre: PPGC da UFRGS, 2001.

102f.:il.
Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR – RS, 2001. Orientadora: Bazzan, Ana Lúcia Cetertich.

1. Informática na Educação. 2. Ensino a Distância. 3. Inteligência Artificial. 3. Diagnóstico cognitivo. 4. *TÆMS*. 5. Adaptabilidade. I. Bazzan, Ana Lúcia Cetertich. II. Título.

Em memória de minha querida mãe.

# Agradecimentos

Ao CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico, pelo auxílio financeiro recebido sob a forma de bolsa, e à CAPES - Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, pelo auxílio financeiro recebido através do PAPED – Programa de Apoio à Pesquisa em Educação a Distância.

Aos professores do Curso de Pós-Graduação em Computação da UFRGS – Universidade Federal do Rio Grande do Sul, que me auxiliaram na construção do conhecimento e na busca do aperfeiçoamento. Em especial, à minha orientadora, Profª. Ana Lúcia Cetertich Bazzan, pela atenção e esclarecimentos. Meu reconhecimento pelo apoio, pelas palavras de estímulo e pela paciência.

Ao bolsista Andrey, pela cooperação na implementação do protótipo; aos bolsistas Gustavo, Tiago e Guilherme e ao colega Luciano, pelos esclarecimentos, presteza e boa vontade que sempre demonstraram; aos colegas de curso, pelas conversas, trocas de experiências, incentivo e ajuda técnica dispensadas; à amiga Alice pela amizade e compreensão; aos amigos, Rodrigo e Rafael, pela ajuda prestada para a realização deste trabalho.

A meu pai Paulo e as minhas irmãs Michele e Cristiane pelo carinho, estímulo e confiança; ao meu namorado Geraldo, pelo amor, atenção e paciência. A todas as pessoas que torceram por mim e acreditaram na minha capacidade.

# Sumário

# Lista de Abreviaturas

BD - Banco de Dados
*DTC* - *Design-to-Criteria*
EAD - Ensino a Distância
ER - Entidade-Relacionamento
*FIS* - *Fault Isolation System*
*GDE* - *General Diagnosis Engine*
*HTML* - *Hypertext Markup Language*
*HTTP* - *Hypertext Transfer Protocol*
IA - Inteligência Artificial
IE - Informática na Educação
*KQML* - *Knowledge Query and Manipulation Language*
*NLE* - *Non-Local Effect*
*PDM* - *Prime Diagnostic Method*
*PSM* - *Problem-Solving Method*
*SHARP* - *Simple Home Agent Resource Protocol*
SMA - Sistema Multiagente
STI - Sistema Tutor Inteligente
*TÆMS* - *Task Analysis, Environment Modeling and Simulation*
UUT - Unidade de teste

# Lista de Figuras

# Lista de Tabelas

# Resumo

A evolução da Informática na Educação exige ambientes de ensino capazes de se adaptarem ao contexto de acordo com as características individuais do aluno, permitindo interatividade, e que gerem um diagnóstico do comportamento desse aluno. Com base nestes argumentos, o objetivo deste trabalho é propor um sistema de diagnóstico independente do domínio, capaz de analisar o comportamento do aluno em cursos de Ensino a Distância. O professor organiza o material em estruturas de tarefas *TÆMS* (uma linguagem independente do domínio para descrição de planos de resolução de tarefas), gerando uma biblioteca de planos que deverão ser executados pelo aluno. As informações referentes à navegação do aluno pelo material são gravadas em um *log*. O processo de diagnóstico ocorre através do confronto entre as informações do *log* e os planos gerados pelo professor (esta comparação é baseada em um modelo causal geral que pode ser utilizado para diagnosticar diferenças entre quaisquer estruturas *TÆMS*). Se forem detectadas divergências no processo de diagnóstico, o sistema gerará um arquivo texto contendo os sintomas detectados e as possíveis causas para que estes tenham ocorrido.

**Palavras-Chaves:** Informática na Educação, Ensino a Distância, Inteligência Artificial, diagnóstico cognitivo, *TÆMS*, adaptabilidade.

**TITLE**: “USE OF DOMAIN-INDEPENDENT TOOL FOR COGNITIVE DIAGNOSIS IN DISTANCE LEARNING”

# Abstract

The evolution of Informatics in Education demands environments capable of adapting to the context according to the student's individual characteristics. This permits interactivity and creates a diagnosis of this student's behavior. Based on these facts, this work proposes a domain-independent diagnosis system, which is capable of analyzing the student's behavior in Distance Learning courses. The teacher creates the material in *TÆMS* tasks structures (a domain-indepedent language for description of plans and tasks), generating a plan library that should be executed by the student. The information regarding the student's navigation through the material is recorded in a log file. The diagnosis process happens when the information of the log is confronted with the task structures generated by the teacher (this comparison is based on general causal model that can be used to diagnose differences among any structures TÆMS). If divergences are detected in the diagnosis process, the system will generate a file text containing the detected symptoms and the possible causes.

**Keywords:** Informatics in Education, Distance Learning, Artificial Intelligence, cognitive diagnosis, *TÆMS*, adaptability.

# 1 Introdução

A Ciência da Computação, juntamente com a Psicologia e a Educação, tem buscado aperfeiçoar ferramentas computacionais, voltadas principalmente para o ensino individualizado. Novas abordagens do uso da Informática na Educação (IE) têm trazido boas perspectivas para esta área.

As primeiras ferramentas surgiram na década de 50, como métodos de ensino auxiliados por computador, e utilizavam técnicas e equipamentos que na época representavam as crenças sobre como se deveria utilizar o computador como recurso pedagógico. Na década de 60, foram utilizadas técnicas para decidir qual o conteúdo a ser apresentado ao aluno com base em sua resposta anterior. Na década de 70, foram criados os chamados sistemas generativos, capazes de gerar automaticamente o material instrucional, comparando a solução do sistema com a do aluno. Esses sistemas, no entanto, não eram capazes de responder a perguntas do aluno a respeito de "como" foi obtida a solução, pois não possuíam uma representação do processo de resolução dos problemas propostos.

Os Sistemas Tutores Inteligentes (STI) surgiram na década de 80, com o objetivo de fazer com que os sistemas deixassem de ser um mero livro eletrônico, onde apesar de o conteúdo programático ser colocado de forma atrativa e atual, não havia uma utilização mais personalizada e adequada ao perfil do aluno que utilizava o sistema, uma vez que o comportamento do ambiente era sempre o mesmo, sessão após sessão.

Na década de 90, ganha força o paradigma de Ensino a Distância (EAD), cujo objetivo é o desenvolvimento de ambientes e de metodologias que propiciem o aprendizado remoto. O EAD tem sua origem na educação por correspondência, cujos cursos utilizam materiais impressos distribuídos pelos correios, ou os meios de comunicação de massa, como a televisão e o rádio. Apesar de não ser uma prática de uso recente, com a utilização da Internet (mídia já popularizada e acessível), esse paradigma apresenta-se como uma inovação, criando novas possibilidades para a democratização do ensino. Novas formas de comunicação e de interação passaram a propiciar a troca de conhecimentos, desconsiderando as distâncias físicas e temporais. Professores e alunos passam a usufruir do ensino a distância via *web*, em que os meios eletrônicos de comunicação intermediam o processo de ensino e de aprendizagem. Segundo Tarouco [TAR 2001], as razões para a utilização do EAD são: falta de tempo para assistir às aulas presenciais; distância (dificuldade de deslocamento); finanças (custo); oportunidade de fazer cursos não oferecidos no local ou período de tempo apropriados; possibilidade de entrar em contato com outros estudantes de diferentes classes sociais, culturais, econômicas e experimentais.

Entretanto, a utilização pura e simples de recursos computacionais costuma resultar na construção de ambientes de aprendizagem estáticos que, ou não condizem com as modernas teorias educacionais, ou apresentam ao aluno uma avalanche de material instrucional, deixando totalmente sob sua responsabilidade a ação a ser tomada no processo de aprendizagem. Para que isso não aconteça, além de recursos computacionais, é indispensável o uso de recursos propiciados pela Inteligência Artificial (IA).

Os benefícios obtidos com a utilização da IA são importantes, pois a IE tem evoluído desde a utilização do computador no ensino, de laboratórios de informática e do desenvolvimento de *software* educacionais a ambientes de ensino na Internet, sistemas

inteligentes de ensino e cursos virtuais. Estes ambientes devem ser capazes de se adaptarem ao contexto de acordo com as capacidades individuais do aluno, permitir interatividade e diagnosticar o comportamento cognitivo desse aluno. Este comportamento consiste num conjunto de informações sobre o modo como o aluno interage com o ambiente de aprendizagem, e o sistema deve monitorar esta interação e diagnosticar possíveis erros cometidos pelo aluno.

Nas pesquisas de IA na educação, o diagnóstico cognitivo é considerado algo extremamente difícil [KON 2000]. Uma das razões é que, geralmente, o diagnóstico do comportamento é baseado em catálogos de erros, que são difíceis de criar e aplicáveis somente a domínios específicos. Quando o assunto muda, é necessário desenvolver e implementar um novo catálogo de erros. Na tentativa de resolver esses problemas, são utilizados métodos de gerar erros dinamicamente, porém, estes também apresentam dificuldades. Self [SEL 93] afirma que o diagnóstico cognitivo, sendo um problema complexo, apresenta um grande desafio: expressar os aspectos cognitivos não cobertos pelo diagnóstico de falhas e caminhar para um *framework* padrão para a área de diagnóstico. Com este trabalho, pretende-se contribuir exatamente nesta direção.

Considerando a evolução da IE e tomando como referências os estudos sobre sistemas gerais de diagnóstico, constatou-se a necessidade de um sistema de diagnóstico cognitivo independente do domínio. Assim, a proposta deste trabalho é projetar e implementar um sistema de diagnóstico baseado em modelo, adaptativo e genérico, capaz de diagnosticar o comportamento do aluno em cursos de EAD.

Este trabalho está dividido em 8 capítulos que descrevem todo o processo de estudo, projeto, implementação e validação do sistema proposto, dispostos numa seqüência lógica para facilitar sua compreensão, conforme segue.

O capítulo 2 é dedicado à exposição de alguns conceitos e idéias fundamentais à construção de sistemas de diagnóstico. Além disso, apresenta uma análise do processo de diagnóstico, baseada num *framework* proposto por Benjamins [BER 95], que reflete a convergência das abordagens atuais.

Uma breve revisão bibliográfica sobre os sistemas de diagnóstico de falhas em dispositivos físicos, que serviu como embasamento para a proposta apresentada neste trabalho, é descrita no capítulo 3. Estão resumidas as abordagens de Davis, Genesereth, Reiter, de Kleer e Pipitone, que abordam a teoria do raciocínio sobre princípios básicos, cada qual tratando de implementar um algoritmo de diagnóstico com características próprias.

O capítulo 4 trata especificamente sobre diagnóstico cognitivo. Introduz alguns conceitos e apresenta o sistema de diagnóstico baseado em modelo, proposto por Koning [KON 2000]. Para diagnosticar o comportamento do aluno, modelos são gerados automaticamente por um simulador qualitativo, e divergências entre o comportamento real do aluno e o comportamento esperado, gerado pelo simulador, são diagnosticadas.

As ferramentas utilizadas para atingir a independência do domínio no sistema, estão descritas no capítulo 5. O *framework TÆMS* e o escalonador *Design-To-Criteria* (*DTC*) são ferramentas utilizadas para criar componentes de controle genéricos para agentes inteligentes e são utilizadas na resolução de problemas complexos.

O projeto do sistema de diagnóstico cognitivo proposto neste trabalho é descrito no capítulo 6. Além da descrição do modelo proposto, as seções contêm explicações sobre a

geração dos planos, a função do *DTC* e do monitoramento, e sobre como ocorre o processo de diagnóstico.

O capítulo 7 descreve o processo de implementação do protótipo do sistema proposto, detalhando seu ambiente de implementação. Também apresenta uma visão geral do sistema e a descrição da sua interface, relata os experimentos realizados com o protótipo e apresenta os resultados obtidos.

As considerações finais são apresentadas no capítulo 8, onde são relatadas as principais contribuições obtidas com a utilização do sistema proposto e perspectivas de trabalhos futuros.

Os anexos apresentam a descrição textual de uma estrutura de tarefas *TÆMS* completa para o domínio da Computação Simbólica e Numérica, as tabelas do banco de dados do sistema que devem ser previamente preenchidas, um manual de instalação do sistema e as publicações relacionadas ao trabalho de dissertação.

# 2 Classificação e análise de métodos de diagnóstico

Para que um sistema utilizado com fins educacionais seja efetivo e contextualizado no trabalho de professores e alunos, não basta que sejam privilegiados somente aspectos técnicos, mas, principalmente, sua adequação pedagógica ao contexto em que se insere, considerando as características individuais do aluno e a instrução individualizada, de forma a facilitar a criação de estruturas conceituais e metodológicas adequadas à capacidade e interesse deste aluno.

Do ponto de vista psicopedagógico, os sistemas de ensino por computador têm sido classificados em três grupos. Para os *behavioristas*, normalmente o controle do processo está nas mãos do autor do sistema e é baseado exclusivamente no comportamento e desempenho que o aluno apresenta no curso de seqüências pré-definidas. Já nos chamados *ambientes cognitivos*, o controle do processo é misto, sendo baseado na diagnose de ações do aluno em interação através dos conceitos e medidas possíveis oriundas da Psicologia Cognitiva. Nos chamados *ambientes abertos*, o controle da interação pode estar com o aluno, ou seja, a flexibilidade do sistema depende do modelo do aluno. Quanto mais fiel for a imagem que o sistema possui do estado do conhecimento e da forma de raciocinar de seu aluno, mais flexível e proveitosa será a interação. Sendo assim, o modelo do aluno deve ser capaz de representar todas as qualidades possíveis para uma avaliação pedagógica, tanto quantitativa (desempenho) como qualitativa (fatores e características afetivas do aluno).

O modelo do aluno depende de uma avaliação pedagógica, baseada em *quanto* um aluno conhece sobre um determinado tópico ou domina uma habilidade específica, o que acarreta estudar e detectar o conhecimento do aluno em relação ao domínio. Entretanto, ao considerar fatores motivacionais e afetivos, a avaliação pedagógica se complementa, pois adiciona-se *o que* o aluno conhece sobre o domínio e *qual sua intensidade em desenvolver* (predisposição) mais seu *conhecimento*. Neste caso, o sistema educacional é apoiado por um processo de diagnose, através de modelagem.

Este capítulo é dedicado à classificação e análise dos métodos de diagnóstico. A seção 2.1 descreve como os métodos de diagnóstico podem ser estruturados, como diagnóstico baseado na especificação, diagnóstico baseado em sintomas e diagnóstico cognitivo. A seção 2.2 apresenta uma análise do processo de diagnóstico, baseado num *framework* proposto por Benjamins [BER 95].

## 2.1 Estrutura dos métodos de diagnóstico

Geralmente, um diagnóstico parte da observação de algum comportamento que é reconhecido como um desvio sobre uma situação desejada ou prevista, ou seja, observa-se um comportamento de mau funcionamento. Para resolver esse problema, são geradas algumas hipóteses sobre a causa do mau funcionamento. Esta tarefa de geração de hipóteses pode ser mais ou menos complexa e mais ou menos controlada, dependendo do domínio e do conhecimento que se tem. Não importa o método, todos partem de uma observação comportamental (valores de testes, sinais, sintomas, etc.) para chegar a um número de hipóteses, preferencialmente ordenadas [BAZ 92]. Segundo Benjamins [BER 95], o diagnóstico consiste em três subtarefas (FIGURA 2.1):

(a) detectar sintomas: detecta se os desvios de comportamento são realmente sintomas, em que sintoma (ou observação de anormalidade) é definido como uma observação que diverge da observação esperada;

(b) gerar hipóteses: baseada nos sintomas, gera as possíveis causas para o problema (também considera as observações de normalidade);

(c) discriminar hipóteses: discrimina as hipóteses geradas com base em observações adicionais.

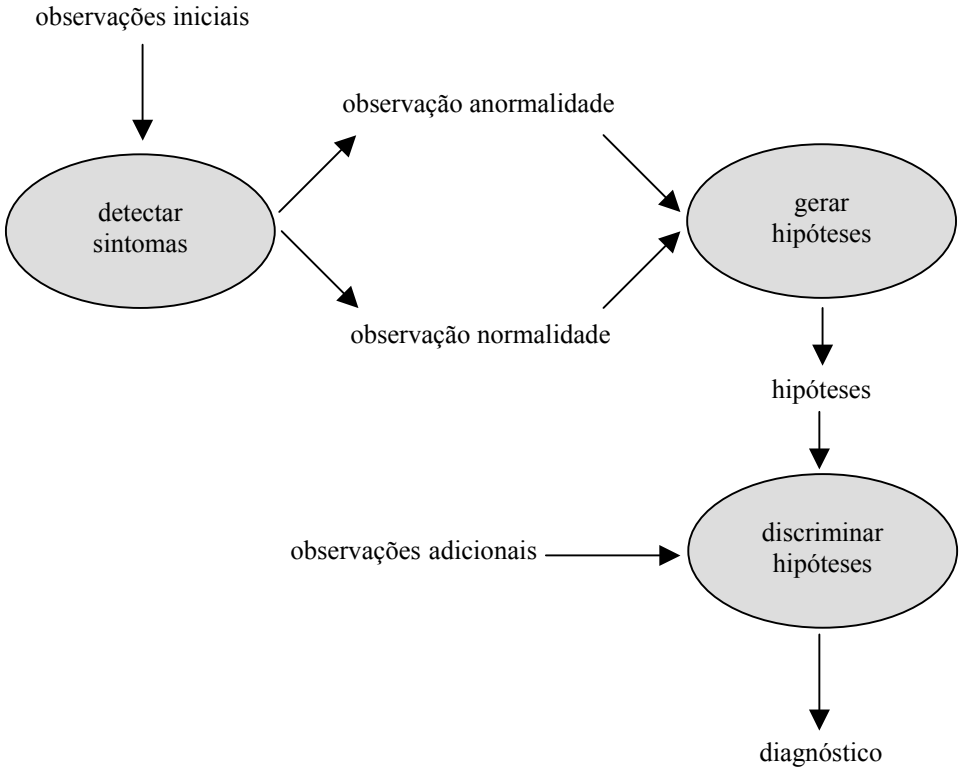


FIGURA 2.1 – Fluxo de dados do *PDM* [BER 95]

A FIGURA 2.1 apresenta a decomposição da tarefa de diagnóstico (ou *Prime Diagnostic Method – PDM*, como é chamada em [BER 95]). Esta tarefa tem como entrada um conjunto de observações iniciais, que são as entradas da tarefa "detectar sintomas". Independente dessas observações serem classificadas como normais ou anormais (no caso de um sintoma), elas serão entradas da tarefa "gerar hipóteses". O conjunto de hipóteses gerado será a entrada para a tarefa "discriminar hipóteses" em que, além das observações iniciais, consideram-se também as observações adicionais. A saída da tarefa de discriminação, que é também a saída do *PDM*, é um conjunto de diagnósticos que explicam tanto as observações iniciais quanto as observações adicionais.

Segundo Viccari [VIC 96], o diagnóstico é um dos principais componentes em um sistema inteligente de ensino, e pode ser estruturado de três maneiras: baseado na especificação, baseado nos sintomas e baseado na cognição.

No diagnóstico baseado na especificação é necessário, inicialmente, obter o modelo ideal do objeto a ser diagnosticado, que não é nada mais que as características corretas do

objeto. O processo de diagnóstico consiste em descobrir as diferenças entre o modelo ideal e o modelo real, e a localização de uma falha consiste na descrição destas diferenças. A partir desta análise é gerado um conjunto de hipóteses de falhas. Cada hipótese é testada e, se o defeito esperado for encontrado, ela e o ponto incorreto podem ser exibidos. Senão, suas táticas são reavaliadas e o processo é reiniciado.

O diagnóstico baseado em sintomas, geralmente é utilizado em sistemas inteligentes modelados de acordo com o domínio do problema. A detecção dos sintomas ocorre através de procedimentos de inferência ou por informação do usuário. Assim, um diagnóstico é o resultado da combinação das inferências e seus sintomas. O processo de diagnóstico pode ser dividido em prevenção, detecção, localização, correção ou previsão da falha.

Quando o diagnóstico se refere a pessoas, isto é, diagnóstico de erros cometidos por usuários, é chamado de diagnóstico cognitivo, pois o modelo é baseado na Psicologia Cognitiva. Segundo Franco [FRA 93], o diagnóstico cognitivo se apóia na modelagem das habilidades (por exemplo: percepção, memória, aprendizado, resolução de problemas, desempenho), na modelagem do comportamento, do conhecimento e objetivos do usuário. Para proporcionar instruções adaptativas ao usuário, o sistema deve conhecer seu estado cognitivo, o que ele sabe, como ele pensa e, de preferência, como ele aprende [OHL 87]. Para isso, são necessárias a representação do conhecimento, a inferência do estado cognitivo do aluno, as ações do sistema necessárias para suportar a interação. É importante planejar com detalhe como deve ser a saída do diagnóstico, que pode ser [FRA 93]:

a) Medida de desempenho: mede a facilidade com que o aluno resolve problemas em determinada área através de testes e/ou exames. É um método essencialmente quantitativo, de fácil aplicação, que suporta apenas ações globais (aumentar ou diminuir o nível de auxílio ou de dificuldade dos exercícios, ou ainda a velocidade de apresentação). Entretanto, não tem capacidade de decidir o que um aluno precisa a cada ponto do aprendizado.

b) *Overlay*: assume que o conhecimento do aluno é um subconjunto do conhecimento do professor (modelo ideal). Enfoca o que o aluno conhece e o que não conhece, sendo incapaz de tratar o que o aluno conhece de errado (falsas concepções).

c) Dicionário de erros: baseia-se num conjunto de erros possíveis a respeito do domínio. Para que este método seja eficiente, é necessário que se tenha o conjunto de todos os erros possíveis, o que se torna impraticável, e além disso se este número de erros possíveis for elevado, o tempo de busca da solução cresce exponencialmente.

d) Simulação: a descrição do estado cognitivo do aluno pode consistir de um modelo de simulação que tenha o mesmo desempenho que ele no domínio de conhecimento. O modelo é executável, ou seja, gera o comportamento do aluno para determinada tarefa, isto é, o que o aluno faria para resolver esta tarefa. Os erros são implicitamente descritos nas regras que compõem o modelo.

## 2.2 Análise do processo de diagnóstico

Esta seção apresenta uma análise do processo de diagnóstico, baseada num *framework* modelado por Benjamins [BER 95], que reflete a convergência de abordagens atuais, em que os principais elementos são: tarefas, métodos para resolução de problemas e conclusões primitivas.

- *Tarefas*. Uma tarefa tem um objetivo e especifica *o que* é necessário atingir. É caracterizada pelo tipo de suas entradas e saídas, e pode ser decomposta em subtarefas (através de um método para resolução de problemas).
- *Métodos para resolução de problemas*. Um método para resolução de problemas (*Problem-Solving Method – PSM*) define *como* atingir o objetivo de uma tarefa. O método tem entradas e saídas, e decompõe uma tarefa em subtarefas e/ou conclusões primitivas. Além disso, um método especifica o fluxo de dados entre os constituintes. O mecanismo de controle do conhecimento determina a ordem de execução e as interações das subtarefas e das conclusões de um *PSM*. Este controle pode ser especificado com antecedência, se conhecido, ou pode ser oportunamente determinado em tempo de execução em situação dinâmica de resolução de problemas.
- *Conclusões primitivas*. Uma conclusão primitiva (ou conclusão) define um passo de raciocínio que pode ser seguido para atingir um objetivo. Especifica como o objetivo pode ser atingido usando o conhecimento do domínio. As conclusões formam os blocos de construções atuais de resolução de problemas. Quando todas as tarefas são decompostas em conclusões, através de *PSMs*, a estrutura do fluxo de dados correspondente forma uma estrutura de conclusão. Se o mecanismo de controle do conhecimento também é especificado, tem-se uma estratégia de resolução de problemas.

A FIGURA 2.2 ilustra a relação entre tarefas, *PSMs* e conclusões. Uma tarefa pode ser realizada por vários métodos e consiste em subtarefas e/ou conclusões. As mesmas conclusões, tarefas e métodos aparecem em vários lugares (por exemplo, *conclusão primitiva 1*) porque podem ser reutilizáveis.

Baseado na literatura, onde são apontadas várias abordagens de diagnóstico (de Kleer e Williams, 1987; Davis, 1984; Genesereth, 1984; Console e Torasso, 1990), Benjamins [BER 95] identificou trinta e oito *PSMs* e catorze tarefas relevantes para diagnóstico. A seguir, será descrita a decomposição básica das três subtarefas do diagnóstico: detectar sintomas (FIGURA 2.3), gerar hipóteses (FIGURA 2.4) e discriminar hipóteses (FIGURA 2.5). Nas figuras, os métodos são representados por retângulos e as tarefas por elipses. Além disso, referências relevantes a sistemas de diagnóstico pertencentes a cada classe também são incluídas (circuladas por linhas pontilhadas).

*Tarefa detectar sintomas* (FIGURA 2.3). Existem vários métodos e subtarefas para detectar a normalidade ou anormalidade das observações. A FIGURA 2.3 apresenta estes métodos e a forma como o método de comparação pode ser decomposto em subtarefas e conclusões. O método "comparação de sintomas detectados" gera o valor esperado para a observação de entrada e compara a observação com a expectativa, resultando em "igual" ou "não igual". Se a observação e a expectativa forem julgadas iguais, a observação é considerada uma observação de normalidade. Caso contrário, a observação é considerada uma observação de anormalidade (sintoma). O método "classificação de sintomas detectados" assume a disponibilidade do conhecimento de classificação do domínio e, baseado nisto, uma observação é classificada como normal ou anormal. O método "uso de sintomas detectados" confia no conhecimento do usuário para determinar o *status* de uma observação.

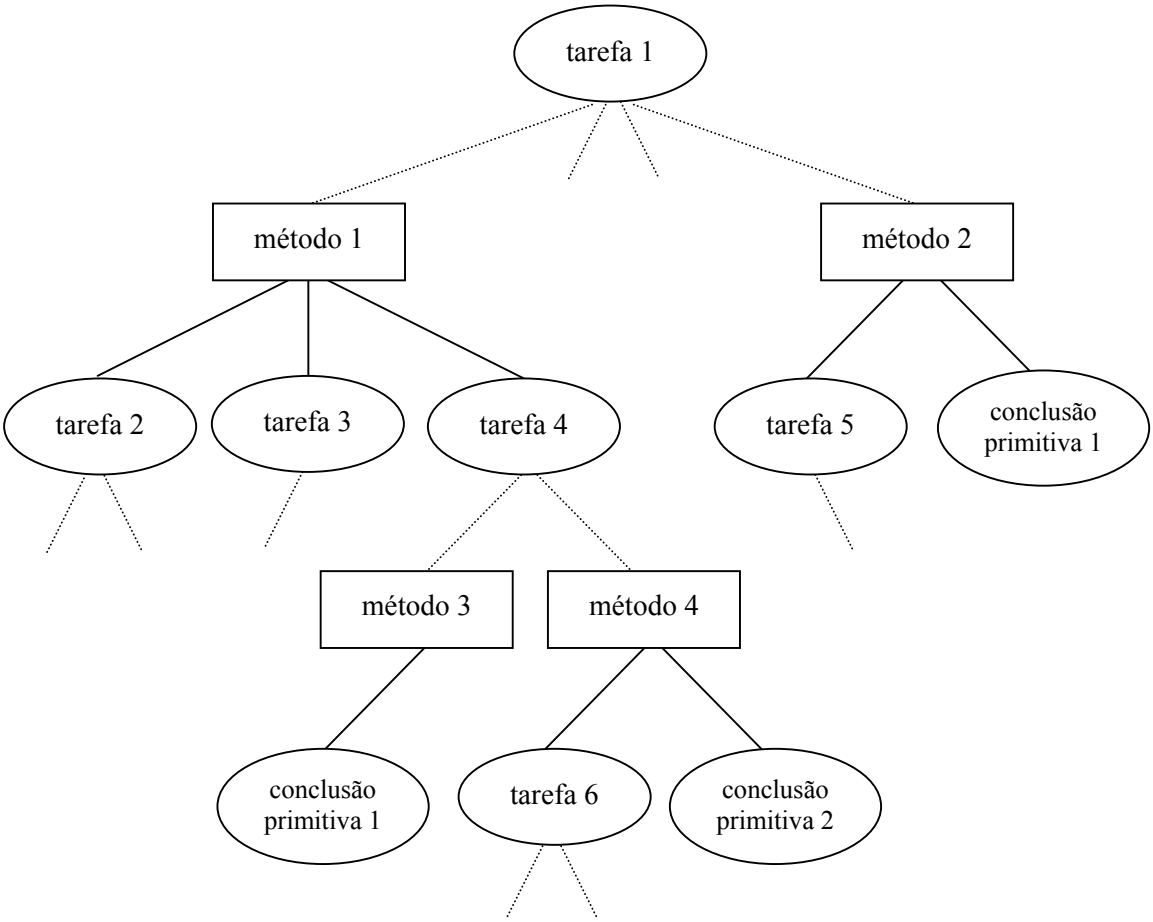


FIGURA 2.2 – Relações entre tarefas, métodos e conclusões [BER 95]. Linhas pontilhadas indicam que os métodos são alternativos para a realização da tarefa. Linhas sólidas decompõem um método em subtarefas e/ou conclusões.

*Tarefa gerar expectativa* (FIGURA 2.3). Dois métodos podem ser utilizados para gerar uma expectativa ou uma observação. O método "*lookup*", que consulta uma base de dados onde todos os valores das expectativas relevantes estão armazenados, e o método "predição", que calcula valores esperados baseado na simulação de um dispositivo modelo.

*Tarefa comparar* (FIGURA 2.3). Esta tarefa decide se uma observação e uma expectativa são iguais ou não. Vários métodos são identificados para esta tarefa, refletindo diferentes precisões de comparação. O método "comparação exata" compara diretamente uma observação e uma expectativa, e é apropriado em domínios onde os valores são exatos. Quando há tolerância, como em sistemas de comportamento, outros métodos são necessários. No método "comparação em ordem de magnitude", uma observação e uma expectativa que diferem mas estão na mesma ordem de magnitude, são consideradas iguais. O método "comparação com limiar" testa se a diferença entre a observação e a expectativa excede ou não um determinado limiar. O método "comparação teleológica" executa uma abstração teleológica de uma observação antes de compará-la à expectativa, para prevenir alarmes falsos; uma observação anormal (sintoma) é, deste modo, uma discrepância de nível teleológico. No método "comparação estatística", uma expectativa é expressa em termos de informação estatística e é comparada com a observação. O método "comparação histórica" utiliza dados históricos de um dispositivo particular para decidir se uma diferença entre uma observação e sua expectativa deve ser considerada uma discrepância.

*Tarefa gerar hipóteses* (FIGURA 2.4). Esta tarefa gera hipóteses que explicam o comportamento do conjunto de observações iniciais (normais e anormais). As observações normais contribuem como uma força discriminatória extra no processo de geração de hipóteses. A FIGURA 2.4 apresenta uma avaliação dos métodos e tarefas relevantes para a geração de hipóteses. O método "geração de hipóteses compiladas" explora associações entre sintomas e causas e, às vezes, é seguido por um filtro de probabilidade. Geralmente, este método é utilizado em domínios em que a compreensão de mecanismos subjacentes é parcial, como em diagnóstico médico. O método "geração de hipóteses baseadas em modelo" consiste em três subtarefas: encontrar um conjunto de entidades modelo que contribuem para uma observação anormal (um conjunto de candidatos tem pelo menos um elemento violado); transformar o conjunto de candidatos em um conjunto de hipóteses, em que cada elemento é uma explicação possível para as observações; utilizar filtros baseados em predições para descartar hipóteses adicionais incompatíveis com o conjunto de hipóteses (apesar de fazer parte da geração de hipóteses, filtrar é uma subtarefa opcional).

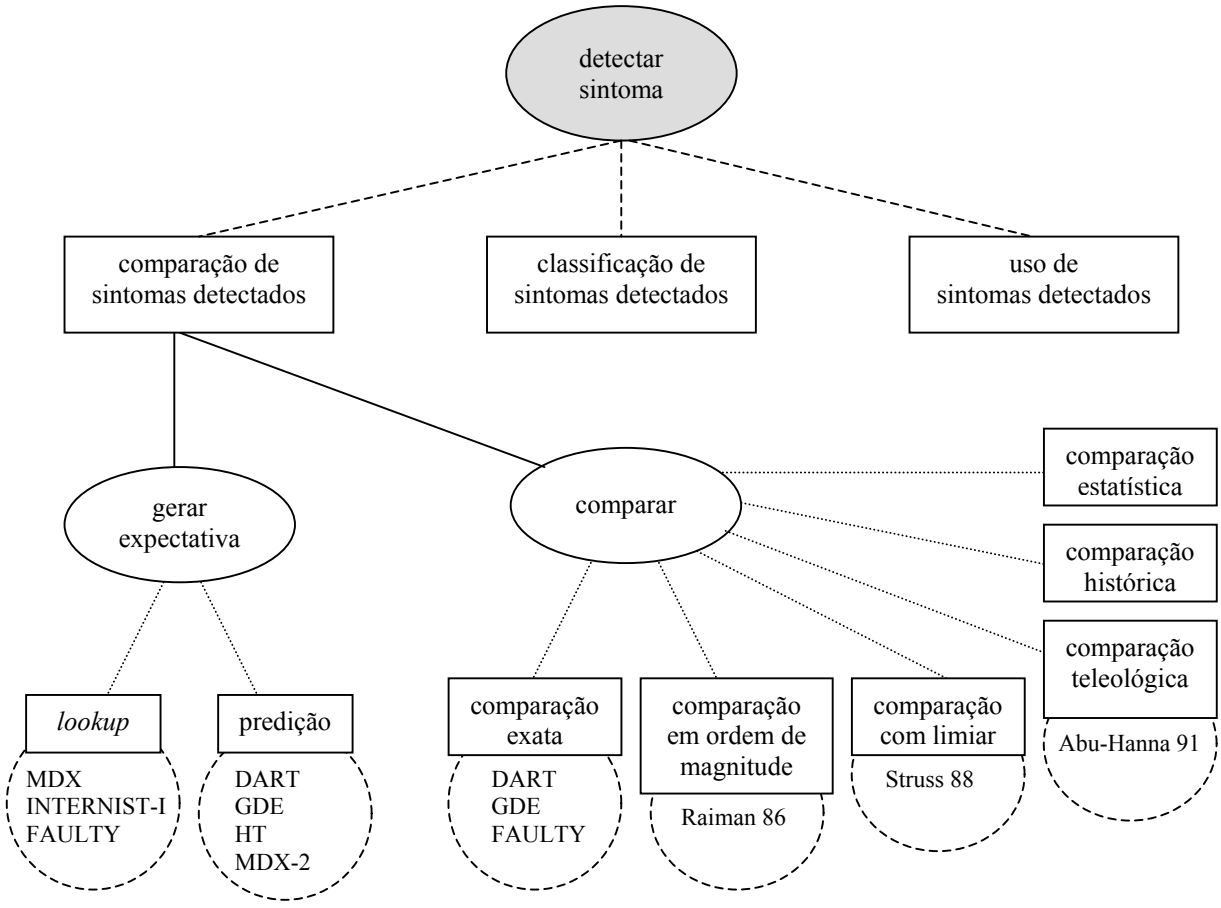


FIGURA 2.3 – Decomposição parcial da tarefa detectar sintomas [BER 95]

*Tarefa encontrar candidatos* (FIGURA 2.4). O método "*trace back*" encontra os candidatos para uma observação anormal, explorando a rede de representação do modelo e localizando um modelo do domínio que represente o comportamento correto. O método "cobertura causal" opera sobre um modelo causal e tem uma natureza abdutiva. A representação da rede do modelo causal determina se os candidatos devem ser gerados incrementalmente ou todos de uma vez. No método "predição", os conjuntos de candidatos

são resultado de um processo de simulação. Durante a simulação, as justificativas para os valores calculados são gravadas. Se uma inconsistência é encontrada entre o resultado da simulação e as observações atuais, os candidatos para esta inconsistência podem ser encontrados nas justificativas armazenadas.

*Tarefa transformar conjunto de hipóteses* (FIGURA 2.4). Uma maneira de transformar um conjunto de candidatos em um conjunto de hipóteses é aplicar o método "conjunto básico", que gera todas as hipóteses que tenham uma interseção não-vazia com qualquer conjunto de candidatos. Este método assume completa independência de causa, o que significa que uma causa individual explica um conjunto de observações independente de outras causas que estão sendo consideradas. O método "*set-cover*" não aplica nenhum critério de parcimônia para podar o conjunto de hipóteses; ele gera todas as hipóteses possíveis consistentes com as observações no momento. Três critérios diferentes de parcimônia fortalecem este método: o método "cardinalidade minimizada", que prefere hipóteses com baixa cardinalidade; o método "subconjunto minimizado", que descarta hipóteses que estão no conjunto de outras hipóteses; o método "interseção", em que todos os conjuntos de candidatos são incrementados e a interseção (não-vazia) gera as hipóteses. Se uma interseção contém mais que um elemento, cada elemento é considerado como uma hipótese separada.

*Tarefa filtrar baseada em predições* (FIGURA 2.4). Esta tarefa utiliza o processo de predição, baseando-se somente em observações iniciais, para mais adiante excluir hipóteses inconsistentes. O método "suspensão de limitação" suspende, para cada hipótese do conjunto de hipóteses, o comportamento da entidade (ou entidades) do modelo correspondente. O restante do modelo é simulado considerando as observações iniciais. Se o resultado do processo de simulação é consistente, a hipótese é válida e permanece no conjunto de hipóteses. Caso contrário, a hipótese não pode explicar de maneira consistente as observações iniciais e é descartada. Quando um valor observado atinge sua expectativa, a confirmação acontece. O método "confirmação" considera que qualquer hipótese (entidade no modelo) envolvida em uma confirmação pode ser inocente e, por isso, pode ser descartada do conjunto de hipóteses. Existem duas formas para decidir o envolvimento de uma hipótese em uma confirmação: uma simulação detalhada ou uma análise de discrepância. Usando a noção de confirmação para podar o conjunto de hipóteses, assume-se que aquela falha mascarada não ocorra. O método "simulação de falha" instancia um modelo de falha correspondendo a uma das hipóteses. Se o comportamento predito desse modelo de falha é inconsistente com as observações, as hipóteses podem ser descartadas (assumindo só um modelo de falha para cada hipótese).

*Tarefa discriminar hipóteses* (FIGURA 2.5). Nesta tarefa, observações adicionais são reunidas e interpretadas para discriminar as hipóteses. A saída da tarefa de discriminação é um conjunto de diagnósticos, cada um explicando uma observação inicial e adicional. A FIGURA 2.5 apresenta uma decomposição parcial desta tarefa. O método de discriminação consiste em quatro subtarefas: "selecionar hipóteses", "coletar dados", "detectar sintomas" e "atualizar hipóteses", descritas a seguir (a tarefa detectar sintomas foi descrita anteriormente).

*Tarefa selecionar hipóteses* (FIGURA 2.5). Nesta tarefa uma hipótese requer que dados adicionais sejam selecionados, podendo ser realizada de várias maneiras. O método "seleção randômica" é uma abordagem simples e útil quando as relações entre as hipóteses não estão ordenadas. No método "seleção inteligente", todas as hipóteses são associadas a um custo estimado para testar todo o conjunto de hipóteses, iniciando por uma hipótese

específica. A estimativa pode ser baseada em diferentes tipos de conhecimento, dependendo do domínio. Por exemplo, ela pode ser baseada em custos locais ou no número de testes esperados para serem executados, ou uma combinação dos dois (custos globais). As hipóteses são ordenadas de acordo com o custo estimado.

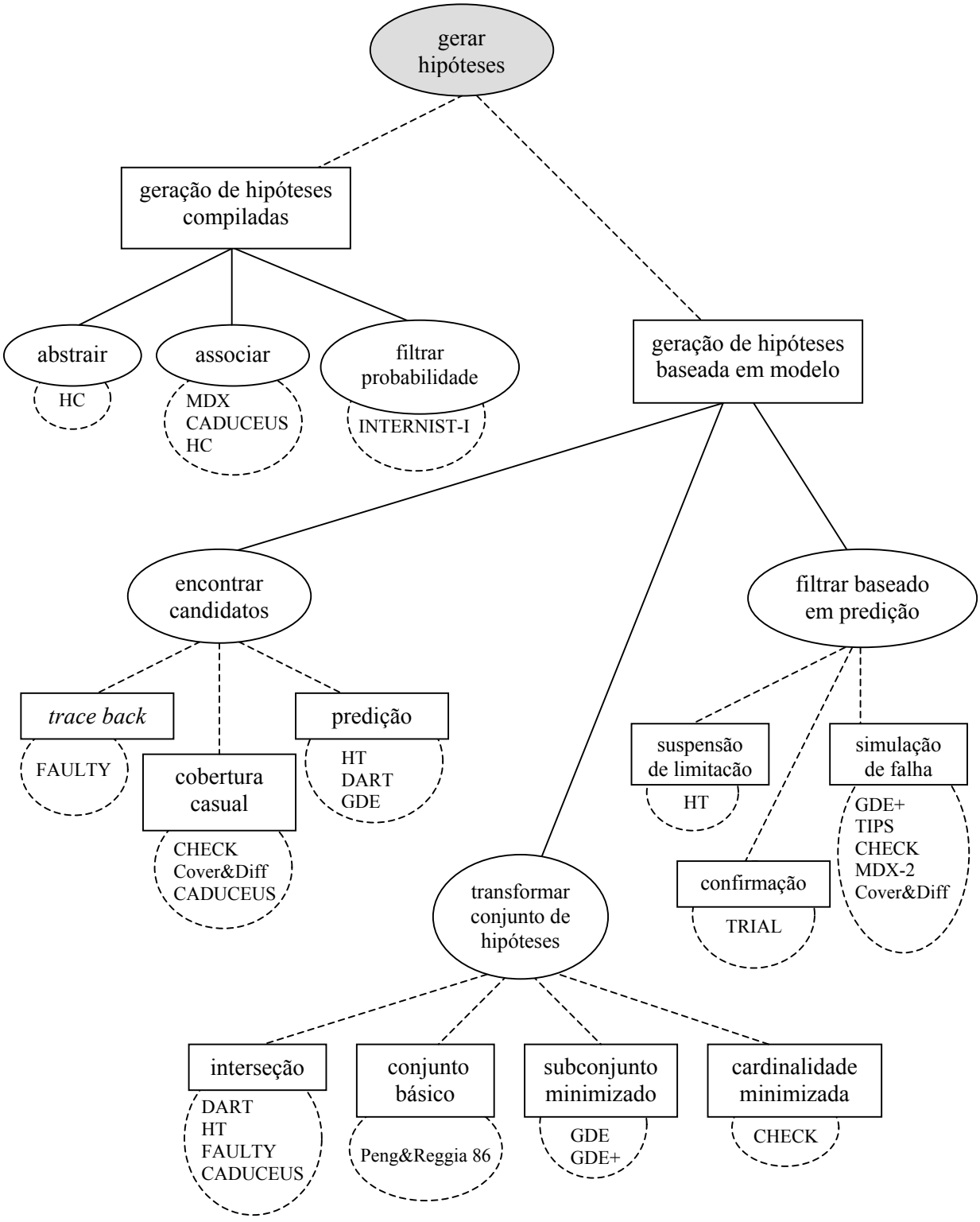


FIGURA 2.4 – Decomposição parcial da tarefa gerar hipóteses [BER 95]

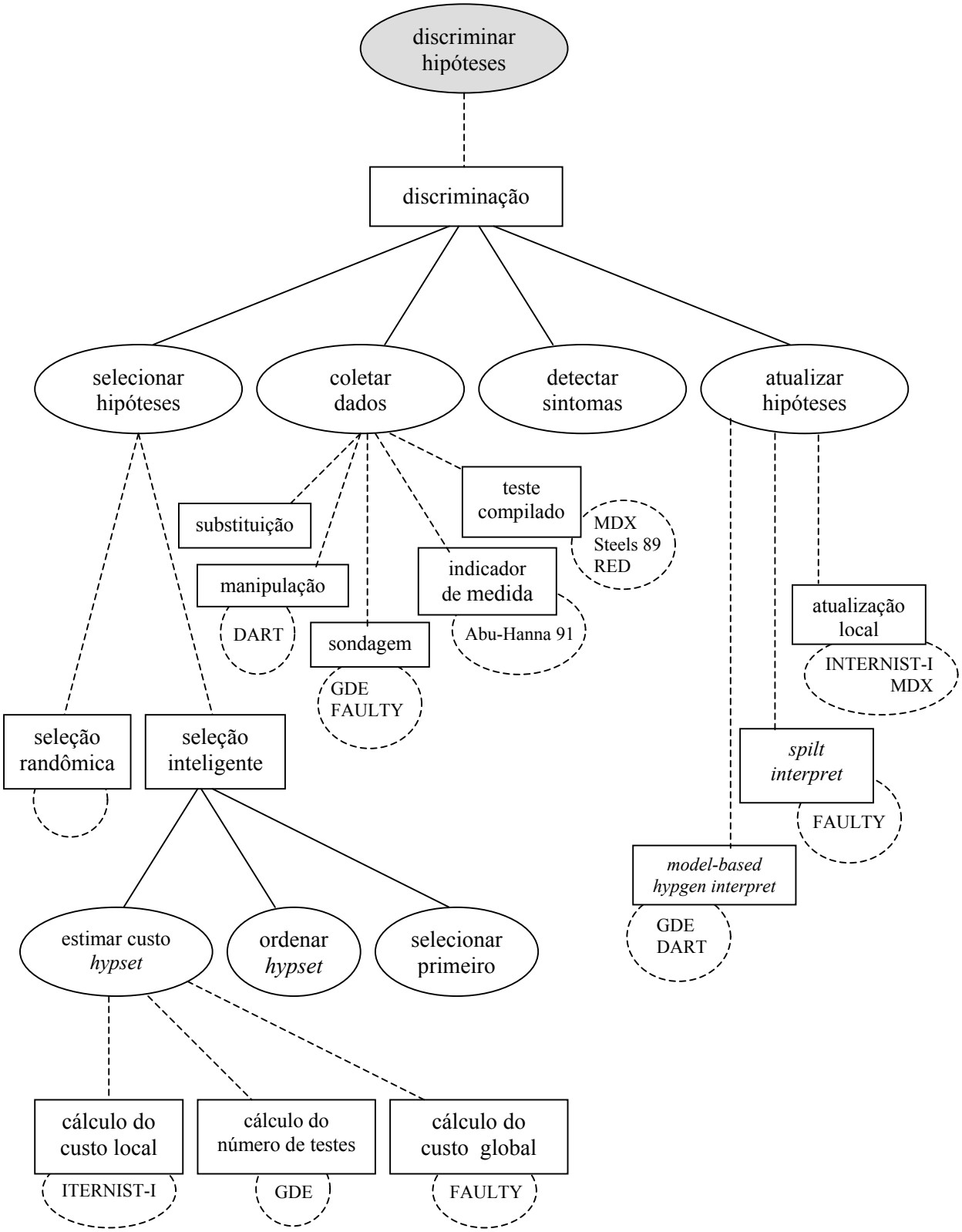


FIGURA 2.5 – Decomposição parcial da tarefa discriminar hipóteses [BER 95]

*Tarefa coletar dados* (FIGURA 2.5). Esta tarefa adquire observações adicionais. O método "teste compilado" requer, para cada hipótese, que o conhecimento esteja disponível sobre os procedimentos de testes. Cada nodo (hipótese) em uma classificação hierárquica é um especialista e sabe como se ajustar às observações. O método "sondagem" obtém observações adicionais através da sondagem de algumas medidas de pontos. As observações são obtidas de componentes que correspondem às hipóteses, e é necessário que esses componentes estejam acessíveis para as observações. Quando é difícil obter as medidas dos pontos, o método "indicador de medida" providencia uma maneira. Um indicador é uma observação alternativa, porém parcial, inexata ou indireta, que determina uma indicação do *status* das hipóteses. Outro método para coletar dados adicionais é o método "manipulação", que força as entradas de um dispositivo e observa os resultados. No método "substituição", o componente suspeito é substituído por um componente correto.

*Tarefa atualizar hipóteses* (FIGURA 2.5). Depois de saber se uma observação é normal ou anormal, ela deve ser interpretada com base no conjunto de hipóteses de forma que isto explique as observações iniciais e adicionais. Esta tarefa pode envolver o mesmo raciocínio que a tarefa geração de hipóteses, porque ao invés de utilizar informações iniciais como na geração de hipóteses, utilizam-se observações adicionais para atualizar o conjunto de hipóteses. Um método simples de atualização é interpretar os resultados dos testes localmente, ou seja, ou a hipótese está descartada ou continua dependendo do dado adquirido. Esta é uma abordagem útil quando as hipóteses estão sem conexão. Se as hipóteses estão de alguma forma relacionadas umas às outras, e uma simples suposição de falha acontece, o método *interpret split* pode ser aplicado, no qual o conjunto de hipóteses é dividido em duas partes e uma é descartada, dependendo do *status* das observações adicionais. Outro método para interpretar dados adicionais está no princípio da geração de hipóteses com base em modelo, com adição de novas observações.

# 3 Diagnóstico de falhas em dispositivos físicos

Na literatura sobre teoria e implementação de sistemas de raciocínio para diagnóstico, aparecem várias abordagens distintas: a abordagem tradicional, baseada no *Algoritmo D* [ROT 66]; a abordagem denominada experimental, em que as heurísticas têm um papel dominante [ABR 80] e, uma terceira abordagem, baseada em princípios básicos (*first principles*), que usa a descrição da estrutura de um dado sistema (partes de um dispositivo físico e suas interconexões) aliada a um conjunto de observações do comportamento do sistema. Se a observação conflita com o comportamento esperado, tem-se um problema de diagnóstico: determinar aqueles componentes do sistema que, uma vez funcionando anormalmente, explicam a discrepância entre o comportamento correto e o observado.

Esta última abordagem despertou o interesse da comunidade de Inteligência Artificial na década de 80, pelas razões descritas a seguir [BAZ 93]:

- raciocinar a partir de princípios básicos oferece um significativo grau de independência do dispositivo. As ferramentas que trabalham diretamente em cima de descrições de estrutura e comportamento tornam explícitas as suposições sobre as descrições;
- um sistema baseado em raciocínio sobre princípios básicos é mais fácil de construir porque há uma maneira de enumerar sistematicamente o conhecimento requerido: a estrutura e o comportamento do dispositivo;
- um sistema baseado em raciocínio sobre princípios básicos também é mais fácil de manter, já que as modificações no projeto do dispositivo são relativamente mais fáceis de acomodar. Podem-se modificar as especificações de estrutura e de comportamento, em vez de ter que determinar como cada mudança deve modificar o comportamento global e a estratégia de detecção de falha;
- a habilidade para enumerar o conhecimento sistematicamente ajuda na definição do escopo do programa e da sua competência. Sabem-se quais partes da máquina foram modeladas e em qual nível de detalhe;
- raciocínio sobre princípios básicos oferece a possibilidade de lidar com fatos novos. O sistema não depende de um catálogo de manifestações de erros observados, já que ele tem uma visão de que qualquer discrepância entre o comportamento observado e o esperado é um *"bug"*, e usa conhecimento sobre a estrutura do dispositivo para determinar as possíveis fontes desse *"bug"*. Como resultado, ele é capaz de raciocinar sobre *"bugs"* inéditos, no sentido de não serem parte do "conjunto de treinamento" e que são manifestados por sintomas não previamente vistos.

Vários autores como Davis, Genesereth, Reiter, de Kleer e Pipitone, abordam a teoria do raciocínio sobre princípios básicos, cada qual tratando de implementar um algoritmo de diagnóstico com características próprias. Praticamente, todas as abordagens baseiam-se em descrições de estrutura e comportamento. O que varia é a implementação destas descrições: lógica de primeira ordem, representação esquemática (topologia), cálculo proposicional, etc. As grandes variações encontram-se no modo de reconhecer e computar candidatos a falha: Davis [DAV 84] adota a técnica de suspensão de restrições; Reiter [REI 87] usa um provador de teoremas sobre a descrição lógica dada; de Kleer [DEK 87] sugere um algoritmo que manipula múltiplos candidatos e pode lidar com probabilidades; Genesereth [GEN 84] baseia-se em álgebra *booleana*; Pipitone [PIP 86] usa um editor

gráfico para captar a estrutura e regras para a seleção de candidatos. As próximas seções descrevem estas abordagens que, apesar de se adaptarem bem a ambientes estáticos, não se adaptam de maneira satisfatória ao diagnóstico cognitivo.

## 3.1 Abordagem de Davis

A abordagem de Davis baseia-se nos itens descritos a seguir [BAZ 93]:

a) Descrição da estrutura sob visões funcional e física.

b) Descrição do comportamento baseada em restrições.

c) Caminhos de interação causal: como um componente pode afetar outros.

d) Geração de candidatos utilizando técnica de suspensão de restrições: uma vez modelado o comportamento como uma rede de restrições interconectadas, se o dispositivo funciona anormalmente, então as saídas previstas não casam com as reais. O tratamento destas contradições ou inconsistências é feito perguntando-se "há alguma restrição (comportamento de um componente) cuja retirada levará a rede a um estado consistente?". Cada restrição encontrada corresponde a um componente cujo comportamento pode ser responsável pelo sintoma observado.

e) Categorias de falhas: ferramenta para tratar o dilema complexidade *versus* completeza. A cada representação está associado um conjunto de suposições "do que é mais provável dar errado", que pode ser organizado em uma lista de categorias de falhas. A cada categoria está associada uma coleção de caminhos de interação; logo, tem-se uma ordenação dos caminhos a serem considerados, o que permite restringir os caminhos iniciais, tornando possível restringir a geração de candidatos sem, contudo, excluir permanentemente qualquer caminho.

f) Adjacência: os dispositivos interagem porque eles são, de alguma forma adjacentes – eletricamente (montados juntos), fisicamente (conectados termalmente), eletromagneticamente (não isolados), etc. Cada uma destas definições pode ser usada como base para uma representação diferente do dispositivo. A multiplicidade de definições possíveis ajuda a explicar por que algumas falhas são especialmente difíceis de diagnosticar: elas resultam da interação entre componentes que são adjacentes em uma forma (representação) que é sutil ou pouco usual. Adjacência pode, ainda, ajudar a determinar uma boa representação, isto é, aquela na qual os componentes estejam compactamente representados num nível de abstração tal que a falha seja local a um componente.

g) Singularidade de causas: uma vez tendo uma boa representação, proveniente de alguma forma de adjacência, é possível determinar se a hipótese de "uma única causa inicial" se mantém.

Depois de descrever as organizações funcional e física e o comportamento do sistema, o terceiro passo é o mecanismo de detecção de falhas que trabalhe sobre estas descrições.

Segundo Bazzan [BAZ 93], a abordagem tradicional para detecção de falhas apresenta uma série de defeitos, por envolver a realização de um conjunto completo de testes e por não ajudar na determinação de qual falha considerar ou de qual componente suspeitar, o que leva ao problema da complexidade: o tamanho do dispositivo é determinante no tamanho das árvores de decisão para todas as falhas possíveis.

O uso de “detecção de discrepância” é uma resposta para os problemas da abordagem tradicional. Os dois pontos básicos desta técnica são: i) a substituição de expectativas violadas por modelos de falhas específicas e, ii) o uso de registros de dependência para rastreamento até a possível fonte da falha. Em vez de varrer cada falha possível e explorar suas conseqüências, esta técnica procura discrepâncias entre valores esperados em uma operação normal e aqueles realmente obtidos. Qualquer componente no caminho entre uma entrada e uma saída incorreta pode ter sido responsável pelo comportamento incorreto. O simulador constrói uma rede de dependência, através do registro da propagação de valores, ao simular o circuito. O uso destes registros como guia para quais componentes examinar parece ser um modo efetivo de focalizar atenção nos candidatos, porém pode trazer problemas na determinação da consistência dos candidatos.

Uma das contribuições do trabalho de Davis [DAV 84] é o desenvolvimento da técnica da “suspensão de restrições” como reforço sobre a detecção de discrepâncias a fim de lidar com a consistência. A rede de restrições pode indicar se se tem um conjunto consistente de valores alocados nas entradas e saídas. Se, por exemplo, for observado um valor incorreto – distinto do previsto – em uma saída, será relatada uma contradição: não há como todas as regras serem efetivas, isto é, todos os componentes trabalhando conforme esperado.

Normalmente, as contradições em redes de restrições são manuseadas através da retirada de um dos valores inseridos na rede. Aqui, porém, tem-se certeza dos valores (conhecidos e medidos), mas não das restrições (comportamento dos componentes). Portanto, toma-se uma visão dual e, em vez de procurar por um valor para retirar, procura-se uma restrição cuja retirada levará a rede a um estado consistente. Se a rede atinge este estado, sabe-se que o candidato retirado pode ser responsável pelos sintomas. Há algumas suposições importantes decorrentes deste raciocínio, entretanto a técnica de suspensão de restrições fornece um mecanismo de gerenciamento destas suposições. A abordagem tradicional para diagnóstico assume que se conhece como o componente pode estar falhando: exibindo um dos comportamentos incorretos encontrados no conjunto de modelos de falha. Davis [DAV 84], por outro lado, procede através da suspensão de todas as suposições sobre como um componente pode estar se comportando, o que permite aos sintomas informar o que os componentes podem estar fazendo.

## 3.2 Abordagem de de Kleer

Nesta abordagem foi implementado o sistema *GDE* (*General Diagnosis Engine*), testado no domínio de circuitos digitais, com características inéditas como a capacidade de lidar com múltiplas falhas, o procedimento incremental de cálculo dos candidatos através de conjuntos mínimos de suposições violadas, a separação clara entre diagnóstico e previsão do comportamento, a combinação de previsão baseada em modelo com diagnóstico seqüencial, propondo novas medidas para localizar as falhas e a incorporação de probabilidade condicional, calculadas a partir da estrutura do dispositivo e do modelo de seus componentes [DEK 87].

Tratando a questão das múltiplas falhas, há a possibilidade de que o espaço de candidatos potenciais cresça exponencialmente com o número de falhas consideradas. Para lidar com esse problema de ineficiência, explora-se a suposição da manutenção da verdade. Para calcular a melhor medida a ser realizada para isolar o conjunto de componentes, são necessárias as probabilidades de falha *a priori* de cada componente individual. O processo

de diagnóstico é guiado por sintomas. Cada sintoma refere-se a uma ou mais suposições que possivelmente foram violadas (por exemplo, componentes que podem ter falhado). A meta do diagnóstico é identificar e refinar o conjunto de candidatos consistente com as observações.

Um candidato é uma hipótese particular de como o dispositivo real difere do modelo (projeto). Deve-se identificar o conjunto completo mínimo de candidatos. Este conjunto de candidatos é construído em dois estágios: reconhecimento de conflito (conflito é o conjunto de suposições que suportam um sintoma levando a uma inconsistência) e geração dos candidatos. O reconhecimento de conflito usa as observações feitas bem como o modelo do dispositivo. A seguir, para gerar os candidatos, usa-se o conjunto mínimo de conflitos para construir um conjunto completo mínimo de candidatos.

## 3.3 Abordagem de Reiter

Reiter [REI 87] propõe uma teoria geral para o problema do diagnóstico, que requer que o sistema seja descrito em uma lógica apropriada, e leva a um algoritmo capaz de computar todos os diagnósticos.

A generalidade necessária é obtida através da utilização de lógica de primeira ordem, como linguagem para descrição dos sistemas. Um sistema é definido como um par (*DS, componentes*), onde *DS* é a descrição do sistema ou um conjunto de sentenças de primeira ordem, e os componentes são um conjunto de constantes ($c_1$, $c_2$, $c_n$). É definido ainda um predicado unário *AB* interpretado como anormal. Pode-se escrever (*DS*, *componentes*, *OBS*), para um sistema (*DS, componentes*) com observações *OBS*, pois o diagnóstico de um sistema envolve observações. Em cima desta descrição traduzida em lógica, é derivado um algoritmo que computa todos os diagnósticos para um dado sistema.

Ao fim do desenvolvimento da teoria, conclui-se que poucas características da lógica de primeira ordem foram requeridas. Assim, a linguagem de representação pode ser generalizada. Adicionalmente, foi feita uma relação com lógica *default* após a observação que o raciocínio do diagnóstico é não monotônico no sentido de que pode acontecer que nenhum dos diagnósticos do sistema sobreviva a uma nova observação.

## 3.4 Abordagem de Genesereth

O *DART* é um programa para diagnóstico independente do dispositivo [GEN 84]. O sistema difere das abordagens anteriores por ser mais geral, usando uma linguagem independente do dispositivo para descrever o sistema, e um procedimento de inferência para o diagnóstico. O programa deve ser usado juntamente com um testador que possa manipular e observar um dispositivo em funcionamento anormal. O módulo diagnosticador recebe do testador uma descrição de falha, prescreve testes e recebe resultados e, por fim, identifica os componentes falhos responsáveis pelo desvio.

A característica de independência do dispositivo, tanto da linguagem de descrição quanto do procedimento de diagnóstico em si, torna mais fácil aplicar o programa em vários níveis de abstração e explorar a hierarquia inerente a muitos projetos de computação. É decorrência, ainda, o uso de técnicas de propagação de restrições (limitações), o que evita as combinações desnecessárias.

O *DART* é um diagnosticador que trabalha diretamente sobre informações a respeito do projeto relativo a um dispositivo em funcionamento anormal. Entende-se por projeto qualquer arranjo do mundo real que atinja um resultado desejado, e por dispositivo, qualquer implementação de um projeto. Dado um conjunto de sintomas, o sistema gera testes, aceita os resultados e, por fim, aponta pelo menos uma falha no dispositivo ou uma direção não diagnosticada que contém falha.

O sistema começa com a descrição de projeto para um dispositivo falho e uma descrição parcial do comportamento do mesmo. Inicialmente, computa um conjunto de proposições suspeitas (candidatos). Se esse conjunto contém somente um elemento, o diagnóstico está encerrado. Senão, o *DART* tenta gerar um teste para discriminar os suspeitos. Se obtém sucesso, o teste é executado, suas conseqüências são calculadas e o processo se repete. Senão, retorna o conjunto de suspeitos como saída.

Por razões de eficiência, o *DART* não gera uma árvore de decisão completa antes de executar os testes.

A abstração estrutural de uma descrição de projeto, em que detalhes estruturais são omitidos, é uma vantagem para diagnóstico, pois freqüentemente é possível diagnosticar falhas sem considerar os detalhes omitidos. Por outro lado, a perda de informação pode levar a não diagnosticabilidade. Outro nível de abstração refere-se à abstração comportamental de uma descrição de projeto, em que detalhes de comportamento são omitidos. Para muitos projetos, a abstração comportamental é determinada pela hierarquia estrutural. O comportamento de componentes em um nível na hierarquia estrutural é, freqüentemente, descrito em termos diferentes do comportamento em outros níveis.

## 3.5 Abordagem de Pipitone

O *FIS* (*Fault Isolation System*), descrito em [PIP 86], é um sistema especialista que auxilia um técnico no diagnóstico de falhas em partes de um equipamento eletrônico. O *FIS* foi projetado principalmente para diagnosticar sistemas analógicos, isolando falhas, em vez de amplificadores, geradores e grandes componentes.

Assume-se que o engenheiro do conhecimento tem uma documentação, descrevendo a função e estrutura de partes específicas de um mecanismo eletrônico, chamado de unidade em teste (*UUT*) e utiliza o *FIS* para criar um modelo computacional dessa unidade. Sob supervisão de um técnico, o *FIS* posteriormente usa o modelo para recomendar testes e analisa os resultados dos mesmos, até que os componentes falhos sejam identificados.

Entre as metas do projeto destacam-se [BAZ 93]: minimizar a dificuldade de aquisição do conhecimento; calcular a probabilidade de que uma hipótese de falha esteja correta, recomendando ao técnico o próximo teste a realizar, para maximizar o ganho de informações e minimizar o custo de medição. A capacidade de realizar uma aquisição de conhecimento eficiente, a habilidade de raciocinar com dados qualitativos e um método de raciocínio probabilístico especializado, para detecção de falhas neste tipo de dispositivo, são características importantes do *FIS*.

A abordagem básica é a de caminhar por regras causais locais para obter dinamicamente todas as causas possíveis dos testes com resultados anormais. O sistema não assume que há uma única falha na *UUT*. Igualmente, não são usadas associações globais e empíricas. A modelagem no *FIS* pode ser vista como superficial para componentes individuais e profunda, para uma *UUT* como um todo.

# 4 Diagnóstico cognitivo baseado em modelo

Segundo Ohlsson [OHL 87], o propósito do diagnóstico cognitivo é "testar se as expectativas pressupostas pelo plano atual do sistema estão consistentes com o modelo do aluno". Como estas expectativas podem mudar, o processo de diagnóstico precisa ser explícito, de forma que isto possa incorporar uma análise de questões problemáticas, que devem ser controladas diferentemente em cada contexto. Por isso, Self [SEL 93] acredita que uma maneira de realizar diagnóstico cognitivo é definir um procedimento meta diagnóstico explícito, em que expectativas e objetivos são definidos e raciocinados dinamicamente.

Este capítulo é dedicado ao diagnóstico cognitivo baseado em modelo. A seção 4.1 apresenta um mapeamento entre diagnóstico baseado em modelo do comportamento do dispositivo e do comportamento do aluno. O sistema de diagnóstico cognitivo proposto por Koning [KON 2000], é descrito na seção 4.2. Algumas questões relevantes sobre diagnóstico cognitivo são discutidas na seção 4.3.

## 4.1 Diagnóstico de dispositivos físicos x diagnóstico cognitivo

As teorias de Davis [DAV 84], Reiter [REI 87] e de Kleer [DEK 87] proveram *frameworks* gerais para diagnóstico que, na realidade, são interessantes para diagnóstico de dispositivos físicos como um circuito lógico, por exemplo. Porém, o diagnóstico cognitivo tem problemas para identificar desvios de conhecimento (erros, falsas concepções) na base de conhecimento do aluno. Para resolver isto, Self [SEL 93] propõe algumas extensões aos *frameworks* gerais, como utilizar modelos hierárquicos, modelos de erros, evidências do diagnóstico, erros sistemáticos e deslizes.

No diagnóstico de dispositivos físicos baseado em modelo, a idéia básica é a detecção de erros como a discrepância entre o comportamento de um dispositivo e a predição do comportamento feito pelo uso de um modelo de dispositivo. Assim, todo comportamento errôneo é definido através de divergências do comportamento esperado (ou seja, do comportamento representado no modelo). Conseqüentemente, diagnósticos baseados em modelos não incorporam necessariamente catálogos de erros, e deste modo não dependem de catálogos de erros perfeitos. Inicialmente, todos os componentes são assumidos como operações corretas. No caso de uma discrepância, estas suposições são reconsideradas.

A FIGURA 4.1 [KON 2000] ilustra o mapeamento em alto nível entre diagnóstico baseado em modelo do comportamento do dispositivo e do comportamento do aluno. No sistema de diagnóstico cognitivo proposto por Koning, modelos são gerados automaticamente por um simulador qualitativo, e divergências são diagnosticadas, concluindo-se que o aluno não aplicou corretamente determinadas observações. O componente de diagnóstico utilizado é o *GDE*, que pode ser aplicado a diferentes domínios, desde que estes obedeçam aos princípios de modelagem requeridos pelo algoritmo de diagnóstico. O *GDE* [DEK 87], descrito na seção 3.2, é uma ferramenta de diagnóstico capaz de prover um *framework* genérico para diagnóstico baseado em modelo, e de controlar erros simples e complexos. O procedimento do diagnóstico consiste em três passos principais: detecção do conflito, geração de candidatos e discriminação dos candidatos (veja FIGURA 2.1). O objetivo final do diagnóstico é encontrar um candidato que considere todos os conflitos detectados.

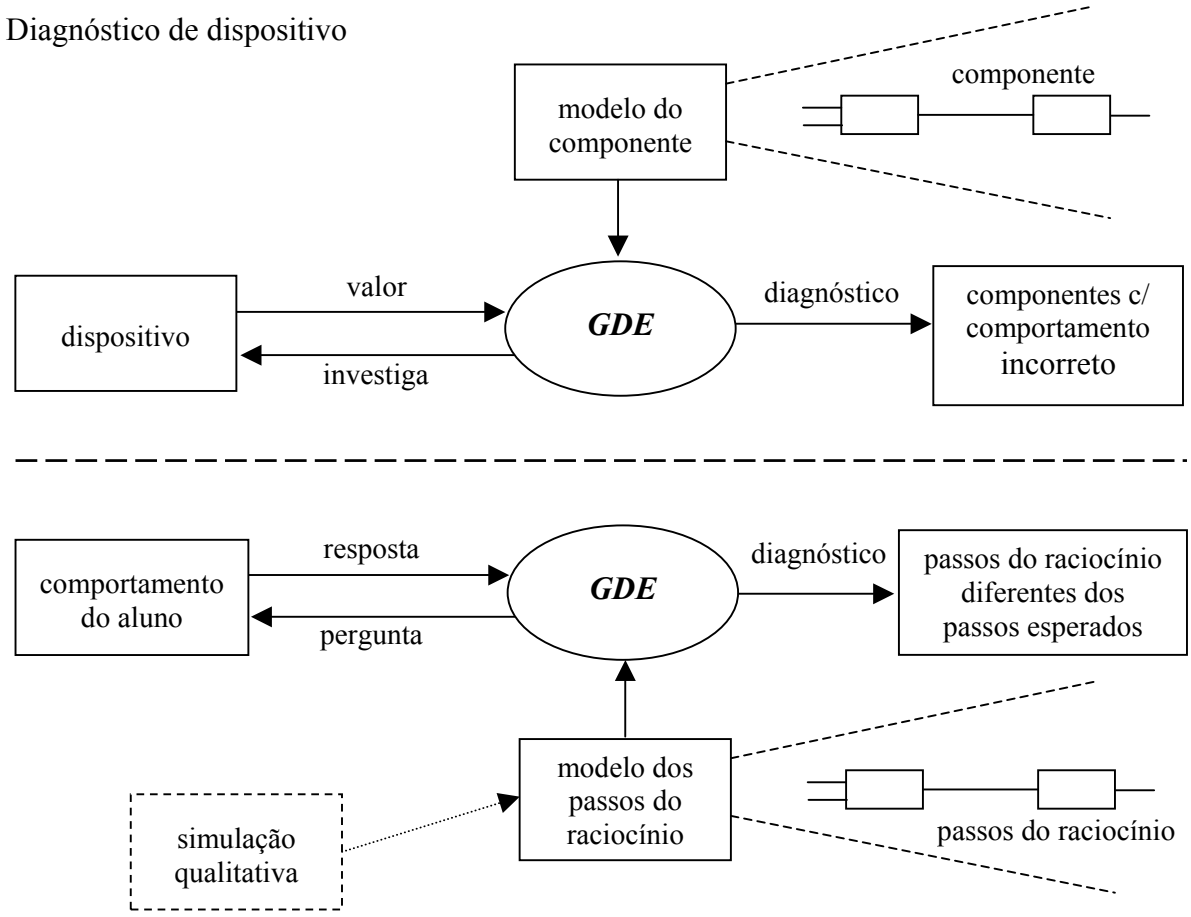


FIGURA 4.1 – Diagnóstico de dispositivo *versus* diagnóstico cognitivo [KON 2000]

Koning [KON 2000] justifica o uso do simulador qualitativo afirmando que simulações computacionais podem ser usadas para construir ambientes interativos, através dos quais as pessoas podem desenvolver conhecimento sobre o comportamento dos sistemas. Apesar de a simulação ocupar uma posição sólida na área de sistemas educacionais, pesquisas demonstram que estas só são efetivas quando uma direção formal é provida. Automatizar funções para prover esta direção requer um modelo de simulação articulado. Este modelo deve manifestar todas as características relevantes do comportamento "real" do sistema, e conter controles apropriados, através dos quais características são indexadas para proporcionar uma comunicação instrutiva entre o aluno e o sistema. Simuladores qualitativos são capazes de prover uma base com esses modelos, portanto, o desafio é a manipulação automatizada da direção do sistema educacional.

Prover direção significa que o ambiente de aprendizagem deve ser capaz de adaptar a interação à situação atual, referente tanto ao assunto específico quanto ao aluno com quem está interagindo [KON 2000]. Esta direção pode assumir muitas formas, como gerar explicações, apresentar exemplos ou sugerir tarefas. Assim, independente da direção escolhida, esta requer conhecimento sobre o aluno e, para isto, é necessário avaliar seu

comportamento. Dessa forma, um sistema de ensino deve diagnosticar o comportamento do aluno na resolução de problemas. O modo como o aluno interage com o ambiente de aprendizagem reflete o comportamento e, portanto, o sistema deve monitorar estas interações e diagnosticar divergências, baseado em erros cometidos pelo aluno. Nos métodos existentes, o propósito do diagnóstico é visto freqüentemente como uma explicação ou indicação das causas dos erros cometidos pelo aluno. Estas causas podem ser erros ou falsas concepções. Existe uma diferença importante entre elas. Erros estão situados em nível comportamental, e falsas concepções em nível conceitual. Portanto, falsas concepções surgem de concepções errôneas sobre o domínio, e erros são comportamentos desencadeados, provavelmente, por falsas concepções.

Um dos motivos pelos quais o diagnóstico do comportamento do aluno na resolução de problemas é considerado extremamente difícil, é a utilização de catálogos de erros ou falsas concepções. É complicado construir catálogos consistentes e, além disso, estes servem apenas para um domínio específico, sendo que quando o domínio muda, um novo catálogo deve ser implementado.

Com um foco voltado à detecção de erros, disposto portanto a diagnosticar a resolução de problemas do aluno em nível comportamental, Koning [KON 2000] define diagnóstico como: *um diagnóstico é (no mínimo) um conjunto de passos do raciocínio que não foi executado pelo aluno conforme o previsto, baseado em algumas observações*. Deste modo, o diagnóstico explica o comportamento do aluno baseado em *como* este comportamento diverge do comportamento esperado, em vez de *por que* isto acontece. A principal desvantagem desta técnica é que, por estar baseada no modelo de passos necessários para a resolução do problema, nenhum conhecimento sobre falsas concepções referentes ao domínio é considerado.

## 4.2 Arquitetura para ambientes de aprendizagem

A FIGURA 4.2 ilustra como um simulador pode ser embutido em um ambiente de aprendizagem interativo, apresentando uma visão abstrata da interação entre o aluno e o simulador, e as funções de aprendizagem e treinamento mais importantes. O "conhecimento genérico qualitativo" refere-se a uma biblioteca de modelos fragmentados, a diferentes conjuntos de regras para determinar mudanças de estado, e a um cenário que pode ser usado para gerar um novo exercício. A saída do simulador qualitativo forma a base para o modelo do domínio. Pela manipulação de objetos da interface, o aluno pode controlar o simulador, respondendo a questões ou pedindo ajuda. Existem dois *loops* de fluxo de controle principais: quando o comportamento do aluno é consistente com o modelo, o *loop* é "interface com o aluno → avaliação do desempenho → seqüência do assunto". Ao contrário, quando uma discrepância é detectada entre o comportamento do aluno e o modelo, o *loop* é "interface com o aluno → avaliação do desempenho → diagnóstico do comportamento → geração da explicação". Este *loop* é executado até que o diagnóstico seja satisfatório.

A parte central, apresentada na FIGURA 4.2, é associada à saída gerada pelo simulador, e serve de entrada para todas as unidades funcionais do sistema, representadas pelas setas cinzas. É importante que estas funções operem independentes do domínio, o que significa que a saída do simulador deve compor-se de duas partes entrelaçadas, porém bem distintas. A primeira parte compõe-se de todos os fatos que juntos representam o comportamento do sistema que está sendo simulado, e a segunda parte implementa um tipo

de visão meta-nível do conteúdo deste modelo, o que consiste num vocabulário qualitativo. Deste modo, as unidades funcionais podem avaliar o modelo usando estes termos, independente do domínio.

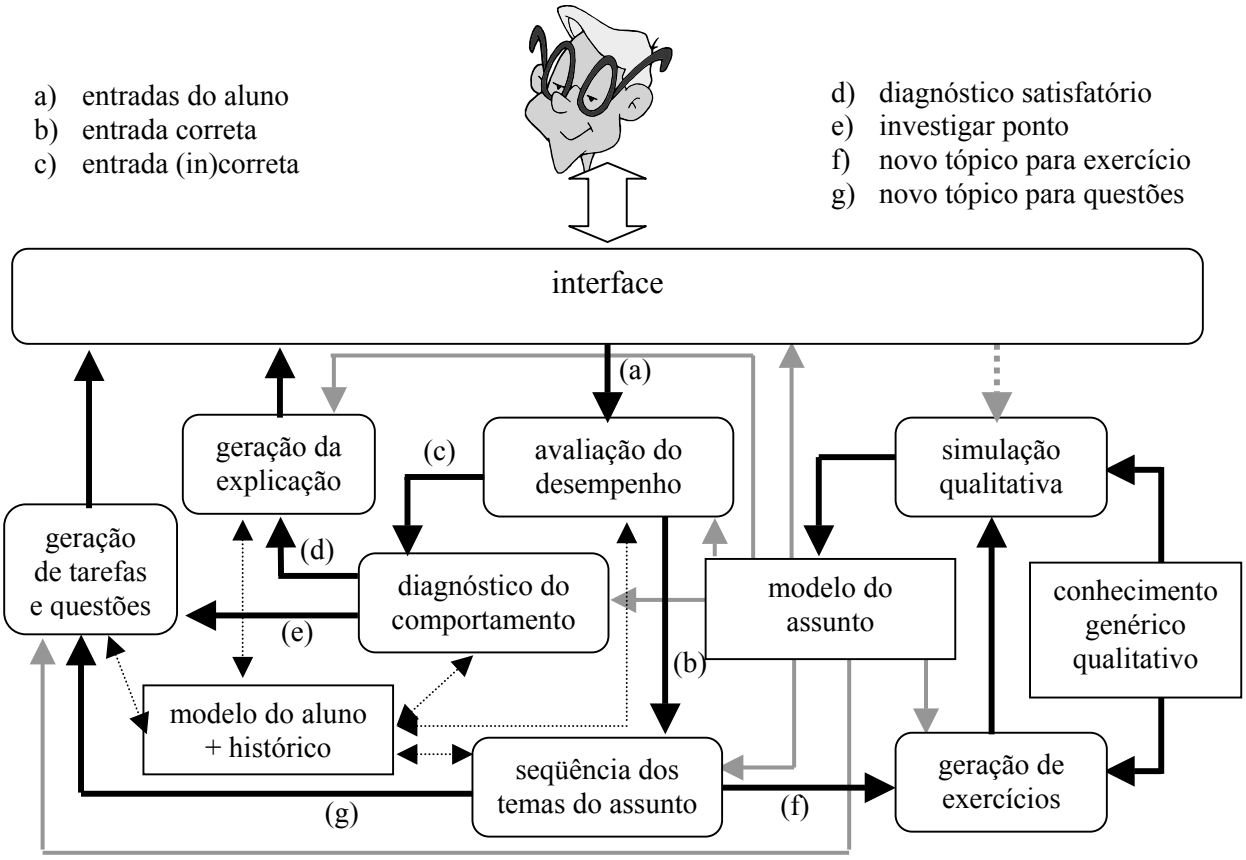


FIGURA 4.2 – Arquitetura conceitual de um sistema educacional [KON 2000]

## 4.3 Diagnóstico cognitivo x manutenção do modelo do aluno

A avaliação do comportamento do aluno refere-se ao diagnóstico cognitivo e, freqüentemente, é considerada semelhante ao modelo do aluno. O modelo do aluno deve ser capaz de representar tanto informações quantitativas (desempenho) quanto qualitativas (fatores e características afetivas do aluno), provendo, assim, possibilidades de uma boa avaliação pedagógica.

Ao contrário da maioria das abordagens existentes, a proposta em [KON 2000] distingue o diagnóstico do comportamento da construção do modelo do aluno, através de uma decomposição funcional da tarefa "modelar aluno", conforme ilustra a FIGURA 4.3. A modelagem do aluno é dividida em duas partes: global e local. A parte global da modelagem do aluno refere-se à construção do modelo do aluno em si. Este modelo contém dados de vários tipos, como o nome do aluno e o número de erros cometidos no último exercício, e pode persistir em muitas seções educacionais. Adicionalmente, o modelo permite incluir informações sobre o que o aluno acredita saber (ou não), incluindo

falsas concepções. A parte local da modelagem do aluno, em princípio, é independente da parte global, e pode ser dividida em três funções principais: monitoração, diagnóstico e reparo, que correspondem às três funções educacionais da arquitetura descrita na seção 4.2.

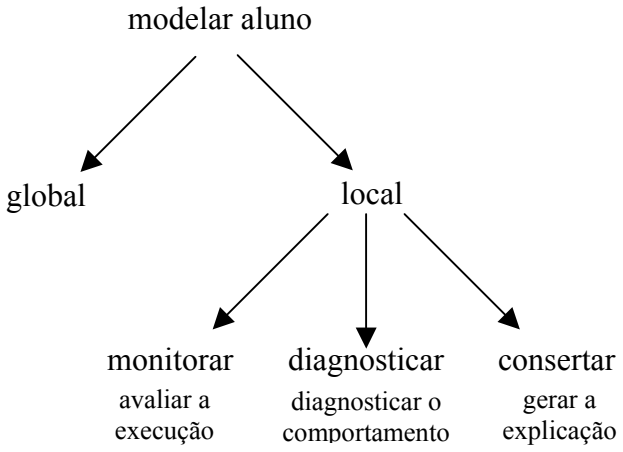


FIGURA 4.3 – Decomposição funcional da tarefa modelar aluno [KON 2000]

A decomposição da tarefa "modelar aluno" é importante justamente pelo fato de que a representação do aluno no sistema (modelo do aluno) ainda é um ponto frágil em ambientes de ensino tradicionais, pois se tem pouco conhecimento sobre os processos da aprendizagem humana, imprecisão e subjetividade dos fatores emotivos e motivacionais envolvidos e, desta forma, é possível gerar um diagnóstico do comportamento do aluno independente de se ter ou não um modelo do aluno perfeito.

# 5 *TÆMS* e *DTC*: independência do domínio

O *framework TÆMS* e o escalonador *Design-to-Criteria* (*DTC*) são ferramentas utilizadas para criar componentes de controle generalizados e independentes do domínio para agentes inteligentes. É importante enfatizar que estas ferramentas são utilizadas na resolução de problemas complexos, em que os agentes estão situados em um ambiente, capazes de perceber esse ambiente e executar tarefas, e têm representações explícitas das tarefas candidatas e de diferentes caminhos a seguir para a execução destas. As tarefas são quantificadas, apresentam características de execução diferentes e relacionamentos entre elas. Isto significa que há um alto grau de interconectividade na resolução de problemas do agente, e decidir o que o agente deve fazer e com quem ele deve se coordenar não é um processo trivial [WAG 2001].

A motivação das pesquisas por independência do domínio está na capacidade de reusar componentes de controle em diferentes aplicações, como no projeto de uma casa inteligente [LES 98], controle de exames de pacientes internados num hospital [DEC 98], controle de processos [ZHA 99], diagnóstico [BAZ 98, HOR 9-a, HOR 2000], entre outros. As seções a seguir descrevem as ferramentas utilizadas para atingir a independência do domínio, e apresentam um resumo de algumas das aplicações citadas acima.

## 5.1 *TÆMS*

O *TÆMS* (*Task Analysis, Environment Modeling, and Simulation*) [DEC 94] é um *framework* modelador de tarefas independente do domínio, usado para descrever e raciocinar sobre processos de resolução de problemas complexos. Desta forma, representa tarefas hierárquicas, planeja ações, atividades candidatas e caminhos alternativos de soluções numa perspectiva quantificada.

Os modelos *TÆMS* são abstrações hierárquicas dos processos de resolução de um problema, que descrevem caminhos alternativos para execução do objetivo planejado, representando as tarefas, as interações entre elas e as limitações de recursos. Um exemplo de uma estrutura de tarefas *TÆMS* é apresentado na FIGURA 5.1, onde as elipses representam as tarefas e os retângulos os métodos. Todas as ações primitivas no *TÆMS*, chamadas métodos, são caracterizadas estatisticamente por distribuições de probabilidade discreta em três dimensões: qualidade, custo e duração. Qualidade é um conceito abstrato que descreve a contribuição de uma ação particular para a resolução do problema como um todo. Duração descreve a quantidade de tempo que a ação modelada pelo método levará para executar. Custo descreve o custo financeiro ou em qualquer outra unidade inerente à execução da ação. Na FIGURA 5.1, por exemplo, *Get-Hot-Water* é um método com distribuição de qualidade de 125, custo de 6 e duração de 4, em 100% dos casos.

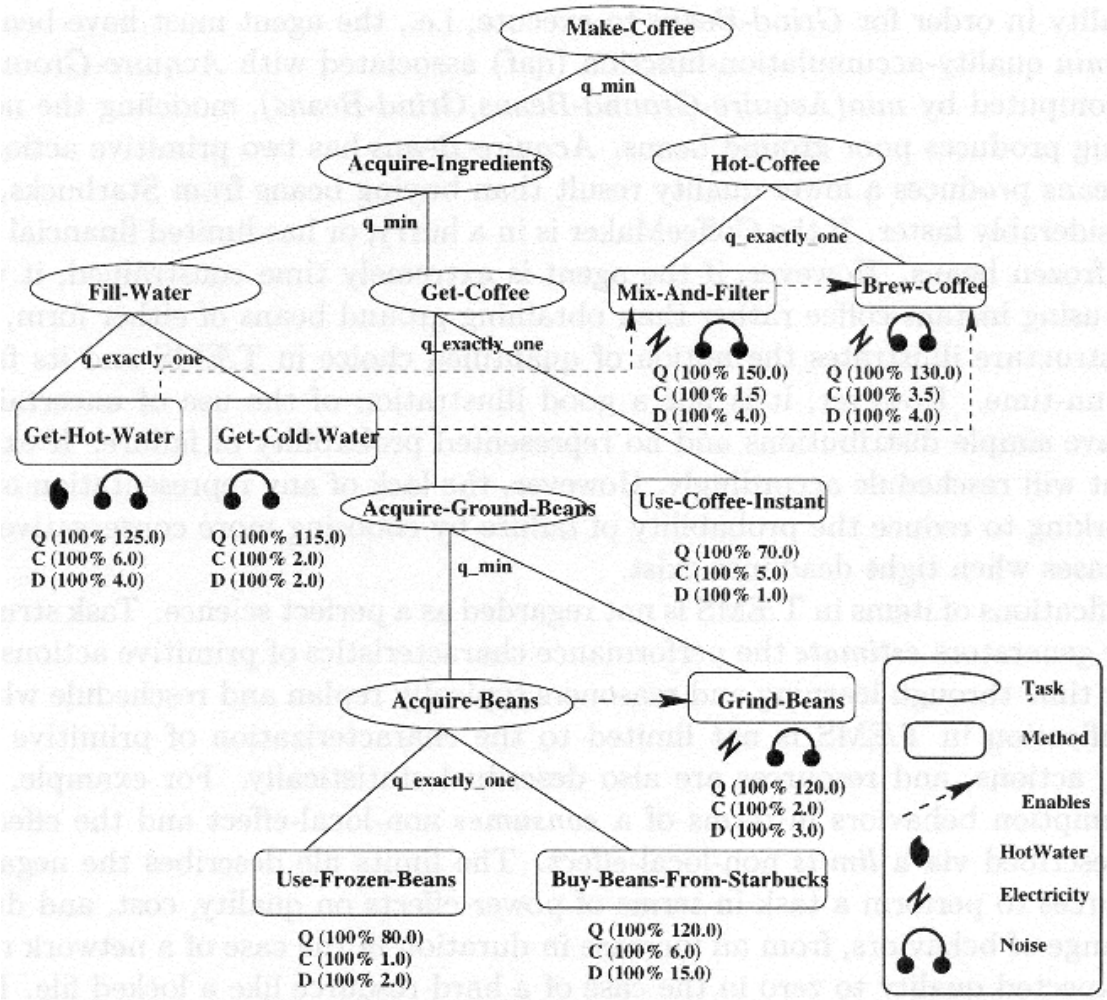


FIGURA 5.1 – Estrutura de tarefas *TÆMS* para preparar café [LES 98]

Um modelo *TÆMS*, cujo propósito principal é analisar, explicar ou predizer a performance de um sistema ou de um componente, tem três níveis: objetivo, subjetivo e generativo [DEC 94]. O nível objetivo descreve o essencial, a estrutura da tarefa "real" na resolução de um problema particular, o que é quase equivalente a uma descrição formal de uma situação de resolução de um único problema, sem informações sobre os agentes em particular. O nível subjetivo descreve como os agentes vêem e interagem com a situação de resolução do problema através do tempo. Este nível é essencial na avaliação dos algoritmos de controle, porque enquanto o comportamento individual e o desempenho do sistema podem ser medidos objetivamente, os agentes devem tomar decisões só com informações subjetivas. Finalmente, o nível generativo descreve as características estatísticas necessárias para gerar as situações objetivas e subjetivas do domínio. Um modelo de nível generativo consiste em uma descrição de processos ou distribuições geradoras, de onde pode ser derivada uma gama de instâncias de problemas alternativos.

### 5.1.1 Nível objetivo

O nível objetivo descreve a estrutura essencial da resolução de um problema em uma determinada situação. O foco está em como os inter-relacionamentos entre as tarefas afetam a qualidade e a duração de cada tarefa. O modelo básico é um grupo de tarefas que fazem parte de um ambiente e induz as tarefas (*T*) a serem executadas pelos agentes. Os grupos de tarefas são independentes uns dos outros, porém existem inter-relacionamentos entre as tarefas dentro de um único grupo. Os grupos ou as tarefas podem ter prazos finais (*D(T)*). A qualidade de execução ou os resultados de cada tarefa influenciam a qualidade do resultado do grupo de tarefas (*Q(T)*) de forma precisa. Essas quantidades podem ser utilizadas para avaliar o desempenho de um sistema.

Uma tarefa individual que não tem subtarefas é chamada método (*M*), e é a menor unidade de trabalho. Uma tarefa pode ser executada por um ou mais métodos, sendo que cada método levará uma determinada quantidade de tempo e produzirá uma determinada qualidade como resultado. A qualidade do desempenho de um agente em uma tarefa individual é uma função de cronometragem e escolha das ações dos agentes (efeitos locais), e a possibilidade de execuções de outras tarefas (efeitos não-locais). O propósito básico do modelo objetivo é especificar formalmente como a execução e a cronometragem das tarefas afetam as medidas de qualidade [DEC 94].

A qualidade da tarefa ou de um grupo de tarefas ($Q(T)$) baseia-se no relacionamento entre subtarefas. Esta função de qualidade é construída recursivamente – cada grupo de tarefas consiste de tarefas, cada uma dessas tarefas constitui-se de subtarefas, etc. – até que tarefas executadas individualmente (métodos) são alcançadas. Formalmente, o relacionamento entre subtarefas é definido como *SUBTASK*(*T,* T, *Q*), em que T é o conjunto de todas as subtarefas de *T* e *Q* é uma função de qualidade $Q(T, t)$ : [*tasks* x *times*] → [*quality*], que retorna a qualidade associada a *T* no tempo *t* [DEC 94]. As semânticas de um ambiente particular são modeladas pela escolha apropriada da função de qualidade (exemplo de funções: mínima, máxima, somatório ou média). Cada método *M* produzirá potencialmente, em um tempo *t*, uma qualidade máxima q(*M*, *t*) depois que uma determinada quantidade de tempo d(*M*, *t*) decorrer. A qualidade da tarefa ou de um grupo de tarefas e a qualidade dos métodos produzem efeitos locais.

Qualquer tarefa *T*, contida em um método que começa a executar antes da execução de um outro método *M* terminar, pode afetar a execução do método *M* através de efeitos não-locais (*e*), o que é descrito por $NLE(T, M, e, p_1, p_2, ...)$, em que os *p* são parâmetros da classe de efeitos. Para esse modelo, existem duas saídas possíveis na aplicação de efeitos não-locais em *M*: efeitos de duração, em que d(*M*, *t*) (duração) é alterada, e efeitos de qualidade, em que q(*M*, *t*) (qualidade máxima) é alterada. Desse modo, uma classe de efeitos *e* é uma função $e(T, M, t, d, q, p_1, p_2, ...)$ : [*task* x *method* x *time* x *duration* x *quality* x *parameter 1* x *parameter 2* x ... ] → [ *duration* x *quality* ] [DEC 94].

A direção de um efeito não-local depende do tempo de execuções dos métodos, da qualidade das tarefas antecedentes ao efeito e da disponibilidade de informação entre os agentes (em modelos multiagentes). Os efeitos não-locais são uma parte importante do *framework TÆMS*, porque provêm muitas das características que fazem um ambiente de tarefas único e diferente de outros [DEC 94]. As classes de efeitos não-locais mais comuns são: habilita, facilita, impede e precede.

Os recursos também podem produzir efeitos não-locais. O *TÆMS* permite que recursos sejam representados diretamente nas estruturas de tarefas e, desta forma, os efeitos dos recursos podem mudar a duração ou a qualidade de outras tarefas, o que é

representado por um conjunto de efeitos não-locais (também chamados *NLE – Non-Local Effect*). Os métodos são relacionados aos recursos por um tipo de NLEs (por exemplo, usa, consome, enche, ...) e outras NLEs executam dos recursos para os métodos (por exemplo, acesso exclusivo, largura de banda limitada, consumível, ...). As NLEs dos métodos para recursos mudam o estado do recurso, e as NLEs dos recursos para os métodos afetam a duração e a qualidade máxima do método.

### 5.1.2 Nível subjetivo

O nível subjetivo de um ambiente descreve quais as partes do nível objetivo de uma determinada situação estarão disponíveis para um agente. Isto responde questões como "quando parte da informação está disponível", "para quem está disponível" e "qual é o custo daquela parte da informação para o agente", – uma descrição de como o agente pode interagir com o seu ambiente – quais as opções que estão disponíveis para ele [DEC 94].

Os agentes possuem crenças, que representam aquilo em que eles acreditam em relação ao ambiente no qual estão inseridos. Essas crenças afetam as ações através de alguns mecanismos de controle. O mecanismo de controle do agente usa o conjunto de crenças atual do agente para atualizar três subconjuntos especiais destas crenças, identificados como conjuntos de informação agrupados, comunicação e método de execução de ações para serem computados e, além disso, um componente de controle deve descrever a duração de suas deliberações [DEC 94].

O modelo suporta computação paralela, e qualquer interação importante entre métodos executando em paralelo é representada por efeitos não-locais. As ações de um agente podem afetar tanto o ambiente em que estão inseridos quanto os outros agentes. O ambiente pode ser afetado através do método de execução (que pode ser um método de execução simples ou um método de execução com monitoramento, suspensão e preempção), afetando a qualidade da execução do método ou as informações disponíveis no conjunto de crenças, e outros agentes podem ser afetados pelas ações de comunicação, afetando suas crenças.

### 5.1.3 Nível generativo

Usando os níveis objetivo e subjetivo do *TÆMS*, é possível modelar qualquer situação individual; e acrescentar o nível generativo ao modelo permite determinar qual o desempenho esperado de um algoritmo durante um longo período de tempo, com muitos episódios de resolução de problemas individuais. Os parâmetros estatísticos do nível generativo também podem ser usados pelos agentes em seus raciocínios subjetivos, como exemplo: um agente pode tomar as decisões de controle baseado no conhecimento da duração esperada dos métodos.

Um modelo do nível generativo pode ser construído através de análises cuidadosas de um ambiente real já modelado, ou através da observação de propriedades estatísticas de episódios reais.

## 5.2 *DTC*

O escalonador *DTC* (*Design-to-Criteria*) é um controle local do agente, utilizado na tomada de decisões. O papel do escalonador é gerar todos os caminhos de ações possíveis para atingir um determinado objetivo e escolher o melhor entre eles, considerando [WAG 2001]:

- os objetivos ou metas locais do agente (suas preferências por determinados tipos de soluções);
- as limitações de recursos locais do agente e as condições do ambiente; e
- as considerações não-locais expressas pelo módulo de coordenação do agente.

A idéia geral é avaliar as opções considerando limitações e preferências de diversas fontes e encontrar o melhor caminho para atingir o objetivo das tarefas selecionadas.

O escalonador de tarefas interpreta a situação e os objetivos do agente através da leitura das estruturas de tarefas *TÆMS*, e cria vários caminhos de ações possíveis. O escalonamento das atividades para a resolução dos problemas modeladas no *TÆMS*, requer quatro itens principais [WAG 2001]:

- encontrar um conjunto de ações para atingir a tarefa de nível mais alto;
- seqüenciar as ações;
- encontrar e seqüenciar as ações em tempo real; e
- gerar um escalonamento das tarefas que satisfaça os objetivos dinâmicos do agente.

A necessidade de trabalhar em tempo real está intimamente ligada ao fato de que o *DTC* é projetado para ambientes abertos, quando é comum ocorrerem mudanças e, neste caso, o agente reinvoca o *DTC* para gerar um caminho de ações apropriado àquela nova situação.

Múltiplas aproximações de modelos *TÆMS* para executar tarefas, junto com as características de qualidade, custo e duração das ações primitivas, dão flexibilidade ao agente na resolução do problema, o que descreve como os agentes podem responder a novas situações e gerar novas soluções para o problema.

O escalonador *DTC* requer métodos heurísticos avançados porque, devido à complexidade computacional inerente às tarefas, não é possível utilizar técnicas de pesquisa exaustiva para encontrar soluções ótimas. Além disso, os *deadlines* e as limitações de recursos das tarefas, mais a existência de inter-relacionamentos complexos, impedem o uso de uma única heurística para produzir soluções ótimas ou até mesmo boas. O *DTC* enfrenta essas combinações explosivas através de aproximações, resolução de problemas direcionada ao objetivo, tomando decisões heurísticas e corrigindo erros de heurísticas.

## 5.3 Exemplos da utilização das ferramentas

Esta seção apresenta um resumo de algumas das aplicações que utilizam as ferramentas descritas nas seções anteriores, utilizadas para criar componentes de controle generalizados e independentes do domínio para agentes inteligentes.

### 5.3.1 O projeto de uma casa inteligente

A aplicação de SMA em ambientes inteligentes é interessante devido à distribuição funcional e espacial do ambiente que se aplica naturalmente ao modelo multiagente. No projeto *UMASS IHome* [LES 98] foi implementado a simulação de um ambiente doméstico sofisticado (FIGURA 5.2), povoado por agentes inteligentes e distribuídos (incluindo a simulação de robôs), que controlam aparelhos eletrodomésticos e negociam os recursos compartilhados. O foco do trabalho é a coordenação dos recursos compartilhados pelos agentes durante a realização de suas tarefas.

Os agentes domésticos executam determinadas tarefas, selecionam ações candidatas baseados na preferência dos ocupantes e na disponibilidade de recursos e, além disso, coordenam os recursos compartilhados como eletricidade, ruído, temperatura e água quente. Tarefas, recursos, habilidades e ações primitivas são representadas e quantificadas em estruturas de tarefas *TÆMS*.

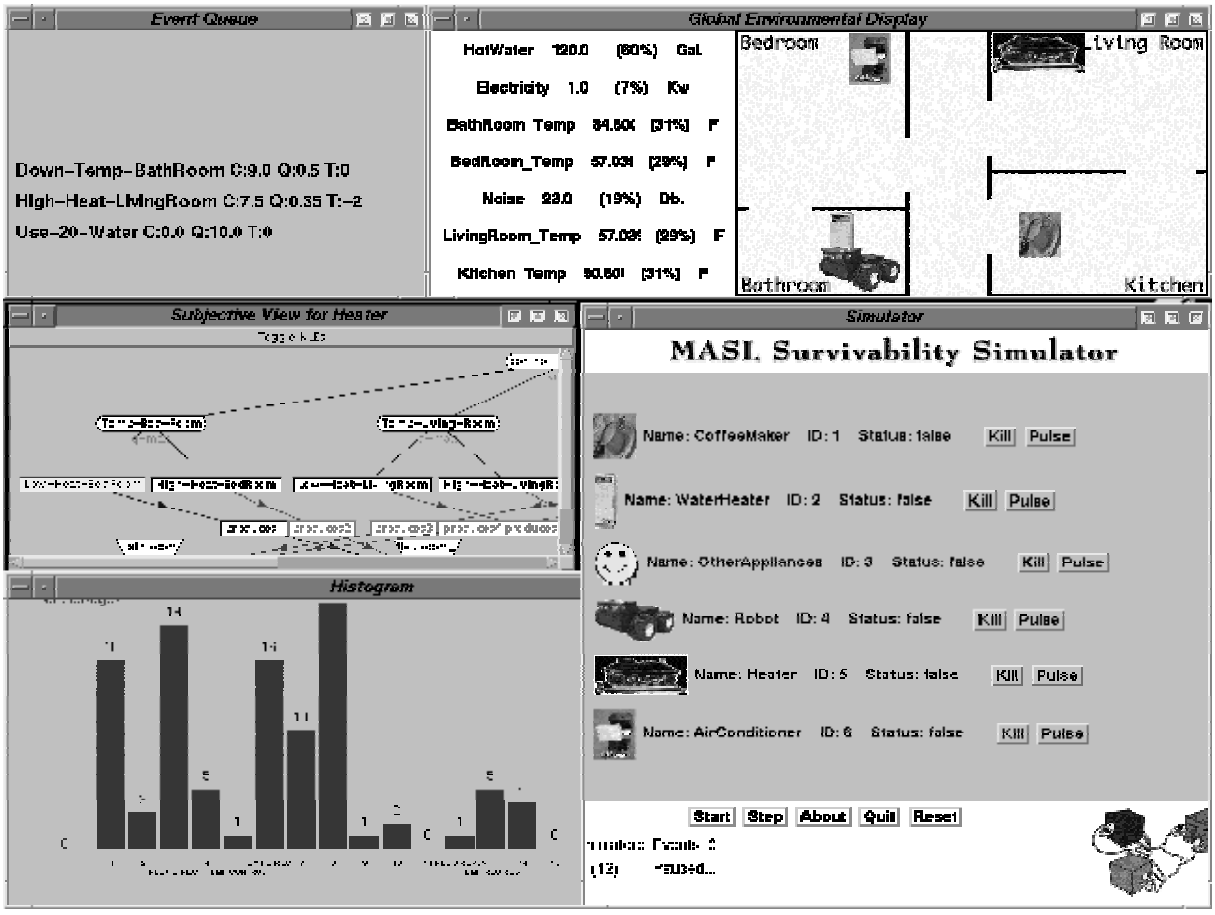


FIGURA 5.2 – Simulação do ambiente *IHome* [LES 98]

A FIGURA 5.1 apresenta a estrutura de tarefas *TÆMS* usada pelo agente *CoffeeMaker* para representar o processo de preparação do café. Na estrutura de tarefas são descritos caminhos alternativos para obter água, obter café e preparar o café. Observe a tarefa *Acquire-Ground-Beans*, que apresenta duas subtarefas: *Acquire-Beans* e *Grind-*

*Beans*. A subtarefa *Acquire-Beans* é decomposta em duas ações primitivas, *Use-Frozen-Beans* e *Buy-Beans-From-Starbucks* que, assim como *Grind-Beans*, são descritas em termos de qualidade, custo e duração. Observe que usar grãos congelados produz um resultado de qualidade inferior ao resultado obtido com a compra de grãos, porém o custo é menor e a execução da tarefa é consideravelmente mais rápida. Portanto, se o agente *CoffeeMaker* estiver com pressa ou com recursos financeiros limitados, ele pode escolher usar grãos congelados. Se o agente, por outro lado, estiver com muita pressa, ainda tem a possibilidade de executar o *Get-Coffee* e usar café instantâneo. Os ícones pequenos debaixo de alguns métodos representam os recursos utilizados. A linha pontilhada que liga *Acquire-Beans* e *Grind-Beans* representa um inter-relacionamento do tipo *enable*, o que indica que a subtarefa *Acquire-Beans* deve ser executada antes da subtarefa *Grind-Beans*. A função *q_min* associada à subtarefa *Acquire-Ground-Beans* indica que sua qualidade é calculada por *mim(Acquire-Beans,Grind-Beans*), ou seja, grãos de má qualidade ou um moedor de má qualidade, irão produzir grãos moídos de má qualidade.

A casa inteligente é uma pequena casa construída e executada usando um ambiente de simulação multiagente genérico, dividida em quatro peças: quarto, sala, cozinha e banheiro, unidas por um corredor.

A população da casa inteligente inclui um robô móvel e agentes eletrodomésticos como o *Dryer*, *TV*, *DishWasher*, *WaterHeater*, *VacuumCleaner*, *Heater*, *AirConditioner*, *CoffeeMaker* e o agente *OtherAppliances*. O agente *OtherAppliances* é um lugar reservado para outros eletrodomésticos, ainda não modelados pelos agentes. Este agente faz requisição de recursos e exercita o sistema como se *x* agentes tivessem sido adicionados. A comunicação entre os agentes é feita através de *KQML* (*Knowledge Query and Manipulation Language*).

Alguns experimentos foram realizados, conforme relatado a seguir. Nos experimentos o *IHome* é povoado por sete agentes: *DishWasher*, *Robot*, *WaterHeater*, *CoffeeMaker*, *Heater*, *AirConditioner* e *OtherAppliances* (que simula a presença de múltiplos agentes). A comunicação foi monitorada em cada experimento, assim como o consumo de recursos e o comportamento dos agentes. O ambiente é constante em todas as execuções (largura de banda, performance de execução das ações, etc.), enquanto a disponibilidade dos recursos é variada. A temperatura, em todas as peças da casa, é 76ºF. Os agentes relacionados à temperatura (*Heater, AirConditioner*) são reativos e têm a tarefa de manter a temperatura inicial escolhida pelos ocupantes. Da mesma forma, o agente *WaterHeater* trabalha para manter o nível de água quente entre o mínimo definido e o máximo permitido.

O objetivo dos experimentos é que os agentes cumpram suas tarefas no tempo especificado. Para os agentes reativos, como o agente *AirConditioner*, o objetivo é satisfazer a limitação inicial expressa, como manter a temperatura em 76ºF e manter a água do tanque acima do mínimo definido.

No primeiro experimento, os recursos são configurados como segue: 15 Kw de eletricidade; inicialmente, 140 galões de água quente residem no tanque de água e a capacidade máxima do tanque é de 200 galões; o nível máximo de ruído permitido em qualquer momento é 120 Db; e a temperatura no quarto e na sala é 50ºF; no banheiro e na cozinha é 90ºF.

A FIGURA 5.3 apresenta os resultados da primeira execução. Neste experimento, observa-se que os agentes que requerem múltiplos recursos e possuem seqüências de ações longas para realizarem suas tarefas, como o *CoffeeMaker* e o *DishWasher*, executam suas

tarefas "pobremente", comparado ao seu desempenho independente. Como os agentes requerem múltiplos recursos ao mesmo tempo, estes ficam limitados e, geralmente, os agentes são incapazes de obter os recursos necessários para a execução de suas tarefas.

| Agent | Nb of Tasks | | Final Quality | | Resources Mes. | | Conflict | | | | Tasks Dropped | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Alone | IHome | Alone | IHome | Alone | IHome | Alone | IHome | | | Alone | IHome |
| | | | | | | | | E | R | N | | |
| Dishwasher | 4 | **2** | 135 | **76** | 10 | 10 | 0 | 21 | 1 | 5 | 0 | **2** |
| Robot | 5 | 5 | 10 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| WaterHeater | | 83 | | 10 | | | | 26 | 3 | 0 | 0 | 0 |
| OtherAppliances | 37 | 33 | 10 | 10 | 42 | 36 | 0 | 8 | 15 | 2 | 0 | 4 |
| CoffeeMaker | 4 | **0** | 80 | **0** | 3 | 3 | 0 | 1 | 11 | 24 | 0 | **4** |
| | 4 | 4 | 125 | 125 | 3 | 3 | 0 | | | | 0 | 0 |
| | 4 | 4 | 125 | 125 | 3 | 3 | 0 | | | | 0 | 0 |
| Heater | 8(41) | 7 (77) | 40 | 40 | 8 | 7 | 4 | 8 | 3 | 0 | 0 | 1 |
| AirConditioner | 8 (41) | 12 (77) | 35 | 20 | 16 | 24 | 4 | 12 | 4 | 0 | 0 | 0 |

FIGURA 5.3 – Resultados do primeiro experimento [LES 98]

Ao contrário, os agentes mais reativos e com planejamento longo (*WaterHeater*, *Heater*, *AirConditioner*) apresentam um resultado melhor. A única diferença entre as execuções individuais e as execuções em grupos é que nos casos mais longos eles demoram muito para atingir os resultados esperados. O *Heater* e o *AirConditioner* trabalham até o clique 77 para atingir a meta de temperatura de 76ºF, em contraste com 41 cliques necessários no caso individual.

O comportamento dos agentes *CoffeeMake* e *DishWasher* indicam um problema com o protocolo *SHARP* (protocolo de coordenação de recursos). Estas tarefas são sempre interrompidas por outras tarefas prioritárias. As prioridades das tarefas não são aumentadas quando elas são interrompidas e, desta forma, elas não se tornam menos interruptíveis com o passar do tempo ou quando estão mais próximas do prazo final. O problema também origina-se da implementação da preferência pessoal – prioridades dos agentes nestes experimentos nem sempre refletem as preferências pessoais dos ocupantes e, desta maneira, a noção da função da utilidade global (até mesmo uma visão local) é um tanto confusa.

O segundo experimento é idêntico ao primeiro, com exceção da tarefa *CoffeMake*, que é designada com prioridade mais alta no sistema, habilitando-a a obter os recursos necessários para realizar suas tarefas. Entretanto, com essas configurações, as tarefas que controlam a temperatura levam até 90 cliques para atingir suas temperaturas alvo.

No terceiro experimento, os recursos são configurados similarmente, com exceção da capacidade máxima do tanque de água, que é reduzida para 60 galões, e o fato de ele estar vazio no início do experimento. Com isso, a negociação sobre o recurso água quente fica mais complicada. Nesse caso, o *DishWasher* só consegue realizar uma dentre as quatro tarefas que deveria executar. Este agente enviou 84 mensagens solicitando os recursos necessários para que ele pudesse realizar suas tarefas (ele é cancelado pelo *WaterHeater*). Com a limitação de água quente, os agentes *AirConditioner* e *Heater* falharam, ou seja,

não conseguiram atingir suas temperaturas alvo. Esse resultado se deve ao comportamento do agente *DishWasher* que requisita e reserva eletricidade e, dessa forma, interfere no comportamento dos agentes controladores de temperatura. Quando o agente *DishWasher* recebe a quantidade desejada de água quente, ele libera a reserva de eletricidade, mas esse comportamento confunde os agentes controladores de temperatura, resultando em desempenhos inferiores.

### 5.3.2 Diagnóstico e adaptabilidade em SMA

Baseado no trabalho de Horling e seus colegas [HOR 2000], descrito a seguir, conclui-se que a adaptação é importante em qualquer sistema computacional, e essencial em sistemas multiagentes, em que se tem um ambiente variável, capaz de mudar rapidamente de um momento para o outro, por causa de erros ou interações inesperadas. Fica claro que um componente de diagnóstico robusto e flexível, aliado a modelos informativos e dados, é parte integrante da adaptabilidade de um sistema multiagente, tornando os sistemas mais eficazes.

Agentes que trabalham sob condições do mundo real enfrentam um ambiente capaz de mudar rapidamente de um momento para outro, devido a erros, interações inesperadas ou intrusões. Para controlar tais situações de maneira natural e eficiente, os agentes devem ser capazes de se adaptarem, ou por estruturas internas, ou consertando as crenças influenciadas por estruturas externas, que provocaram seu mau funcionamento no sistema. O primeiro passo para atingir adaptabilidade é o diagnóstico, ou seja, o sistema deve ser capaz de descobrir e determinar com precisão a causa de uma falha.

Projetar sistemas utilizando paradigma multiagente oferece várias vantagens, como a utilização de recursos distribuídos e, conseqüentemente, a execução de múltiplos objetivos em paralelo e a redução do risco de um único ponto de falha. Em ambientes dinâmicos, algumas suposições que os projetistas normalmente fazem durante a construção dos processos de interações podem ser incorretas. O projetista, por exemplo, pode assumir uma certa taxa de transferência na rede local, ou um método particular pode produzir resultados de uma determinada qualidade, porém, ambos podem facilmente mudar com o tempo. O fracasso da ação de suposições incorretas pode levar o sistema a um desempenho degradado, resultados incorretos, ou, no pior dos casos, a travá-lo. Uma solução seria o projetista ser extremamente abrangente e tentar produzir planos de precaução para todos os aspectos do sistema, como, por exemplo, executar planos de coordenação redundantes, a fim de reduzir as chances de falhas. O problema é que isto aumenta o *overhead* do sistema a níveis indesejáveis e reduz a eficiência global, pois o sistema desperdiça esforços evitando falhas improváveis.

Sistemas multiagentes deveriam ser projetados com a questão da adaptabilidade em mente, a fim de criar sistemas mais robustos sem sacrificar a eficiência. Um agente que trabalha em condições mutáveis deve conhecer o ambiente, objetivos e comportamento esperados, e ser capaz de gerar novos planos de coordenação em situações em que suas expectativas não são satisfeitas. Para iniciar o comportamento adaptativo, é necessário detectar cada expectativa que falhou e, determinadas as causas dessas falhas, corrigi-las. Esta condição pode ser satisfeita e a adaptabilidade promovida, incorporando-se à capacidade de diagnóstico independente do domínio e à coordenação nos agentes. A importância do diagnóstico na adaptabilidade de um sistema multiagente é discutida no trabalho de Horling e colegas [HOR 2000], que relata as informações necessárias para

produzir o diagnóstico e, em particular, a coordenação dirigida por um modelo causal, exemplificando o processo no contexto do ambiente *IHome* (descrito na seção anterior).

No ambiente *IHome*, os eletrodomésticos são controlados por agentes autônomos. Esse ambiente proporciona condições de funcionamento interessantes, permitindo a criação de cenários envolvendo recursos compartilhados, objetivos contraditórios e interações cooperativas. Para avaliar o papel do diagnóstico em sistemas multiagentes, considere o cenário descrito a seguir. Em uma casa, tem-se uma lavadora de louças e um aquecedor de água, relacionados pelo fato de que a lavadora de louças usa a água quente produzida pelo aquecedor de água. Em circunstâncias normais, a lavadora de louças assume que tem água suficiente disponível para que ela possa operar, e o aquecedor irá tentar manter sempre consistente o nível de água no tanque. Devido a esta suposição e ao desejo de reduzir atividades desnecessárias na rede, normalmente é desnecessária a coordenação entre os dois agentes. Se, por exemplo, essa suposição não for válida, quando o dono da casa decide tomar uma ducha num momento conflitante (há uma suposição de que chuveiros só são utilizados pela manhã), ou, se o aquecedor de água está no modo de conservação de energia e só produz água quente quando requisitado, a lavadora de louças não terá recursos suficientes para executar sua tarefa. Se o sistema não tem diagnóstico ou monitoramento, a lavadora de louças poderia continuar, mas faria um serviço "pobre", ou então, pararia por quantidade insuficiente de água quente. Utilizando diagnóstico, porém, a lavadora de louças poderia, diante do mau funcionamento observado por sensores internos ou pelo retorno do usuário, determinar que um recurso está faltando e não está sendo objeto de coordenação. Isto poderia ser o suficiente para produzir um diagnóstico preliminar, e a lavadora de louças simplesmente invocaria um protocolo de coordenação de recursos. Depois de revisar suas suposições, experiências e interações com o aquecedor de água, a lavadora de louças poderia validar e refinar o diagnóstico e, talvez, atualizar suas suposições de modo que a água quente fosse um recurso coordenado, ou que há determinados períodos durante o dia em que a coordenação é recomendada. Esse exemplo deixa claro que até mesmo diagnósticos simples podem prover uma providência de robustez justa para algumas circunstâncias.

Para o diagnóstico funcionar, é necessário que o comportamento esperado seja conhecido *a priori*, servindo de base para comparação. A informação exata necessária depende das capacidades de diagnóstico do agente, que geralmente se encaixam em uma das três categorias: conhecimento sobre o comportamento operacional esperado do agente, incluindo suposições ambientais; métodos para detectar divergências destas expectativas, e formas de diagnosticar estas divergências quando elas são encontradas.

Horling e seus colegas [HOR 2000], propõem que as informações sejam codificadas de forma independente do domínio utilizando, para isso, o *TÆMS* (descrito na seção 5.1). Como as estruturas de tarefas *TÆMS* são uma representação fiel do comportamento esperado do agente local e suas interações com outros agentes e recursos, e as informações providas foram usadas no atual contexto de coordenação, o diagnóstico pode ser realizado de forma independente de domínio.

Informações descrevendo suposições externas pertinentes também são necessárias para que o sistema de diagnóstico possa argumentar sobre as interações do agente com seu ambiente, como informações sobre a rede computacional (largura da banda, latência), características dos recursos (ligado ou não, padrões de uso). O conhecimento organizacional, que é a informação que o agente tem sobre onde e como ele se ajusta a sua sociedade, é um subconjunto desta categoria.

Quando informações descritivas e modelos são incorporados ao agente, o processo de utilização de informações para detectar possíveis inconsistências torna-se relevante. Considere o caso simples de detecção de resultados de métodos *TÆMS* anormais. Na estrutura *TÆMS*, são descritos os resultados do custo, qualidade e tempo esperados para cada método. Com essas informações, o sistema de diagnóstico pode usar um monitor de comparação para determinar quando ocorreu uma divergência de comportamento. Além disso, podem ser usados também suposições sobre limiares de desempenho para determinar a severidade da divergência, e modelos de aprendizagem *on-line* para rastrear tendências de comportamento, determinando se a divergência é causada por uma mudança fundamental no ambiente ou é apenas temporária.

Utilizando conhecimento sobre os métodos de interação (também baseado no *TÆMS*), o sistema de diagnóstico pode determinar se a falha foi provocada por algum método que havia executado com sucesso ou não. Se a origem do método for local, um *log* de atividades pode ser checado para buscar mais evidências. Porém, se a origem do método for remota, então a lista de agentes externos conhecidos deve ser utilizada para localizar o responsável. Uma boa base para detectar possíveis falhas é a comparação baseada em modelos combinada com uma coleção de evidências dirigidas [HOR 2000].

### *5.3.2.1 Executando o diagnóstico*

O agente pode basear-se em suas expectativas e seus métodos de detecção de falhas para iniciar o processo de diagnóstico, que é organizado através de um modelo causal. Esse é baseado em um conjunto de nodos de diagnóstico organizados em grafos acíclicos, em que cada nodo corresponde a um diagnóstico particular, com níveis variáveis de precisão e complexidade. No sistema de diagnóstico, o modelo causal age como um mapa, permitindo progredir de diagnósticos de sintomas facilmente detectáveis, para hipóteses de diagnósticos mais precisos, conforme for necessário.

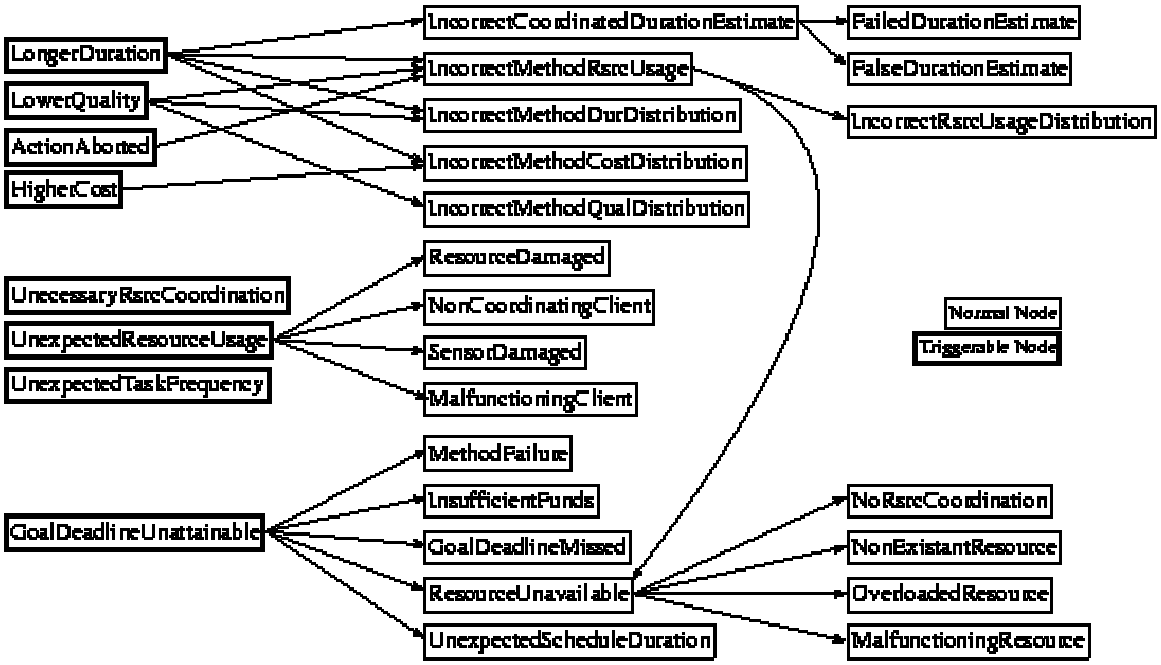


FIGURA 5.4 – Estrutura do modelo causal do projeto *IHome* [HOR 2000]

O modelo causal apresentado na FIGURA 5.4 foca as questões de coordenação, comportamento e recursos. Os nodos que aparecem em negrito no diagrama executam periodicamente uma comparação de checagem simples, para determinar se uma falha pode ter ocorrido. Essa atividade de checagem é um mecanismo primário para iniciar o processo de diagnóstico. O tempo de execução pode ser apresentado considerando a atividade de diagnóstico do agente, no exemplo da lavadora de louças, que executou uma atividade atingindo uma qualidade mais baixa que o esperado. Utilizando o modelo causal determinado, o nodo *LowerQuality* (baixa qualidade da tarefa) seria ativado e o diagnóstico resultante ativaria os nodos filhos *IncorrectMethodRsrcUsage* (método incorreto do uso de recursos), *IncorrectMethodDurDistribuition* (método incorreto da distribuição do tempo) e *IncorrectMethodQualDistribuition* (método incorreto da distribuição da qualidade). O primeiro método tenta encapsular a idéia de que algo deu errado com os recursos que se esperava fossem utilizados por ele, enquanto os outros dois tratam possíveis discrepâncias no tempo e na qualidade. *IncorrectMethodRsrcUsage* produziria um diagnóstico ativando *IncorrectRsrcUsageDistribuition* e *Resource Unavailable* (recurso indisponível). Como o recurso não fora usado, então *ResourceUnavailable* seria ativado e, conseqüentemente, os seus quatro nodos filhos. Desses, *NoRsrcCoordination* (não há coordenação sobre o recurso), possivelmente combinado com *OverloadedResource* (recurso sobrecarregado) determinariam o problema exato.

O modelo causal direciona os objetivos previamente projetados. O fluxo automático do diagnóstico, através da estrutura de grafo, permite ao projetista adicionar ou remover sub grafos e nodos à vontade, aumentando ou diminuindo as particularidades do diagnóstico oferecidas no modelo. Assim, embora o modelo apresentado no trabalho de [HOR 2000] seja voltado para um conjunto específico de falhas, em geral, o modelo causal permite criar sistemas de diagnóstico para qualquer tipo de falhas em um conjunto de sintomas, ou seja, independente do domínio, que podem ser observados ou deduzidos, sendo diferentes uns dos outros.

Falhas envolvendo múltiplos sintomas podem ser elegantemente controladas pelo modelo causal, incorporando um nodo único para a falha que usa evidências de diagnóstico de vários outros nodos. Uma falha detectada pode ser causada por efeitos cumulativos de várias outras divergências, que não foram diagnosticadas individualmente. O projetista também é livre para criar nodos independentes do domínio, sendo possível transportar o modelo de um sistema para outro, com pequenas modificações. Além disso, Horling [HOR 2000] acredita que a estrutura descrita anteriormente pode ser independente de coordenação, sendo formados diagnósticos precisos sobre atividades de coordenação, independentes de detalhes do protocolo.

#### *5.3.2.2 Exemplo da geração de diagnóstico*

Alguns cenários do ambiente *IHome* e a importância da adaptabilidade para um diagnóstico com sucesso relatados em [HOR 2000], serão descritos nesta seção.

*a) Detectando uma parada do sistema*

A FIGURA 5.5 apresenta um cenário com três agentes conectados através de tubos de água, cada um com suas próprias capacidades, objetivos e conhecimentos. A imagem central da figura é o aquecedor de água, que produz água quente até uma taxa estabelecida

pelos pedidos dos agentes que ele abastece. O proprietário da casa determinou que o aquecedor de água produzisse a quantia mínima possível de água quente, minimizando custos. Por essa razão, o aquecedor não manterá água no tanque, obrigando os agentes que precisarem de água quente coordenarem o seu uso. Em outro lugar da casa, uma outra aplicação com objetivo genérico que precisa ser atingido antes de um determinado momento, está usando água quente e funcionando normalmente, coordenando sua necessidade com o aquecedor de água. O terceiro agente, uma lavadora de louças, tem como objetivo lavar os pratos que estão atualmente em sua posse. Essa operação iniciou com sucesso, mas por algum tipo de mau funcionamento, chegou a um estado em que está executando eternamente o método "pré-enxágüe-quente", que utiliza água quente. Seu componente de coordenação está controlando a água quente necessária para o ciclo de execução. O que acontece é que a lavadora de louças tem um nível de prioridade maior que a outra aplicação genérica, dominando o aquecedor de água e forçando-o a rejeitar pedidos de outras aplicações.

Sem diagnóstico, esse cenário terminaria em fracasso. A lavadora de louças nunca completaria seu trabalho, o aquecedor de água produziria muito mais água que normalmente seria necessário, e a outra aplicação não chegaria ao final do seu objetivo. Cada agente, porém, tem informações e um modelo causal semelhante ao apresentado na FIGURA 5.4. As características de cada agente, relevantes nessa situação, podem ser observadas na FIGURA 5.5.

A lavadora de louças detecta o problema e busca evidências da situação atual, produz um diagnóstico com alto grau de confiança através do *log* das atividades locais, dos programas e das suposições sobre a freqüência esperada das tarefas. Esse diagnóstico é enviado para o aquecedor de água, que pode reduzir a prioridade do pedido da lavadora de louças ou cortar o fluxo de água nas duas linhas. A outra aplicação, desavisada de problemas que são controlados pelo aquecedor de água, só sabe que suas tentativas de coordenação têm sido constantemente recusadas. Então, examina os diagnósticos do aquecedor de água, descobre que o recurso de água quente está sobrecarregado e, por isso, deve reprogramar-se.

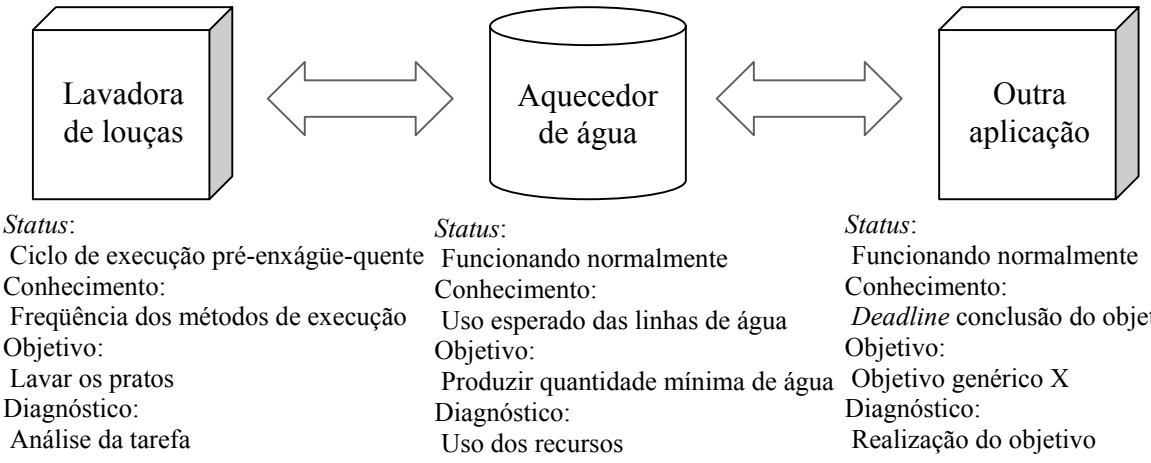


FIGURA 5.5 – Visão de um cenário de falha [HOR 2000]

*b) Coordenação*

Coordenação é um cenário interessante. Existem vários modelos de coordenação possíveis de serem usados em sistemas multiagentes, variando de formas completamente explícitas – comunicação verbal de suposições bem conhecidas – a acordos implícitos.

No exemplo descrito em [HOR 2000], o nodo *UnecessaryRsrcCoordination* (coordenação de recurso desnecessária) inicia seu trabalho monitorando a coordenação dos recursos do sistema. Se ele detecta que o pedido de um recurso particular vem sendo satisfeito sempre, cria uma hipótese de que não é necessário coordenação sobre aquele recurso, porque ele está sendo completado automaticamente. O componente de resolução de problema do agente deve reagir a este diagnóstico, interrompendo a coordenação sobre tal recurso. Com o passar do tempo, o objeto de diagnóstico persistente monitora o recurso, observa se os métodos solicitados estão sendo atendidos, e ajusta o diagnóstico conforme for necessário. Esse diagnóstico provê o *feedback* necessário para o agente manter suposições de situações específicas que sejam eficientes em seu ambiente. Um processo de diagnóstico mais complicado poderia classificar relações de coordenação baseado em tipos de coordenação, ciclos temporais ou sensitividade da quantidade de recursos perdidos.

*c) Discrepância nos resultados do método*

Um aspecto fundamental da adaptabilidade é como o agente responde a resultados inesperados. Se um método falha, o sistema replanejará cegamente ou procurará as razões que provocaram o fracasso? Como o agente reage quando o desempenho varia dentro do que é considerado normal?

Um problema desse tipo é demonstrado no primeiro exemplo, em que o diagnóstico *NoRscrCoordination* foi usado para marcar a falha percebida, permitindo ao agente consertar seu programa original em vez de gerar um novo, podendo cometer o mesmo erro. Um cenário mais interessante envolve uma reação do agente em diferentes níveis de discrepâncias – quando um agente deveria adaptar, ignorar ou consertar o problema? O *TÆMS* permite ao projetista explicitar o comportamento esperado do método, e algoritmos de aprendizagem podem ser empregados para manter a estrutura com o passar do tempo. O conhecimento organizacional mais detalhado sobre o comportamento do método pode ser usado para determinar limiares, permitindo ao agente discriminar entre variações aceitáveis e não aceitáveis, em divergências de desempenho a longo prazo. O sistema de diagnóstico pode usar essas informações e outras evidências disponíveis para prover a descrição do problema específico, quando a solução do problema pode ser usada para escolher a ação apropriada.

Em um processo envolvendo a duração do método, por exemplo, um agente executa o método *X*, esperando que leve entre 10 e 15 cliques para terminar (um clique é uma unidade arbitrária de tempo), como codificado em sua estrutura de tarefa. Além disso, o projetista especificou em seu conhecimento organizacional, que durações de até 40 cliques são tolerâncias aceitáveis, tendo sido projetada considerando coisas como perturbações de atividades da rede ou ruídos nos sensores dos dados. Se *X* fosse terminado em 25 ou 35 cliques, o agente poderia notificar o evento e modificar as características de desempenho no *TÆMS*, depois de determinar que a divergência não foi causada por falta de expectativas inexatas (por exemplo, falta de recursos). Se, em vez disso, *X* levar 100 cliques para ser terminado, está fora da faixa de valores esperados e, também, muito acima

das tolerâncias aceitáveis, portanto, o agente não deveria adaptar suas expectativas a esta nova situação. Essa falha iniciaria a atividade de diagnóstico que irá monitorar o futuro comportamento desse método. Depois de um tempo, como *X* é executado novamente, um quadro mais claro do seu desempenho atual poderia ser gerado, determinando se as falhas foram um evento único, ou uma falha de *software* ou de *hardware*.

#### *5.3.2.3 Detecção do diagnóstico*

Assim como diagnosticar falhas em sistemas multiagentes é um problema interessante, também é importante examinar o efeito das descobertas e a freqüência do diagnóstico do comportamento de sistemas globais. Se o processo de diagnóstico for muito sensível, podem-se desperdiçar esforços monitorando comportamentos que operam normalmente, ou adaptando falhas que não existem. Por outro lado, um sistema de diagnóstico mais cético pode ignorar alguns pontos, criando problemas maiores.

No exemplo descrito na seção 5.4.3.2, aplicações que utilizam água e o aquecedor de água, interagem. O aquecedor de água tenta manter um nível de água quente pré-estabelecido no tanque ou coordena o uso de água quente, assegurando que a água esteja disponível quando solicitada. O desafio é determinar quando a água deveria ser coordenada: o agente deveria gastar energia para assegurar que a água necessária esteja disponível, ou deveria usar a água e aceitar o fato de que há pouca probabilidade de fracasso devido à falta de recursos?

Para diagnosticar a situação, a aplicação que utiliza água examina o padrão da coordenação com o aquecedor de água e os efeitos dos resultados quando a coordenação termina. Assim, se o agente detecta que as respostas a seus pedidos são quase sempre positivas, pode optar por deixar de coordenar. Porém, resultados de execução negativos farão o agente começar a coordenar novamente.

Dois parâmetros variam em diferentes experiências: a freqüência na qual a coordenação e os resultados são examinados, e a quantidade necessária de evidências para um diagnóstico ser produzido. O cenário permite executar, durante uma quantidade fixa de tempo, depois que o número de mensagens de coordenação, a qualidade global, o número de métodos de execução e o número de mudanças de coordenação forem registrados. Um agente ótimo minimizaria o número de tentativas e maximizaria a qualidade. O número de mudanças e execuções de coordenação não é bom ou ruim para si próprio, mas é indicativo de taxas e resultados de adaptação que o agente exibiu.

# 6 Sistema de diagnóstico proposto

Para propor o sistema de diagnóstico cognitivo baseado em modelo e independente do domínio, descrito neste capítulo, buscou-se embasamento teórico nos trabalhos descritos nos capítulos 3 e 4. No capítulo 3 foram apresentados *frameworks* gerais para diagnóstico que, na realidade, são interessantes para diagnóstico de dispositivos físicos como um circuito lógico. Considerando que em sistemas de diagnóstico cognitivo é mais difícil identificar o "conhecimento defeituoso", Self [SEL 93] propõe algumas extensões aos *frameworks* gerais, como a utilização de modelos. Mesmo assim, alguns dos principais problemas dos métodos gerais de diagnóstico permanecem; entre eles, o fato de o diagnóstico cognitivo ser interativo, pois o objeto a ser diagnosticado é um agente ativo no processo de diagnose, o que é imensamente complicado de tratar. Porém, como o diagnóstico cognitivo baseado em modelo não tem uma metodologia própria e segue os princípios básicos dos métodos gerais de diagnóstico, o desafio é ser independente do domínio e, conseqüentemente, desenvolver descrições cognitivas válidas. O trabalho de Koning [KON 2000], descrito no capítulo 4, apresenta um sistema de diagnóstico cognitivo baseado em modelos qualitativos, gerados a partir do simulador *GDE* (descrito na seção 3.2).

No diagnóstico cognitivo baseado em modelo, a idéia básica é a detecção de erros, como a discrepância entre o comportamento real do aluno e o comportamento especificado no modelo. Conseqüentemente, o diagnóstico baseado em modelo não requer catálogos de erros explícitos, mas a representação do comportamento esperado do aluno. Apesar das vantagens do uso de sistemas de diagnóstico baseado em modelo em ambientes de ensino, como a generalidade, reusabilidade e transferência, o gargalo principal, segundo Koning [KON 2000], é o fato de os modelos dos "comportamentos esperados de resolução de problemas", não estarem prontamente disponíveis. Como tal, o custo da construção de catálogos de erros para domínios específicos ou o desenvolvimento de uma teoria generativa de erros pode parecer estar sendo substituído pelo custo da construção de modelos de diagnóstico para um domínio específico. Koning [KON 2000] resolveu esse problema com a geração automática de modelos de diagnóstico, através do uso do simulador qualitativo *GDE*.

Dessa forma, fundamentando-se na proposta de Koning [KON 2000] e com a utilização das ferramentas descritas no capítulo 5, objetiva-se propor um sistema de diagnóstico baseado em modelo, adaptativo, genérico e independente do domínio (desde que o domínio possa ser represenado em estruturas de tarefas *TÆMS*), descrito em detalhes nas próximas seções. Para diagnosticar o comportamento do aluno, estruturas de tarefas são criadas no *TÆMS*, e o comportamento é monitorado (um *log* das ações atuais do aluno é gravado). Assim, diferenças entre o modelo especificado e o comportamento atual do aluno podem ser diagnosticadas, concluindo-se eventualmente por que o aluno não realizou determinada tarefa como o esperado.

## 6.1 Descrição do modelo do sistema proposto

A FIGURA 6.1 apresenta uma visão geral do sistema de diagnóstico proposto que, além de ser genérica e independente do domínio, não requer catálogos de erros explícitos, mas a representação esperada do comportamento do aluno. Na verdade, esta também é uma

suposição delicada: nem sempre o que o professor considera a "representação correta do comportamento do aluno" é a mais adequada. Entretanto, o sistema encontra uma explicação para os desvios de comportamento e adapta a estrutura organizacional de tarefas a fim de que esta seja mais adequada para o aluno em interações futuras. A adaptação que, segundo Franco [FRA 93], é uma das características necessárias para que um sistema seja considerado inteligente, é indispensável em sistemas de diagnóstico, pois os alunos não irão necessariamente ter o mesmo comportamento no curso, o que não significa que tenham cometido erros. Isto pode acontecer porque podem existir mais que uma seqüência de tarefas que leve o aluno a atingir seu objetivo.

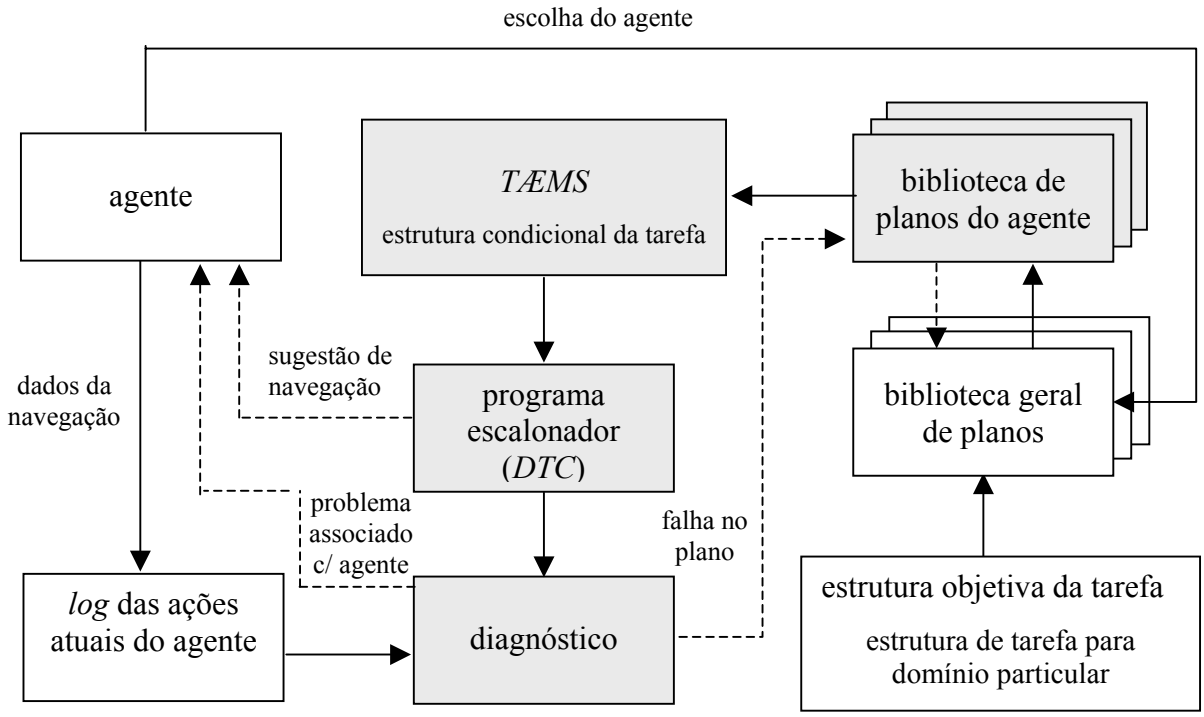


FIGURA 6.1 – Arquitetura conceitual do sistema de diagnóstico (quando ocorrer falha no plano, o processo representado pelos retângulos em cinza deve ser repetido)

Inicialmente, o professor cria um material específico para disponibilizar na *web*, que será descrito em estruturas de tarefas *TÆMS* (ferramenta descrita na seção 5.1). Esse material representa o que chamamos de *estrutura objetiva da tarefa* (ou nível objetivo, definido na seção 5.1.1). Esse nível é considerado objetivo, pois representa aquilo que o professor entende como "comportamento esperado". Baseado nessa estrutura de tarefas, planos são criados (inicialmente, antes de considerar as entradas do aluno, existirá um único plano com todo o material descrito pelo professor). Esses planos formam a biblioteca geral de planos do sistema, que pode conter planos para qualquer domínio possível.

O aluno, que é o agente ativo no sistema, faz suas escolhas de acordo com seu objetivo. Através de um formulário (FIGURA 7.3), ele seleciona um plano e indica se quer ver todo o conteúdo ou apenas parte deste. O sistema oferece duas formas de restrição: restrição de tempo ou restrição de conteúdo. Caso o aluno tenha restrição de tempo, este informa o tempo máximo disponível para estudar o plano escolhido e alguns planos são

gerados satisfazendo essa condição. A outra possibilidade seria escolher alguns dos conteúdos que fazem parte do plano selecionado (FIGURA 7.5). Neste caso, o sistema gera um plano contendo os conteúdos selecionados e todos que forem pré-requisitos desses.

Qualquer plano gerado a partir de seleções feitas pelo agente representa o que chamamos de *estrutura condicional da tarefa*. Esta é uma estrutura objetiva condicionada pela escolha subjetiva do agente, ou seja, um tipo de estrutura entre a estrutura objetiva e a estrutura subjetiva. Esta estrutura é interpretada pelo *DTC* (apresentado na seção 5.2) que retorna um ou mais escalonamentos para o conjunto de tarefas. Baseado nesse escalonamento de tarefas, o sistema sugere ao agente um "roteiro de estudo" (indicando quais as páginas *web* que devem ser exploradas). Enquanto o agente está navegando no material e executando as tarefas necessárias para atingir seu objetivo, informações referentes à navegação vão sendo gravadas em um *log*. O processo de diagnóstico ocorre quando as informações gravadas no *log* são confrontadas com o escalonamento gerado pelo *DTC*. Tal comparação é baseada em um modelo causal geral que pode ser utilizado para diagnosticar diferenças entre quaisquer estruturas *TÆMS*, tendo sido empregado em outros domínios como os descritos no capítulo 5, por exemplo.

Se forem detectadas divergências no processo de diagnóstico, é preciso verificar se estas foram geradas por algum problema associado ao agente ou por uma falha no plano. O problema é associado ao agente quando este não segue a sugestão gerada a partir da estrutura condicional da tarefa. Quando isso ocorre, o agente é avisado e tem a chance de reiniciar o processo. Se o agente não quiser reiniciar, este é um bom sinal de que ele está certo daquilo que fez. Neste caso, o sistema irá criar e incluir um novo plano na biblioteca de planos *para este aluno*. Entretanto, se o agente seguir a sugestão gerada a partir da estrutura condicional da tarefa e sintomas de falha forem encontrados, significa que o plano deve ser alterado e o processo repetido (este processo é representado pelos retângulos em cinza na FIGURA 6.1).

### 6.1.1 Como são gerados os planos

Os planos são gerados a partir dos materiais criados pelos professores para serem disponibilizados na *web*, cujos conteúdos são representados através de uma estrutura de tarefas *TÆMS*, que é, essencialmente, uma árvore de decomposição do objetivo destas tarefas, onde os nodos folhas representam métodos primitivos executáveis e os nodos internos provêm uma organização hierárquica.

Parte de uma estrutura de tarefas *TÆMS*, cujo domínio é a Computação Simbólica e Numérica, é apresentado na FIGURA 6.2 (a estrutura possui 127 nodos e seria, portanto, impossível de ser integralmente representada). As elipses representam as tarefas e subtarefas, e os retângulos os métodos. A FIGURA 6.3 detalha uma parte da estrutura de tarefas apresentada na FIGURA 6.2, com a distribuição de qualidade (Q), custo (C) e duração (D) de cada método, e os inter-relacionamentos *NLE's*. As distribuições de qualidade e tempo variam de acordo com o método; a distribuição do custo foi ignorada por não ser necessária neste caso. Os inter-relacionamentos que aparecem entre tarefas e métodos, são usados para indicar interações, como por exemplo *enable* (a tarefa *zf_12* habilita a tarefa *zf_17*), *facilitate* (a tarefa *zf_11* facilita a tarefa *zf_12*) (veja FIGURA 6.3).

As estruturas de tarefas *TÆMS* são descritas de forma textual, como mostra a FIGURA 6.4. A tarefa *zf_1* é uma das subtarefas da tarefa *teoria*, e tem como subtarefas *zf_11*, *zf_12*, *zf_13*, *zf_14*, *zf_15, zf_16* e *zf_17*. As tarefas que não têm subtarefas, como

por exemplo, *zf_11* (também descrita na FIGURA 6.4), são consideradas métodos e, dessa forma, apresentam a distribuição de qualidade, duração e custo. No exemplo, o professor estipulou que a qualidade esperada é 100, a duração é 5 (o custo é ignorado). A estrutura de tarefas completa para o domínio da Computação Simbólica e Numérica está descrita textualmente no Anexo 1.

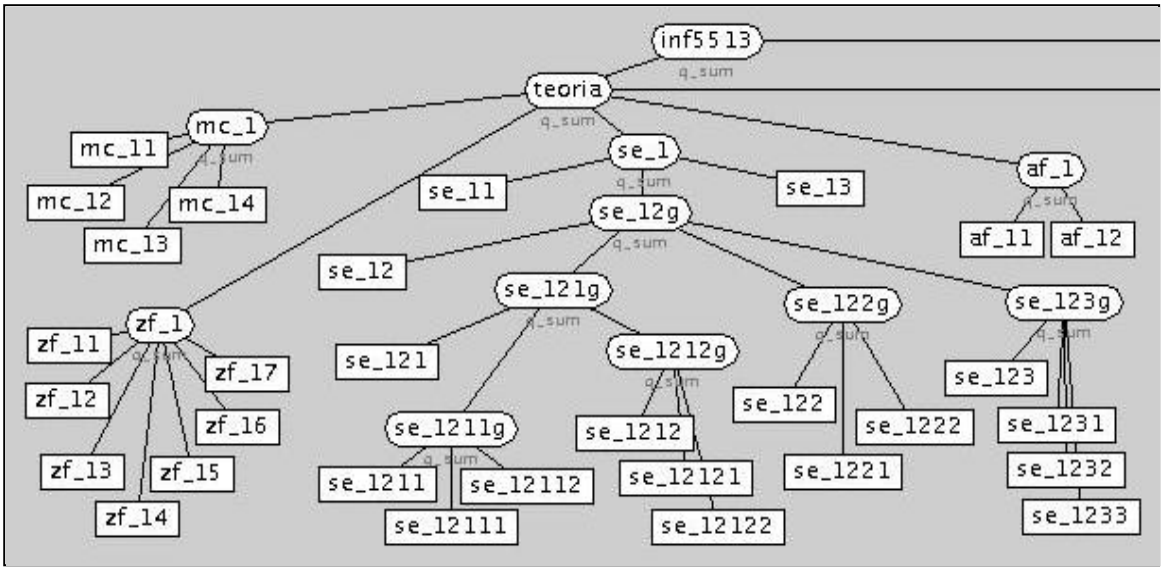


FIGURA 6.2 – Estrutura de tarefas *TÆMS* (representação parcial)

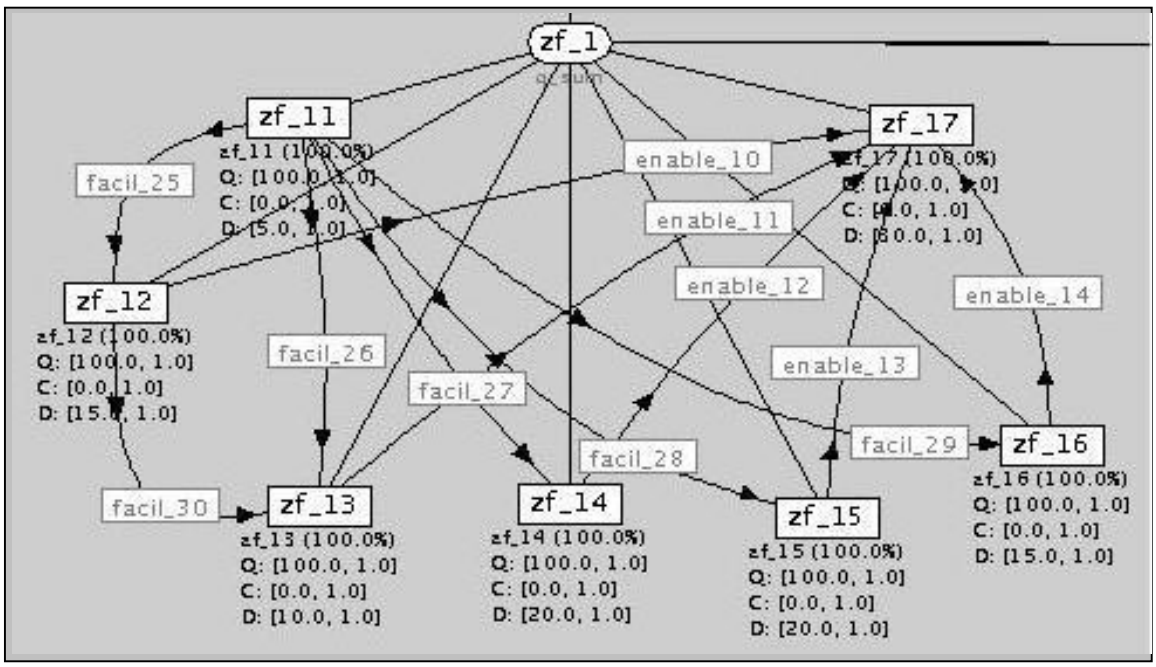


FIGURA 6.3 – Detalhe de parte da estrutura de tarefas *TÆMS* (tarefa zf_1 da Figura 6.2) com distribuição de qualidade e duração e os *NLEs*

O material criado pelo professor irá conter páginas dos tipos teoria, exemplo e exercício. Na realidade, a distribuição de qualidade só é relevante nas páginas do tipo exercício, com o valor definido pelo professor representando a porcentagem de acertos que ele acredita ser o aluno capaz de atingir, considerando o material que fora disponibilizado nas páginas dos tipos teoria e exemplo. Por exemplo, se o valor definido para a distribuição de qualidade de um determinado método é 85, significa que o professor espera que o aluno consiga acertar pelo menos 85% dos exercícios relacionados a este método. A distribuição de tempo é considerada em todos os tipos de páginas, pois representa o que o professor acredita ser o tempo máximo necessário para a compreensão dos conteúdos e/ou resolução dos exercícios propostos.

```
(spec_task
        (label teoria)
        (agent agent_A)
        (subtasks mc_1 zf_1 se_1 ip_1 af_1 ig_1 dn_1 ms_1)
        (supertasks inf5513)
        (qaf q_sum) )

   (spec_task
        (label zf_1)
        (agent agent_A)
        (subtasks zf_11 zf_12 zf_13 zf_14 zf_15 zf_16)
        (supertasks teoria)
        (qaf q_sum) )

(spec_method
        (label zf_11)
        (agent agent_A)
        (supertasks zf_1)
        (outcomes
                (zf_11
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 20  1.0)
                  (cost_distribution  0  1.0) )))
```

FIGURA 6.4 – Parte da descrição textual da estrutura de tarefas que gera a representação gráfica das FIGURAS 6.2 e 6.3

Depois de criado o material e, conseqüentemente, a estrutura de tarefas que irá descrevê-lo (chamada de estrutura objetiva da tarefa), o plano passa a fazer parte da biblioteca geral de planos do sistema. Os planos que fazem parte desta biblioteca são chamados planos gerais, porque suas estruturas de tarefas contêm todo o conteúdo do material criado pelo professor. A partir daí, com a interação do aluno que é o agente ativo no sistema, novos planos podem ser gerados e armazenados na biblioteca de planos do agente. Conforme descrito anteriormente, o aluno poderá escolher se deseja ver todo o conteúdo ou apenas parte dele (no último caso, poderá escolher entre fazer restrições de

conteúdo ou de tempo). Caso o aluno tenha escolhido apenas alguns dos conteúdos que fazem parte do plano geral e estes tenham conteúdos que sejam pré-requisitos ou que iriam facilitar o entendimento dos conteúdos selecionados, o sistema irá informá-lo disto. No caso de conteúdos que iriam apenas facilitar, o sistema pergunta ao aluno se este deve ou não ser incluído no plano e, no caso de conteúdos que sejam pré-requisitos, o sistema avisa que serão incluídos no plano. Com base nas escolhas do aluno, o sistema gera um novo plano e, é claro, uma nova estrutura de tarefas (chamada estrutura condicional da tarefa).

### 6.1.2 A função do *DTC*

O *DTC* recebe a estrutura de tarefa condicional, que é interpretada, e retorna um ou mais escalonamentos possíveis para o conjunto de tarefas. Para isto, são consideradas as distribuições de qualidade e duração associadas a cada método e, principalmente, os inter-relacionamentos. Baseado em um dos *schedules* gerados, o sistema sugere um roteiro de estudo para o aluno.

Um exemplo da saída gerada pelo *DTC* pode ser visto na FIGURA 6.5. Cada linha representa um método e contém as seguintes informações: nome do método, tempo inicial previsto, tempo final previsto, qualidade, custo e duração. A primeira linha, por exemplo, indica que o método *zf_11* deve iniciar no tempo 0 e terminar depois de 5 unidades de tempo, devendo ser executado com uma distribuição de qualidade igual a 100 (o custo é ignorado).

```
zf_11 0.000000 5.000000 100.000000 0.000000 5.000000
zf_12 5.000000 20.000000 10100.000000 0.000000 15.000000
zf_16 20.000000 35.000000 10100.000000 0.000000 15.000000
zf_13 35.000000 45.000000 1020100.000000 0.000000 10.000000
zf_14 45.000000 65.000000 10100.000000 0.000000 20.000000
zf_15 65.000000 85.000000 10100.000000 0.000000 20.000000
zf_17 85.000000 115.000000 100.000000 0.000000 30.000000
10410120192.000000 0.000000 115.000000
```

FIGURA 6.5 – Saída gerada pelo *DTC*

### 6.1.3 O monitoramento

O roteiro de estudo, sugerido pelo sistema ao aluno, poderá ou não ser seguido fielmente. O aluno poderá ver somente os conteúdos listados no roteiro, através de botões de avançar e voltar, ou poderá navegar mais livremente através de *links* internos que fazem parte das páginas. Esses *links*, porém, não permitem ao aluno poder navegar por todo o material, mas apenas nas páginas que estão associadas aos conteúdos relacionados à tarefa que está sendo executada. Esta forma foi escolhida para que o aluno não precise necessariamente seguir apenas os conteúdos sugeridos pelo sistema e, ao mesmo tempo, não corra o risco de "se perder no material".

Enquanto o aluno navega pelo material, algumas informações vão sendo armazenadas, como por exemplo: as páginas que foram acessadas, o tempo de permanência em cada uma delas, e no caso de página do tipo exercício, o número de acertos e o número

de erros, etc. A TABELA 6.1 apresenta um exemplo de alguns dos dados armazenados através do monitoramento.

TABELA 6.1 – Dados obtidos através do monitoramento

| Data | Hora | Cod_aluno | Cod_URL |
|---|---|---|---|
| 10/11/2001 | 10:25:50 | 000016 | 000078 |
| 10/11/2001 | 10:53:12 | 000016 | 000081 |
| 10/11/2001 | 11:14:43 | 000016 | 000082 |
| 10/11/2001 | 11:34:55 | 000016 | 000085 |
| 10/11/2001 | 11:59:02 | 000016 | 000093 |

### 6.1.4 O processo de diagnóstico

A principal função do sistema de diagnóstico é detectar desvios no comportamento do aluno em relação ao plano, encontrar as causas para eles e, se necessário, adaptar o plano original. A forma como o aluno interage com o sistema, reflete seu comportamento (este é principal motivo de executar o monitoramento). Assim, confrontando os dados do *log*, que representam o comportamento real do aluno, com os dados do plano, é possível gerar o diagnóstico do comportamento do aluno.

O processo de comparação é realizado com a ajuda de um modelo causal (veja FIGURA 6.6), que explica as diferenças baseado num conjunto de sintomas, todos relacionados a estrutura da tarefa, sem a necessidade de nenhum conhecimento específico do domínio.

#### *6.1.4.1 O modelo causal*

O modelo causal é um gráfico que mapeia explicações para sintomas, como pode ser observado na FIGURA 6.6. Ele é planejado para ser o mais genérico possível, capaz de analisar várias organizações descritas usando *TÆMS*. Quando um sintoma é detectado através da comparação do comportamento esperado e do comportamento real do aluno, há um conjunto de possíveis explicações para isto, que são independentes do domínio.

As principais classes de sintomas identificados são duração, qualidade e custo inesperados na execução de um método, recursos sobrecarregados ou compromissos que não são satisfeitos. Neste trabalho, custo e recursos não são considerados (detalhes sobre estas classes de sintomas podem ser encontrados em [BAZ 98, HOR 9–a, HOR 2000]). A seguir, serão descritos os sintomas e as possíveis explicações para cada um deles, utilizados neste trabalho.

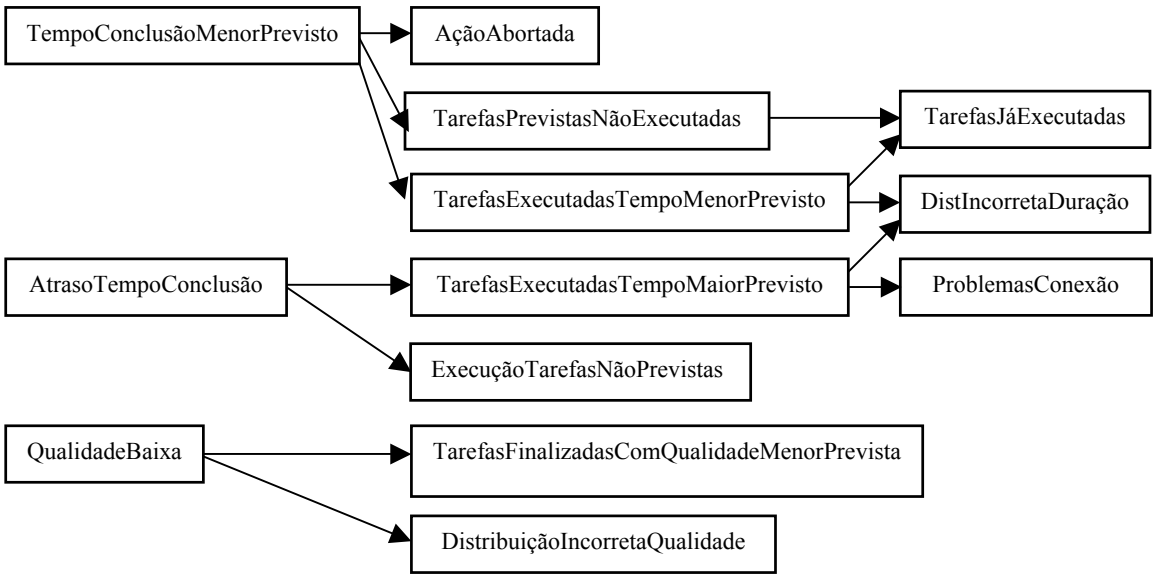


FIGURA 6.6 – Modelo causal

*a) Explicações para sintomas relacionados à duração do método*

Quando o sintoma detectado é a execução de uma tarefa num tempo menor que o previsto, as possíveis explicações são:

- tempo de conclusão menor que o previsto, que pode ser então explicado por:
  - ação abortada;
  - tarefas previstas e não executadas (o aluno não executou todas as tarefas contidas no plano sugerido pelo sistema);
  - tarefas executadas num tempo menor que o previsto;
  - tarefas já executadas (o aluno já executou a tarefa anteriormente, em outra situação) ;
  - distribuição incorreta de duração (a distribuição estatística pode estar incorreta);

- atraso no tempo de conclusão, que pode ser então explicado por:
  - tarefas executadas num tempo maior que o previsto;
  - execução de tarefas não previstas (o aluno executou algumas tarefas que não faziam parte do plano sugerido pelo sistema);
  - problemas na conexão;
  - distribuição incorreta de duração (a distribuição estatística pode estar incorreta).

*b) Explicações para sintomas relacionados à qualidade do método*

Quando o sintoma detectado é a execução de uma tarefa com qualidade abaixo da esperada, as possíveis explicações são:

- qualidade baixa, que pode ser então explicado por:
  - tarefas finalizadas com qualidade menor que a prevista;
  - distribuição incorreta de qualidade (a distribuição estatística pode estar incorreta).

# 7 Descrição da implementação do protótipo

A partir do modelo descrito no capítulo anterior, foi implementado o sistema DIACOM, que é um protótipo que provê as funcionalidades do modelo proposto, com o objetivo de validá-lo. Este capítulo apresenta o processo de construção do protótipo, os experimentos realizados a fim de validá-lo, e os resultados desta validação.

A seção 7.1 contém algumas considerações sobre o ambiente de implementação utilizado no desenvolvimento do protótipo. A seção 7.2 apresenta uma visão geral do sistema DIACOM e a descrição de sua interface. Na seção 7.3 são relatados os resultados obtidos através de testes realizados com o sistema DIACOM. Considerações gerais sobre a implementação são apresentadas na seção 7.4.

## 7.1 Ambiente de implementação

A implementação do sistema DIACOM exigiu vários recursos de *software*, entre eles, editor de páginas *web*, linguagens de programação, sistema de banco de dados e um programa servidor de *software* para serviços específicos de *http*:

- editor de páginas *web – Netscape Composer*, para gerar o material do curso utilizado para validar o protótipo;
- *scripts HTML* [FEA 97, SAV 97], utilizados para gerar páginas *web* dinâmicas através do sistema;
- linguagem de programação *Java* [DEI 99, LEM 99, JAV 2001], com a qual foi implementado o protótipo, por permitir um desenvolvimento rápido, orientação a objetos, interface gráfica e, principalmente, ser compatível com o ambiente Linux;
- sistema de banco de dados relacional *MySQL* [MAS 2000], por estar disponível para o ambiente Linux;
- *Apache Web Server* 4.0 [APA 2001], utilizado como servidor *http*, por exigir poucos recursos computacionais, além de estar disponível para a o ambiente Linux.

A implementação e os testes do sistema DIACOM foram feitos em um computador Pentium III 800 MHz, com 192 MB de memória RAM, 17 GB de disco rígido e com sistema operacional Linux. Optou-se por utilizar o sistema operacional Linux por ser compatível com o *DTC* (sistema que faz o escalonamento das tarefas, descrito na seção 5.2), do qual somente se tem o arquivo executável para esta plataforma.

## 7.2 Uma visão geral do sistema

O sistema DIACOM está dividido em três módulos principais: gerar plano, monitorar e diagnosticar, conforme a representação da FIGURA 7.1:

- gerar plano: recebe um arquivo com a estrutura de tarefas *TÆMS* (estrutura objetiva da tarefa) referente ao plano escolhido pelo aluno, dependendo da opção selecionada (sem restrição, com restrição de tempo, com restrição de conteúdo), gera um novo plano, ou seja, uma nova estrutura de tarefas *TÆMS* (estrutura condicional de tarefas) e, com base nesta estrutura, sugere um roteiro de estudo ao aluno;

- monitorar: grava informações no banco de dados do sistema, referentes à navegação do aluno pelo material, como quais páginas foram acessadas, qual o tempo de permanência em cada uma delas e, no caso de páginas do tipo exercício, grava também o número de acertos e de erros cometidos pelo aluno;
- diagnosticar: recebe informações sobre o plano gerado pelo módulo gerar plano e os dados do *log* gravados no BD (Banco de Dados) pelo módulo de monitoramento, confronta estas informações e, baseado num modelo causal, gera um arquivo texto com os sintomas detectados e possíveis diagnósticos.

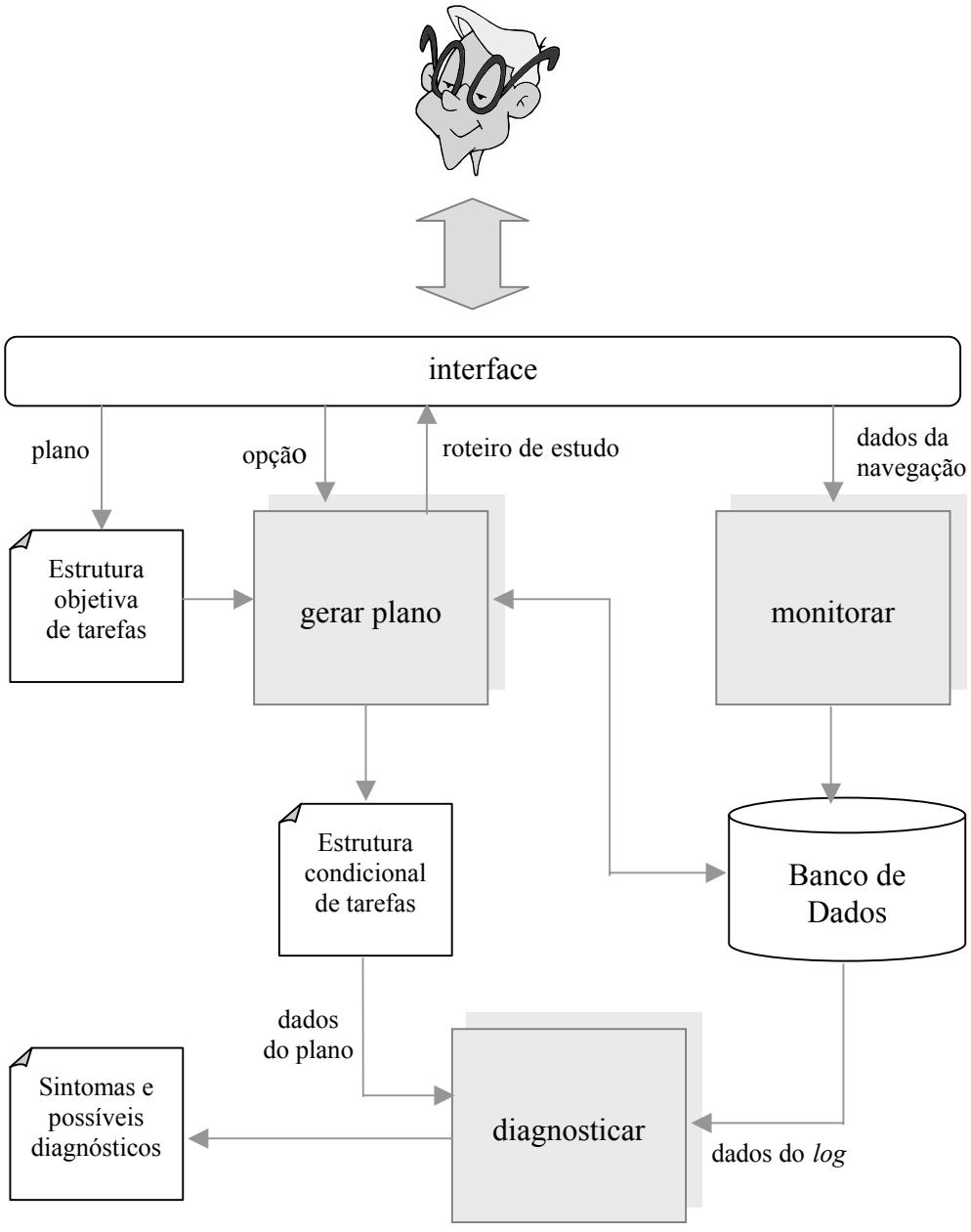


FIGURA 7.1 – Visão geral do sistema DIACOM

### 7.2.1 Descrição da interface

A FIGURA 7.2 mostra a tela de identificação, em que o aluno informa seu nome e sua senha (os alunos devem ser cadastrados previamente no banco de dados do sistema). Depois de identificado, o aluno poderá fazer suas escolhas e, baseado nisso, o sistema irá gerar um roteiro de estudo.


FIGURA 7.2 – Tela de identificação

Os planos que fazem parte da biblioteca geral de planos do sistema são listados na parte superior da tela apresentada na FIGURA 7.3. O aluno deverá selecionar o plano referente ao curso que deseja estudar e, a seguir, informar como pretende gerar seu roteiro de estudo, escolhendo uma entre as três opções disponíveis: sem restrições, com restrição de tempo ou com restrição de conteúdo.

Se o aluno escolhe gerar um plano sem restrições, o sistema irá gerar um roteiro de estudo com todos os conteúdos que fazem parte do plano selecionado. Quando o aluno quiser gerar um plano com restrição de tempo, deverá informar ao sistema o tempo que pretende estudar (veja FIGURA 7.4) e, desta forma, o roteiro de estudo gerado conterá apenas os conteúdos possíveis de serem estudados no tempo especificado pelo aluno, o que é determinado pelo escalonamento.

A última opção, com restrição de conteúdo, permite ao aluno selecionar os conteúdos que deseja estudar, como mostra a FIGURA 7.5. Neste caso, o aluno escolhe quais dos conteúdos da lista da esquerda devem ser incluídos no plano, seleciona-os e "joga-os" para a lista da direita. O sistema gerará o plano com os conteúdos incluídos na lista da direita, porém, antes disso, ele verifica se não existem conteúdos não selecionados pelo aluno, que possam facilitar a compreensão ou que sejam pré-requisito de conteúdos selecionados. No caso de conteúdos que facilitam a compreensão, o sistema emitirá mensagens de aviso, perguntando ao aluno se ele quer ou não incluir o conteúdo no plano (veja um exemplo na FIGURA 7.6). Quando um conteúdo não selecionado for pré-requisito de outro selecionado, o sistema informará ao aluno que tal conteúdo é obrigatório, e será incluído no plano (um exemplo pode ser visto na FIGURA 7.7).

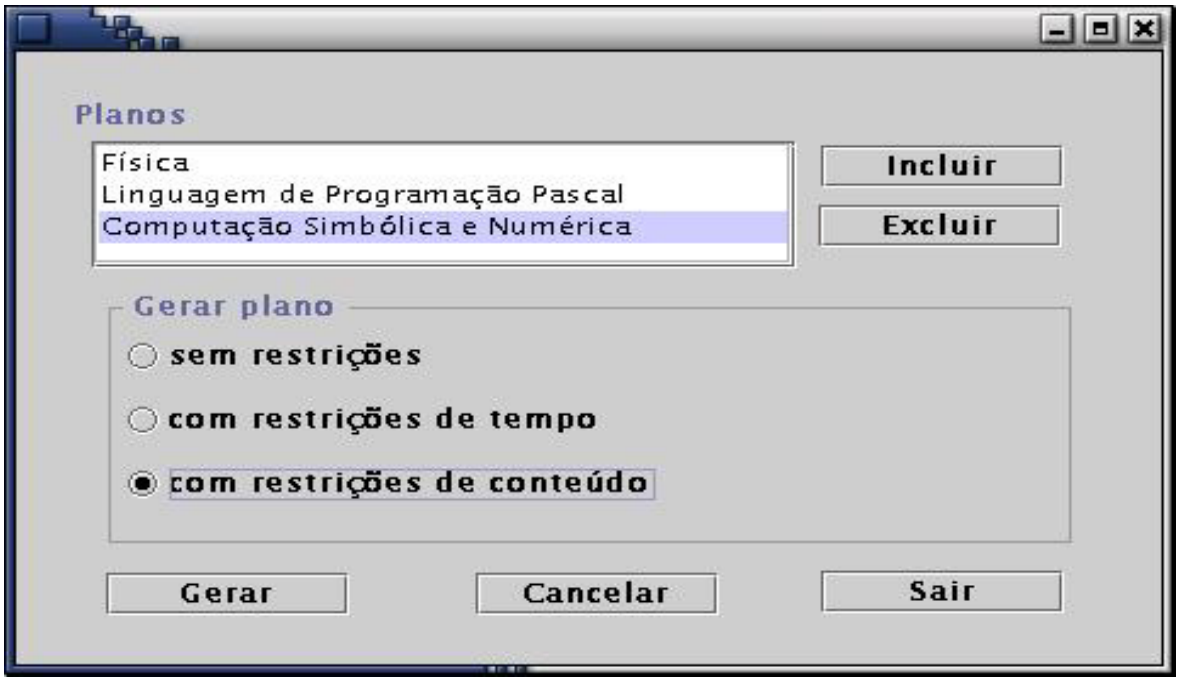


FIGURA 7.3 – Escolha do plano e a forma como este será gerado


FIGURA 7.4 – Tempo de que o aluno dispõe para executar o plano

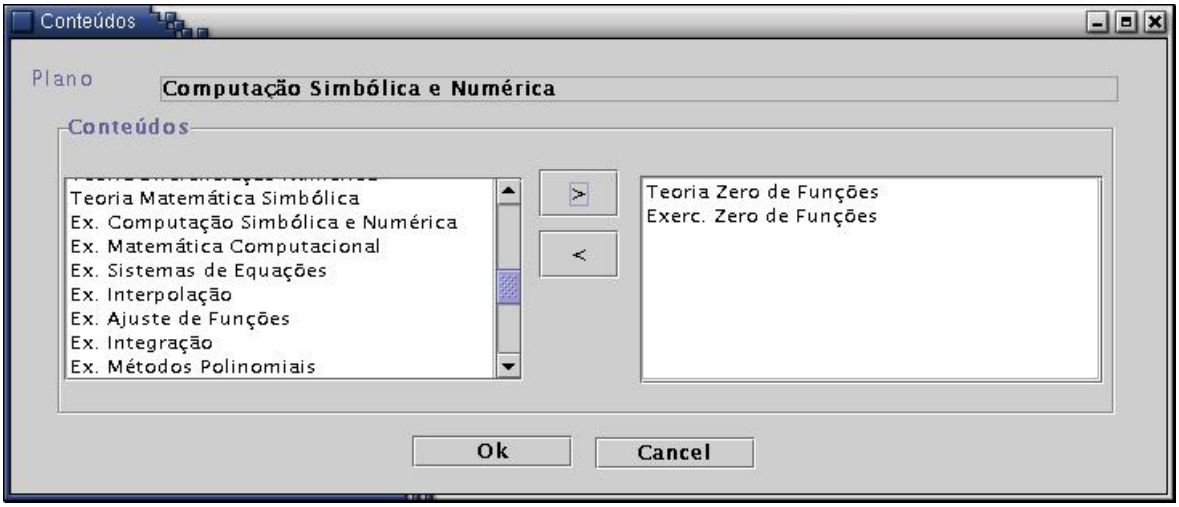


FIGURA 7.5 – O aluno seleciona os conteúdos que deseja estudar

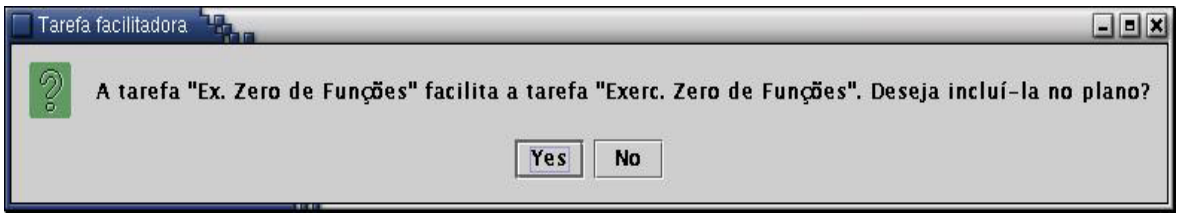


FIGURA 7.6 – Um conteúdo pode facilitar outro


FIGURA 7.7 – Um conteúdo é pré-requisito de outro

Depois de o aluno ter feito suas escolhas e o sistema ter gerado o plano, o *Netscape Comunicator* será executado (o próprio sistema ativa este browser) e uma página contendo o roteiro de estudo é apresentada ao aluno (veja FIGURA 7.8). A seguir, o aluno deverá clicar no botão "Iniciar". Deste ponto em diante, as páginas são divididas em dois *frames*: o *frame* da esquerda contém o roteiro de estudo e dois botões, que permitem ao aluno navegar pelas páginas que fazem parte do roteiro, e o botão "Finalizar", que indica quando o aluno parou de estudar; e o *frame* da direita contém a página referente ao conteúdo listado no roteiro (que aparece no frame da esquerda) em cor diferente (veja a FIGURA 7.9).

Quando o aluno clica no botão "Finalizar", o sistema gerará um arquivo texto, com o diagnóstico deste aluno. O arquivo contém o código do aluno, o código do plano executado, os sintomas detectados e os possíveis diagnósticos (os diagnósticos gerados são explicados com mais detalhes na seção 7.3).


FIGURA 7.8 – Roteiro de estudo

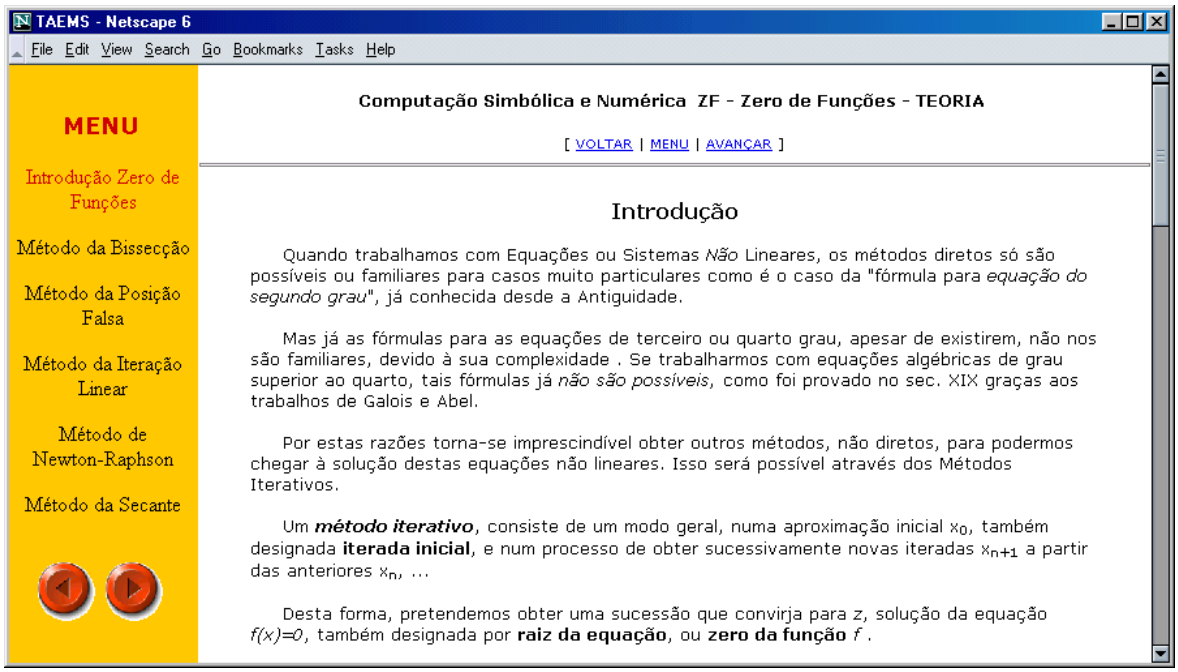


FIGURA 7.9 – Exemplo da interface do curso

### 7.2.2 O banco de dados

O modelo entidade-relacionamento (ER) do sistema DIACOM é apresentado na FIGURA 7.10. O modelo ER possui oito entidades (alunos, planos, planos_alunos, tarefas, métodos, urls, *logs* e avaliações) das quais cinco são previamente preenchidas: alunos, planos, tarefas, métodos e urls (veja o Anexo 2).

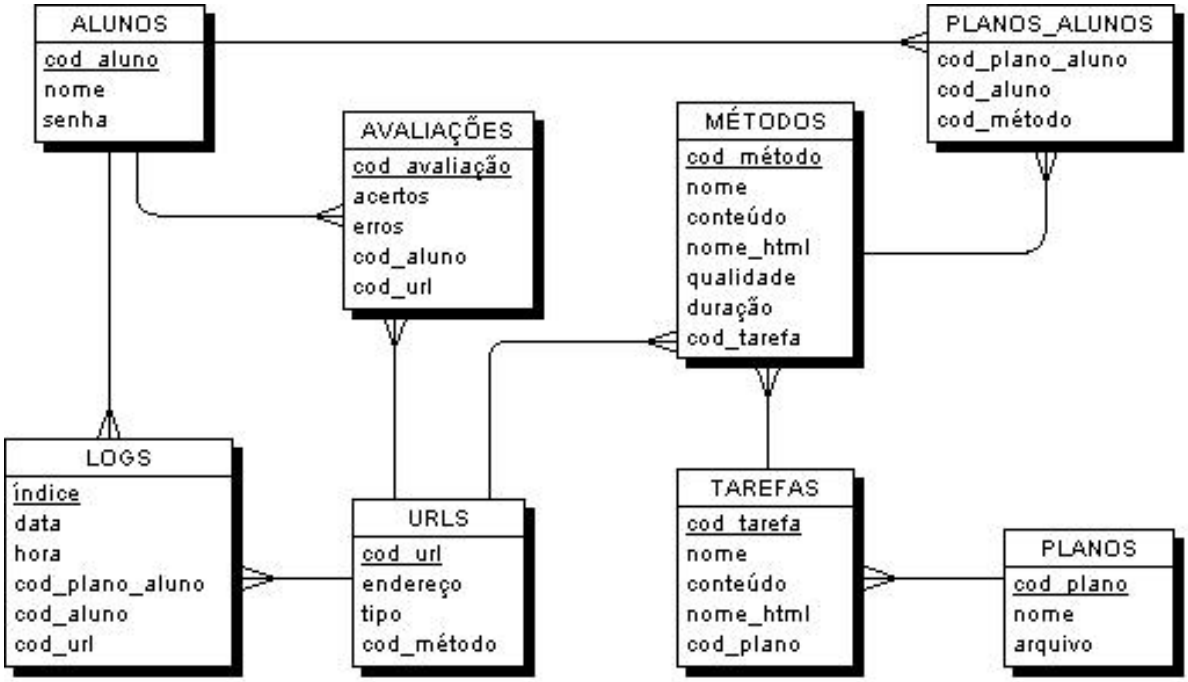


FIGURA 7.10 – O modelo ER do sistema DIACOM

## 7.3 Resultados

O plano inicial era o desenvolvimento do material da disciplina Computação Simbólica e Numérica no semestre 2001/1 (o que aconteceu) e seu emprego na turma do semestre 2001/2, a fim de comparar o desempenho das turmas e, é claro, validar o sistema DIACOM. Porém, devido à mudança de calendário do semestre letivo isso não foi possível. O semestre 2001/2 iniciou na segunda quinzena de dezembro e com apenas algumas semanas de aula, os alunos teriam muita dificuldade em usar o material e, desta forma, gerariam diagnósticos com muitos sintomas.

Dada a impossibilidade de realizar-se um experimento de longo prazo devido às restrições de tempo inerentes ao projeto de dissertação, foram realizados alguns experimentos a fim de observar os resultados gerados pelo sistema em algumas situações. O domínio utilizado para gerar os exemplos foi o curso de Computação Simbólica e Numérica, especificamente o conteúdo Zeros de Funções.

### 7.3.1 Situação A – Tempo de conclusão menor que o previsto

Quando o sistema detecta um sintoma de tempo de conclusão menor que o previsto, os diagnósticos possíveis são: ação abortada, tarefas previstas e não executadas ou tarefas

executadas em tempo menor que o previsto, sendo que as possíveis explicações seriam as tarefas já terem sido executadas pelo aluno em outra situação (o aluno já possui um conhecimento prévio do conteúdo daquela tarefa e pode executá-la em menos tempo) ou a distribuição de duração das tarefas (estipulada pelo professor quando da criação da estrutura de tarefas) estar incorreta. A seguir, serão descritos dois exemplos em que o sistema detecta esses sintomas.

O aluno fez suas opções e, baseado nisso, o sistema gerou um plano de estudo conforme o descrito na TABELA 7.1. Enquanto o aluno navegava pelo material, informações foram armazenadas, como mostra a TABELA 7.2.

TABELA 7.1 – Roteiro de estudo

| **Conteúdos** |
|---|
| Introdução Zeros de Funções |
| Método da Bissecção |
| Método da Posição Falsa |
| Método da Iteração Linear |
| Método de Newton-Raphson |
| Método da Secante |
| Comparação entre os métodos |

TABELA 7.2 – *Log* referente à navegação do aluno

| **Índice** | **Data** | **Hora** | **Cod_aluno** | **Cod_plano_aluno** | **Cod_url** |
|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... |
| 000081 | 10.01.2002 | 08:06:25 | 000001 | 000005 | 000005 |
| 000083 | 10.01.2002 | 08:09:27 | 000001 | 000005 | 000006 |
| 000084 | 10.01.2002 | 08:21:38 | 000001 | 000005 | 000007 |
| 000085 | 10.01.2002 | 08:28:45 | 000001 | 000005 | 000008 |
| 000086 | 10.01.2002 | 08:44:46 | 000001 | 000005 | 000000 |
| ... | ... | ... | ... | ... | ... |

O processo de diagnóstico inicia quando as informações da TABELA 7.1 são confrontadas com as informações da TABELA 7.2, o que pode ser observado na TABELA 7.3. Encontradas divergências, o sistema detecta que algumas tarefas sugeridas no roteiro

de estudo não foram executadas e, das tarefas executadas, algumas foram realizadas em tempo menor que o previsto. Então, baseado no modelo causal descrito na seção 6.1.4.1, o sistema gera um arquivo texto indicando os sintomas detectados e suas possíveis causas. O diagnóstico gerado pode ser visto na FIGURA 7.11.

TABELA 7.3 – Comparação entre o tempo previsto e o tempo real de execução das tarefas

| **Conteúdo** | **Duração prevista** | **Duração real** |
|---|---|---|
| Introdução Zeros de Funções | 05 min | 03 min |
| Método da Bissecção | 15 min | 12 min |
| Método da Posição Falsa | 10 min | 07 min |
| Método da Iteração Linear | 20 min | 16 min |
| Método de Newton-Raphson | 20 min | - |
| Método da Secante | 15 min | - |
| Comparação entre os métodos | 30 min | - |
| **Tempo total** | **115** min | **38** min |

```
Código do aluno: 000001
Código do plano do aluno: 000005

• 03 tarefas previstas não foram executadas
  Método de Newton-Raphson
  Método da Secante
  Comparação entre os métodos

  Tarefas já executadas

• 04 tarefas foram executadas com tempo menor que o previsto
  Introdução Zeros de Funções
  Método da Bissecção
  Método da Posição Falsa
  Método da Iteração Linear

  Tarefas já executadas
  Distribuição incorreta de duração
```

FIGURA 7.11 – Exemplo de diagnóstico – Situação A

No segundo experimento, o aluno gerou o mesmo plano (descrito na TABELA 7.1), porém seu comportamento durante a navegação pelo material foi diferente, o que pode ser observado na TABELA 7.4. Neste caso, confrontando as informações do plano com as informações do *log* (veja a TABELA 7.5), o sistema detectou que todas as tarefas foram realizadas, porém, algumas foram executadas num tempo menor que o previsto, gerando o diagnóstico descrito na FIGURA 7.12.

TABELA 7.4 – *Log* referente à navegação do aluno

| Índice | Data | Hora | Cod_aluno | Cod_plano_aluno | Cod_url |
|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... |
| 000087 | 10.01.2002 | 10:16:08 | 000002 | 000006 | 000005 |
| 000088 | 10.01.2002 | 10:20:14 | 000002 | 000006 | 000006 |
| 000089 | 10.01.2002 | 10:33:16 | 000002 | 000006 | 000007 |
| 000090 | 10.01.2002 | 10:43:32 | 000002 | 000006 | 000008 |
| 000091 | 10.01.2002 | 11:02:38 | 000002 | 000006 | 000009 |
| 000092 | 10.01.2002 | 11:19:50 | 000002 | 000006 | 000010 |
| 000093 | 10.01.2002 | 11:33:54 | 000002 | 000006 | 000011 |
| 000094 | 10.01.2002 | 12:04:55 | 000002 | 000006 | 000000 |
| ... | ... | ... | ... | ... | ... |

TABELA 7.5 – Comparação entre o tempo previsto e o tempo real de execução das tarefas

| Conteúdo | Duração prevista | Duração real |
|---|---|---|
| Introdução Zeros de Funções | 05 | 04 |
| Método da Bissecção | 15 | 13 |
| Método da Posição Falsa | 10 | 10 |
| Método da Iteração Linear | 20 | 19 |
| Método de Newton-Raphson | 20 | 17 |
| Método da Secante | 15 | 14 |
| Comparação entre os métodos | 30 | 30 |
| **Tempo total** | **115** | **107** |

```
Código do aluno: 000002
Código do plano do aluno: 000006

• 05 tarefas foram executadas com tempo menor que o previsto
  Introdução Zeros de Funções
  Método da Bissecção
  Método da Iteração Linear
  Método de Newton-Raphson
  Método da Secante

  Tarefas já executadas
  Distribuição incorreta de duração
```

FIGURA 7.12 – Exemplo de diagnóstico – Situação A

TABELA 7.6 – *Log* referente à navegação do aluno

| Índice | Data | Hora | Cód_aluno | Cod_plano_aluno | Cod_url |
|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... |
| 000095 | 10.01.2002 | 13:26:09 | 000003 | 000007 | 000005 |
| 000096 | 10.01.2002 | 13:31:21 | 000003 | 000007 | 000006 |
| 000097 | 10.01.2002 | 13:49:26 | 000003 | 000007 | 000007 |
| 000098 | 10.01.2002 | 14:01:29 | 000003 | 000007 | 000008 |
| 000099 | 10.01.2002 | 14:27:38 | 000003 | 000007 | 000009 |
| 000100 | 10.01.2002 | 14:48:54 | 000003 | 000007 | 000010 |
| 000101 | 10.01.2002 | 15:03:56 | 000003 | 000007 | 000011 |
| 000102 | 10.01.2002 | 15:41:59 | 000003 | 000007 | 000000 |
| ... | ... | ... | ... | ... | ... |

### 7.3.2 Situação B – Atraso no tempo de conclusão

Se o sintoma detectado for atraso no tempo de conclusão das tarefas, o sistema indica como diagnósticos: tarefas executadas num tempo maior que o previsto (neste caso pode ter ocorrido erro na distribuição de duração da tarefa) ou execução de tarefas não previstas,

ou seja, o aluno executou tarefas que não estavam incluídas no roteiro sugerido pelo sistema.

Neste experimento, o aluno gerou o plano da TABELA 7.1 e navegou pelas páginas do curso conforme os dados apresentados na TABELA 7.6. Confrontando estas informações (veja a TABELA 7.7), o sistema encontrou divergências, ou seja, detectou que algumas tarefas foram executadas num tempo maior que o previsto, gerando o diagnóstico listado na FIGURA 7.13.

TABELA 7.7 – Comparação entre o tempo previsto e o tempo real de execução das tarefas

| Conteúdo | Duração prevista | Duração real |
|---|---|---|
| Introdução Zeros de Funções | 5 | 5 |
| Método da Bissecção | 15 | 18 |
| Método da Posição Falsa | 10 | 12 |
| Método da Iteração Linear | 20 | 26 |
| Método de Newton-Raphson | 20 | 21 |
| Método da Secante | 15 | 15 |
| Comparação entre os métodos | 30 | 38 |
| **Tempo total** | **115** | **135** |

```
Código do aluno: 000003
Código do plano do aluno: 000007

• 05 tarefas foram executadas com tempo maior que o previsto
  Método da Bissecção
  Método da Posição Falsa
  Método da Iteração Linear
  Método de Newton-Raphson
  Comparação entre os métodos

  Distribuição incorreta de duração

  Problemas na conexão
```

FIGURA 7.13 – Exemplo de diagnóstico – Situação B

Outra situação em que o sistema detecta o sintoma de tempo total de conclusão das tarefas maior que o previsto pode ser observada no próximo exemplo. O aluno gerou um plano conforme o descrito na TABELA 7.1, porém executou as tarefas listadas na TABELA 7.8. Assim, confrontando estas informações (veja a TABELA 7.9), o sistema detecta que foram executadas mais tarefas do que as que faziam parte do roteiro de estudo sugerido inicialmente. A FIGURA 7.14 apresenta o diagnóstico gerado para este caso.

TABELA 7.8 – *Log* referente à navegação do aluno

| **Índice** | **Data** | **Hora** | **Cod_aluno** | **Cod_plano_aluno** | **Cod_url** |
|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... |
| 000103 | 10.01.2002 | 16:05:27 | 000004 | 000008 | 000005 |
| 000104 | 10.01.2002 | 16:10:51 | 000004 | 000008 | 000006 |
| 000105 | 10.01.2002 | 16:25:44 | 000004 | 000008 | 000007 |
| 000106 | 10.01.2002 | 16:36:15 | 000004 | 000008 | 000008 |
| 000107 | 10.01.2002 | 16:56:24 | 000004 | 000008 | 000009 |
| 000108 | 10.01.2002 | 17:17:51 | 000004 | 000008 | 000010 |
| 000109 | 10.01.2002 | 17:32:52 | 000004 | 000008 | 000011 |
| 000110 | 10.01.2002 | 18:03:58 | 000004 | 000008 | 000053 |
| 000111 | 10.01.2002 | 18:13:11 | 000004 | 000008 | 000054 |
| 000112 | 10.01.2002 | 18:26:36 | 000004 | 000008 | 000000 |
| ... | ... | ... | ... | ... | ... |

TABELA 7.9 – Comparação entre o tempo previsto e o tempo real de execução das tarefas

| Conteúdo | Duração prevista | Duração real |
|---|---|---|
| Introdução Zeros de Funções | 5 | 5 |
| Método da Bissecção | 15 | 15 |
| Método da Posição Falsa | 10 | 11 |
| Método da Iteração Linear | 20 | 20 |
| Método de Newton-Raphson | 20 | 21 |
| Método da Secante | 15 | 15 |
| Comparação entre os métodos | 30 | 31 |
| Ex. Método da Bissecção | - | 10 |
| Ex. Método da Posição Falsa | - | 13 |
| **Tempo total** | **115** | **141** |

```
Código do aluno: 000004
Código do plano do aluno: 000008

• 02 tarefas não previstas foram executadas
  Ex. Método da Bissecção
  Ex. Método da Posição Falsa

• 03 tarefas foram executadas com tempo maior que o previsto
  Método da Posição Falsa
  Método de Newton-Raphson
  Comparação entre os métodos

  Distribuição incorreta de duração

  Problemas na conexão
```

FIGURA 7.14 – Exemplo de diagnóstico – Situação B

### 7.3.3 Situação C – Qualidade baixa

Quando o sintoma detectado é de qualidade baixa, existem dois diagnósticos possíveis: tarefas finalizadas com qualidade menor que a esperada (o aluno não fez os exercícios propostos ou fez mas não atingiu a qualidade esperada) ou a distribuição de qualidade das tarefas está incorreta.

Esse sintoma só poderá ocorrer quando o aluno acessar páginas do tipo exercício, pois nas páginas dos tipos teoria e exemplo, a qualidade é ignorada. Assim, considere-se o plano gerado pelo aluno apresentado na TABELA 7.10, e a comparação entre a qualidade prevista e a qualidade real atingida pelo aluno, apresentadas na TABELA 7.11, para entender o diagnóstico descrito na FIGURA 7.15.

TABELA 7.10 – Roteiro de estudo

| **Conteúdos** |
|---|
| Introdução Zeros de Funções |
| Método da Bissecção |
| Método da Posição Falsa |
| Método da Iteração Linear |
| Método de Newton-Raphson |
| Método da Secante |
| Comparação entre os métodos |
| Exerc. Método da Bissecção |
| Exerc. Método da Posição Falsa |
| Exerc. Método da Iteração Linear |
| Exerc. Método de Newton-Raphson |
| Exerc. Método da Secante |

TABELA 7.11 – Comparação da qualidade prevista com a qualidade real. O valor da qualidade real de cada tarefa equivale a porcentagem de acertos que o aluno atingiu em determinada página.

| Conteúdos | Qualidade prevista | Qualidade real |
|---|---|---|
| Introdução Zeros de Funções | 100 | - |
| Método da Bissecção | 100 | - |
| Método da Posição Falsa | 100 | - |
| Método da Iteração Linear | 100 | - |
| Método de Newton-Raphson | 100 | - |
| Método da Secante | 100 | - |
| Comparação entre os métodos | 100 | - |
| Exerc. Método da Bissecção | 85 | 60 |
| Exerc. Método da Posição Falsa | 85 | 65 |
| Exerc. Método de Newton-Raphson | 80 | 50 |
| Exerc. Método da Secante | 80 | 60 |

```
Código do aluno: 000005
Código do plano do aluno: 000009

• Tarefas executadas com qualidade mais baixa que a prevista

• Distribuição incorreta de qualidade
```

FIGURA 7.15 – Exemplo de diagnóstico – Situação C

Considerando os cenários descritos acima, observa-se que nas situações A, B e C, os sintomas detectados podem ter ocorrido tanto por um problema associado ao aluno, quanto por uma falha no plano. Quando a causa para o sintoma detectado é “distribuição incorreta de duração” ou “distribuição incorreta de qualidade”, considera-se que o sintoma fora causado por uma falha no plano, porém, quando a causa para o sintoma detectado é “tarefas já executadas” ou “tarefas executadas sem qualidade”, considera-se que o sintoma fora causado pelo aluno. Essas situações podem ser observadas na arquitetura conceitual do sistema de diagnóstico, representada na FIGURA 6.1.

## 7.4 Considerações gerais sobre a implementação

Os testes com o protótipo foram realizados em uma máquina cliente, com sistema operacional Linux, em que estava instalado o sistema DIACOM. No servidor, além do sistema operacional Linux, estavam instalados o *Web Server Apache 4.0*, o *MySQL* e o *DTC* (o servidor mantém o BD e o material instrucional, que são acessados pela máquina cliente através de *servlets Java*). Informações detalhadas sobre a instalação e configurações necessárias para executar o sistema DIACOM estão descritas no Anexo 3.

O protótipo desenvolvido possibilitou a implementação integral de todas as funcionalidades previstas no modelo proposto no capítulo 6. Trabalhos futuros poderão aprimorar essas funcionalidades, como criar uma ferramenta que gere as estruturas de tarefas automaticamente através das páginas *web* criadas pelo professor. Esta e outras sugestões de trabalhos futuros são descritas no capítulo 8.

# 8 Conclusões e trabalhos futuros

Para que um sistema utilizado com fins educacionais seja efetivo e contextualizado no trabalho de professores e alunos, não basta que sejam privilegiados somente aspectos técnicos, mas, principalmente, sua adequação pedagógica ao contexto em que se insere, considerando as características individuais do aluno e a instrução individualizada, de forma a facilitar a criação de estruturas conceituais e metodológicas adequadas à capacidade e interesse deste aluno. Um sistema de diagnóstico contribui exatamente nesta direção, pois diagnosticando o comportamento do aluno no sistema é possível adaptar as tarefas de forma individualizada.

O diagnóstico, nesse sentido, é considerado um componente indispensável em ambientes de ensino, o que não é uma tarefa trivial. Normalmente, os sistemas de diagnóstico baseiam-se em catálogos de erros, que são difíceis de implementar e aplicáveis somente a domínios específicos. Além disso, o fato de o aluno ser um agente ativo no processo de diagnóstico, dificulta a identificação do conhecimento defeituoso na base de conhecimento deste.

Baseado no trabalho de Koning [KON 2000] (descrito no capítulo 4) e na idéia de Self [SEL 93] de que o diagnóstico deve caminhar para um *framework* padrão para a área de diagnóstico, este trabalho descreve o projeto e a implementação de um sistema de diagnóstico baseado em modelo, adaptativo e genérico, capaz de diagnosticar o comportamento do aluno em cursos de EAD (o sistema gera o diagnóstico independente do domínio, permitindo que um único procedimento seja aplicado a vários cursos). Para isto, utilizou-se o *TÆMS* (descrito na seção 5.1), um *framework* modelador de tarefas, independente do domínio, e o escalonador *DTC* (descrito na seção 5.2), que gera os caminhos de ações possíveis para atingir um determinado objetivo e escolhe o melhor entre eles.

No sistema proposto, o professor, inicialmente, cria um material específico para disponibilizar na *web*, que será descrito em estruturas de tarefas *TÆMS*, sobre as quais são criados os planos (estes formam a biblioteca geral de planos do sistema, que pode conter planos para qualquer domínio possível). O aluno faz suas escolhas de acordo com seu objetivo (o sistema oferece duas formas de restrição: restrição de tempo ou restrição de conteúdo) e gera um plano, que é uma nova estrutura de tarefas. Esta é interpretada pelo *DTC* que retorna um ou mais escalonamentos para o conjunto de tarefas. Baseado nesse escalonamento de tarefas, o sistema sugere ao agente um "roteiro de estudo" (indicando as páginas *web* que devem ser exploradas). Enquanto o agente está navegando no material e executando as tarefas necessárias para atingir seu objetivo, informações referentes à navegação vão sendo gravadas em um *log*. O processo de diagnóstico ocorre quando as informações gravadas no *log* são confrontadas com o escalonamento gerado pelo *DTC*. Tal comparação é baseada em um modelo causal geral (descrito na seção 6.1.4.1) que pode ser utilizado para diagnosticar diferenças entre quaisquer estruturas *TÆMS*. Se forem detectadas divergências no processo de diagnóstico, dependendo do tipo de sintoma, o sistema criará e incluirá um novo plano na biblioteca de planos *para este aluno*.

Conclui-se que o sistema implementado atendeu às expectativas iniciais. Os resultados atingidos e as diversas publicações obtidas com esse trabalho (descritas no Anexo 4), constituem-se num indicador importante das possibilidades de aplicação do

sistema proposto. Entretanto, algumas limitações se apresentam e necessitam de um estudo mais amplo.

As limitações existentes hoje no sistema proposto são as seguintes:

- não há uma ferramenta com interface amigável para o professor gerar as estruturas de tarefas *TÆMS*;
- depois de gerado o plano, o aluno deve segui-lo até o final para que o diagnóstico seja gerado corretamente, pois não há a possibilidade de parar e retornar ao mesmo plano, a menos que o processo seja reiniciado;
- o diagnóstico só é gerado se o aluno clicar no botão "Finalizar";
- o diagnóstico é gerado e armazenado em um arquivo texto, o que dificulta a manipulação das informações.

Essas funcionalidades não foram implementadas devido ao tempo para a elaboração deste trabalho. Mas, certamente, todas trariam grandes contribuições se implementadas, por isso, são descritas com mais detalhes na próxima seção.

## 8.1 Trabalhos futuros

As estruturas de tarefas *TÆMS*, utilizadas pelo sistema para gerar os planos, são criadas através de uma descrição textual (veja o exemplo de uma descrição textual no Anexo 1), o que é extremamente trabalhoso e complicado. Uma forma de resolver esse problema (talvez a mais simples) seria criar um editor com uma interface amigável, em que o professor pudesse criar estas estruturas informando o nome da tarefa, quais suas subtarefas, seus métodos e a distribuição de qualidade, custo e duração, qual tarefa facilita qual e qual tarefa é pré-requisito de qual. Uma forma mais eficiente seria criar uma ferramenta em que, através das páginas *web* geradas pelo professor, gerasse automaticamente as estruturas de tarefas necessárias para gerar os planos do sistema.

A segunda limitação descrita anteriormente é mais fácil de ser resolvida. No sistema implementado, depois de gerado o plano, o aluno deve segui-lo até o final para que o diagnóstico seja gerado corretamente. Caso o aluno interrompa a execução do plano com o objetivo de, mais tarde, retornar ao ponto em que havia parado, o sistema gerará um diagnóstico incorreto e não permitirá que o aluno acesse o mesmo plano novamente. Neste caso, é preciso criar uma função que permita ao aluno interromper a execução do plano sempre que desejar, de forma que isto não interfira no diagnóstico.

O fato de o diagnóstico só ser gerado depois de o aluno ter executado todo o plano e ter clicado no botão "Finalizar" resulta na geração de um diagnóstico tardio, ou seja, enquanto o aluno está navegando nas páginas *web* do curso, o professor não tem nenhuma informação sobre seu comportamento. Uma solução seria ir gerando diagnósticos parciais e, no final, gerar um diagnóstico geral.

A última limitação é sobre a forma como o diagnóstico está sendo armazenado. No sistema implementado, para cada plano gerado pelo aluno é criado um arquivo texto com o respectivo diagnóstico. Estas informações serão bem mais úteis se forem armazenadas no banco de dados do sistema. A manipulação dos dados é simplificada e várias operações podem ser executadas sobre estes.

# Anexo 1 – Estrutura de tarefas *TÆMS*

```
;Computação Simbólica e Numérica

(spec_agent
        (label agent_A))

(spec_task_group
        (label inf5513)
        (agent agent_A)
        (subtasks teoria exemplo exercicio)
        (qaf q_sum) )

(spec_task
        (label teoria)
        (agent agent_A)
        (subtasks mc_1 zf_1 se_1 ip_1 af_1 ig_1
dn_1 ms_1)
        (supertasks inf5513)
        (qaf q_sum) )

(spec_task
        (label mc_1)
        (agent agent_A)
        (subtasks mc_11 mc_12 mc_13 mc_14)
        (supertasks teoria)
        (qaf q_sum) )

(spec_method
        (label mc_11)
        (agent agent_A)
        (supertasks mc_1)
        (outcomes
                (mc_11
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label mc_12)
        (agent agent_A)
        (supertasks mc_1)
        (outcomes
                (mc_12
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 35  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label mc_13)
        (agent agent_A)
        (supertasks mc_1)
        (outcomes
                (mc_13
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 30  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label mc_14)
        (agent agent_A)
        (supertasks mc_1)
        (outcomes
                (mc_14
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 25  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label zf_1)
        (agent agent_A)
        (subtasks zf_11 zf_12 zf_13 zf_14 zf_15
zf_16 zf_17)
        (supertasks teoria)
        (qaf q_sum) )

(spec_method
        (label zf_11)
        (agent agent_A)
        (supertasks zf_1)
        (outcomes
                (zf_11
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution  5  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label zf_12)
        (agent agent_A)
        (supertasks zf_1)
        (outcomes
                (zf_12
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label zf_13)
        (agent agent_A)
        (supertasks zf_1)
        (outcomes
                (zf_13
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label zf_14)
        (agent agent_A)
        (supertasks zf_1)
        (outcomes
                (zf_14
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 20  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label zf_15)
        (agent agent_A)
        (supertasks zf_1)
        (outcomes
                (zf_15
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 20  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label zf_16)
        (agent agent_A)
        (supertasks zf_1)
        (outcomes
                (zf_16
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label zf_17)
        (agent agent_A)
        (supertasks zf_1)
        (outcomes
                (zf_17
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 30  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label se_1)
        (agent agent_A)
        (subtasks se_11 se_12g se_13)
        (supertasks teoria)
        (qaf q_sum) )

(spec_task
        (label se_12g)
        (agent agent_A)
        (subtasks se_12 se_121g se_122g se_123g)
        (supertasks se_1)
```

```
        (qaf q_sum) )

(spec_task
        (label se_121g)
        (agent agent_A)
        (subtasks se_121 se_1211g se_1212g)
        (supertasks se_12g)
        (qaf q_sum) )

(spec_task
        (label se_122g)
        (agent agent_A)
        (subtasks se_122 se_1221 se_1222)
        (supertasks se_12g)
        (qaf q_sum) )

(spec_task
        (label se_123g)
        (agent agent_A)
        (subtasks se_123 se_1231 se_1232 se_1233)
        (supertasks se_12g)
        (qaf q_sum) )

(spec_task
        (label se_1211g)
        (agent agent_A)
        (subtasks se_1211 se_12111 se_12112)
        (supertasks se_121g)
        (qaf q_sum) )

(spec_task
        (label se_1212g)
        (agent agent_A)
        (subtasks se_1212 se_12121 se_12122)
        (supertasks se_121g)
        (qaf q_sum) )

(spec_method
        (label se_11)
        (agent agent_A)
        (supertasks se_1)
        (outcomes
                (se_11
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution  5  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_13)
        (agent agent_A)
        (supertasks se_1)
        (outcomes
                (se_13
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 90  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_12)
        (agent agent_A)
        (supertasks se_12g)
        (outcomes
                (se_12
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_121)
        (agent agent_A)
        (supertasks se_121g)
        (outcomes
                (se_121
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution  5  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_122)
        (agent agent_A)
        (supertasks se_122g)
        (outcomes
                (se_122
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution  5  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_1221)
        (agent agent_A)
        (supertasks se_122g)
        (outcomes
                (se_1221
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 25  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_1222)
        (agent agent_A)
        (supertasks se_122g)
        (outcomes
                (se_1222
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 25  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_123)
        (agent agent_A)
        (supertasks se_123g)
        (outcomes
                (se_123
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution  5  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_1231)
        (agent agent_A)
        (supertasks se_123g)
        (outcomes
                (se_1231
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_1232)
        (agent agent_A)
        (supertasks se_123g)
        (outcomes
                (se_1232
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_1233)
        (agent agent_A)
        (supertasks se_123g)
        (outcomes
                (se_1233
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution  5  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_1211)
        (agent agent_A)
        (supertasks se_1211g)
        (outcomes
                (se_1211
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution  5  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_12111)
        (agent agent_A)
```

```
        (supertasks se_1211g)
        (outcomes
                (se_12111
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 30  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_12112)
        (agent agent_A)
        (supertasks se_1211g)
        (outcomes
                (se_12112
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 30  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_1212)
        (agent agent_A)
        (supertasks se_1212g)
        (outcomes
                (se_1212
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution  5  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_12121)
        (agent agent_A)
        (supertasks se_1212g)
        (outcomes
                (se_12121
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 25  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_12122)
        (agent agent_A)
        (supertasks se_1212g)
        (outcomes
                (se_12122
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 20  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label ip_1)
        (agent agent_A)
        (subtasks ip_11 ip_12g ip_13 ip_14)
        (supertasks teoria)
        (qaf q_sum) )

(spec_task
        (label ip_12g)
        (agent agent_A)
        (subtasks ip_12 ip_121 ip_122 ip_123)
        (supertasks ip_1)
        (qaf q_sum) )

(spec_method
        (label ip_11)
        (agent agent_A)
        (supertasks ip_1)
        (outcomes
                (ip_11
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution  5  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ip_13)
        (agent agent_A)
        (supertasks ip_1)
        (outcomes
                (ip_13
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ip_14)
        (agent agent_A)
        (supertasks ip_1)
        (outcomes
                (ip_14
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ip_12)
        (agent agent_A)
        (supertasks ip_12g)
        (outcomes
                (ip_12
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution  5  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ip_121)
        (agent agent_A)
        (supertasks ip_12g)
        (outcomes
                (ip_121
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ip_122)
        (agent agent_A)
        (supertasks ip_12g)
        (outcomes
                (ip_122
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ip_123)
        (agent agent_A)
        (supertasks ip_12g)
        (outcomes
                (ip_123
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label af_1)
        (agent agent_A)
        (subtasks af_11 af_12)
        (supertasks teoria)
        (qaf q_sum) )

(spec_method
        (label af_11)
        (agent agent_A)
        (supertasks af_1)
        (outcomes
                (af_11
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution  5  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label af_12)
        (agent agent_A)
        (supertasks af_1)
        (outcomes
                (af_12
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 30  1.0)
                  (cost_distribution  0  1.0) )))
```

```
(spec_task
        (label ig_1)
        (agent agent_A)
        (subtasks ig_11 ig_12 ig_13g ig_14)
        (supertasks teoria)
        (qaf q_sum) )

(spec_task
        (label ig_13g)
        (agent agent_A)
        (subtasks ig_13 ig_131 ig_132 ig_133)
        (supertasks ig_1)
        (qaf q_sum) )

(spec_method
        (label ig_11)
        (agent agent_A)
        (supertasks ig_1)
        (outcomes
                (ig_11
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution  5  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ig_12)
        (agent agent_A)
        (supertasks ig_1)
        (outcomes
                (ig_12
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ig_14)
        (agent agent_A)
        (supertasks ig_1)
        (outcomes
                (ig_14
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 20  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ig_13)
        (agent agent_A)
        (supertasks ig_13g)
        (outcomes
                (ig_13
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution  5  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ig_131)
        (agent agent_A)
        (supertasks ig_13g)
        (outcomes
                (ig_131
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ig_132)
        (agent agent_A)
        (supertasks ig_13g)
        (outcomes
                (ig_132
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ig_133)
        (agent agent_A)
        (supertasks ig_13g)
        (outcomes
                (ig_133
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label dn_1)
        (agent agent_A)
        (subtasks dn_11 dn_12 dn_13)
        (supertasks teoria)
        (qaf q_sum) )

(spec_method
        (label dn_11)
        (agent agent_A)
        (supertasks dn_1)
        (outcomes
                (dn_11
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label dn_12)
        (agent agent_A)
        (supertasks dn_1)
        (outcomes
                (dn_12
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label dn_13)
        (agent agent_A)
        (supertasks dn_1)
        (outcomes
                (dn_13
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label ms_1)
        (agent agent_A)
        (subtasks ms_11 ms_12 ms_13)
        (supertasks teoria)
        (qaf q_sum) )

(spec_method
        (label ms_11)
        (agent agent_A)
        (supertasks ms_1)
        (outcomes
                (ms_11
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ms_12)
        (agent agent_A)
        (supertasks ms_1)
        (outcomes
                (ms_12
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 30  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ms_13)
        (agent agent_A)
        (supertasks ms_1)
        (outcomes
                (ms_13
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 30  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
```

```
        (label exemplo)
        (agent agent_A)
        (subtasks mc_2 zf_2 se_2 ip_2 af_2 ig_2
dn_2 ms_2)
        (supertasks inf5513)
        (qaf q_sum) )

(spec_task
        (label mc_2)
        (agent agent_A)
        (subtasks mc_21 mc_22)
        (supertasks exemplo)
        (qaf q_sum) )

(spec_method
        (label mc_21)
        (agent agent_A)
        (supertasks mc_2)
        (outcomes
                (mc_21
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label mc_22)
        (agent agent_A)
        (supertasks mc_2)
        (outcomes
                (mc_22
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution  5  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label zf_2)
        (agent agent_A)
        (subtasks zf_21 zf_22 zf_23 zf_24)
        (supertasks exemplo)
        (qaf q_sum) )

(spec_method
        (label zf_21)
        (agent agent_A)
        (supertasks zf_2)
        (outcomes
                (zf_21
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label zf_22)
        (agent agent_A)
        (supertasks zf_2)
        (outcomes
                (zf_22
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label zf_23)
        (agent agent_A)
        (supertasks zf_2)
        (outcomes
                (zf_23
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label zf_24)
        (agent agent_A)
        (supertasks zf_2)
        (outcomes
                (zf_24
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label se_2)
        (agent agent_A)
        (subtasks se_21 se_22)
        (supertasks exemplo)
        (qaf q_sum) )

(spec_method
        (label se_21)
        (agent agent_A)
        (supertasks se_2)
        (outcomes
                (se_21
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 25  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_22)
        (agent agent_A)
        (supertasks se_2)
        (outcomes
                (se_22
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 25  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label ip_2)
        (agent agent_A)
        (subtasks ip_21 ip_22)
        (supertasks exemplo)
        (qaf q_sum) )

(spec_method
        (label ip_21)
        (agent agent_A)
        (supertasks ip_2)
        (outcomes
                (ip_21
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ip_22)
        (agent agent_A)
        (supertasks ip_2)
        (outcomes
                (ip_22
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label af_2)
        (agent agent_A)
        (subtasks af_21)
        (supertasks exemplo)
        (qaf q_sum) )

(spec_method
        (label af_21)
        (agent agent_A)
        (supertasks af_2)
        (outcomes
                (af_21
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 25  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label ig_2)
        (agent agent_A)
        (subtasks ig_21 ig_22g ig_23)
        (supertasks exemplo)
        (qaf q_sum) )

(spec_task
        (label ig_22g)
```

```
        (agent agent_A)
        (subtasks ig_22 ig_221 ig_222 ig_223
ig_224)
        (supertasks ig_2)
        (qaf q_sum) )

(spec_method
        (label ig_21)
        (agent agent_A)
        (supertasks ig_2)
        (outcomes
                (ig_21
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ig_23)
        (agent agent_A)
        (supertasks ig_2)
        (outcomes
                (ig_23
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ig_22)
        (agent agent_A)
        (supertasks ig_22g)
        (outcomes
                (ig_22
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ig_221)
        (agent agent_A)
        (supertasks ig_22g)
        (outcomes
                (ig_221
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 20  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ig_222)
        (agent agent_A)
        (supertasks ig_22g)
        (outcomes
                (ig_222
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ig_223)
        (agent agent_A)
        (supertasks ig_22g)
        (outcomes
                (ig_223
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ig_224)
        (agent agent_A)
        (supertasks ig_22g)
        (outcomes
                (ig_224
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 20  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label dn_2)
        (agent agent_A)
        (subtasks dn_21 dn_22 dn_23 dn_24)
        (supertasks exemplo)
        (qaf q_sum) )

(spec_method
        (label dn_21)
        (agent agent_A)
        (supertasks dn_2)
        (outcomes
                (dn_21
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label dn_22)
        (agent agent_A)
        (supertasks dn_2)
        (outcomes
                (dn_22
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 10  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label dn_23)
        (agent agent_A)
        (supertasks dn_2)
        (outcomes
                (dn_23
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label dn_24)
        (agent agent_A)
        (supertasks dn_2)
        (outcomes
                (dn_24
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 15  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label ms_2)
        (agent agent_A)
        (subtasks ms_21)
        (supertasks exemplo)
        (qaf q_sum) )

(spec_method
        (label ms_21)
        (agent agent_A)
        (supertasks ms_2)
        (outcomes
                (ms_21
                  (density 1.0)
                  (quality_distribution 100  1.0)
                  (duration_distribution 20  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label exercicio)
        (agent agent_A)
        (subtasks mc_3 zf_3 se_3 ip_3 af_3 ig_3
dn_3 ms_3)
        (supertasks inf5513)
        (qaf q_sum) )

(spec_task
        (label mc_3)
        (agent agent_A)
        (subtasks mc_31 mc_32)
        (supertasks exercicio)
        (qaf q_sum) )

(spec_method
        (label mc_31)
        (agent agent_A)
        (supertasks mc_3)
        (outcomes
```

```
                (mc_31
                  (density 1.0)
                  (quality_distribution  90  1.0)
                  (duration_distribution 60  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label mc_32)
        (agent agent_A)
        (supertasks mc_3)
        (outcomes
                (mc_32
                  (density 1.0)
                  (quality_distribution  90  1.0)
                  (duration_distribution 60  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label zf_3)
        (agent agent_A)
        (subtasks zf_31 zf_32 zf_33 zf_34)
        (supertasks exercicio)
        (qaf q_sum) )

(spec_method
        (label zf_31)
        (agent agent_A)
        (supertasks zf_3)
        (outcomes
                (zf_31
                  (density 1.0)
                  (quality_distribution  85  1.0)
                  (duration_distribution 50  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label zf_32)
        (agent agent_A)
        (supertasks zf_3)
        (outcomes
                (zf_32
                  (density 1.0)
                  (quality_distribution  85  1.0)
                  (duration_distribution 60  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label zf_33)
        (agent agent_A)
        (supertasks zf_3)
        (outcomes
                (zf_33
                  (density 1.0)
                  (quality_distribution  80  1.0)
                  (duration_distribution 60  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label zf_34)
        (agent agent_A)
        (supertasks zf_3)
        (outcomes
                (zf_34
                  (density 1.0)
                  (quality_distribution  80  1.0)
                  (duration_distribution 70  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label se_3)
        (agent agent_A)
        (subtasks se_31 se_32)
        (supertasks exercicio)
        (qaf q_sum) )

(spec_method
        (label se_31)
        (agent agent_A)
        (supertasks se_3)
        (outcomes
                (se_31
                  (density 1.0)
                  (quality_distribution  75  1.0)
                  (duration_distribution 60  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label se_32)
        (agent agent_A)
        (supertasks se_3)
        (outcomes
                (se_32
                  (density 1.0)
                  (quality_distribution  70  1.0)
                  (duration_distribution 60  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label ip_3)
        (agent agent_A)
        (subtasks ip_31 ip_32 ip_33)
        (supertasks exercicio)
        (qaf q_sum) )

(spec_method
        (label ip_31)
        (agent agent_A)
        (supertasks ip_3)
        (outcomes
                (ip_31
                  (density 1.0)
                  (quality_distribution  90  1.0)
                  (duration_distribution 60  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ip_32)
        (agent agent_A)
        (supertasks ip_3)
        (outcomes
                (ip_32
                  (density 1.0)
                  (quality_distribution  90  1.0)
                  (duration_distribution 50  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ip_33)
        (agent agent_A)
        (supertasks ip_3)
        (outcomes
                (ip_33
                  (density 1.0)
                  (quality_distribution  90  1.0)
                  (duration_distribution 50  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label af_3)
        (agent agent_A)
        (subtasks af_31)
        (supertasks exercicio)
        (qaf q_sum) )

(spec_method
        (label af_31)
        (agent agent_A)
        (supertasks af_3)
        (outcomes
                (af_31
                  (density 1.0)
                  (quality_distribution  80  1.0)
                  (duration_distribution 50  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label ig_3)
        (agent agent_A)
        (subtasks ig_31 ig_32 ig_33)
        (supertasks exercicio)
        (qaf q_sum) )

(spec_method
        (label ig_31)
        (agent agent_A)
        (supertasks ig_3)
        (outcomes
                (ig_31
                  (density 1.0)
                  (quality_distribution  80  1.0)
                  (duration_distribution 90  1.0)
                  (cost_distribution  0  1.0) )))
```

```
(spec_method
        (label ig_32)
        (agent agent_A)
        (supertasks ig_3)
        (outcomes
                (ig_32
                  (density 1.0)
                  (quality_distribution  90  1.0)
                  (duration_distribution 90  1.0)
                  (cost_distribution  0  1.0) )))

(spec_method
        (label ig_33)
        (agent agent_A)
        (supertasks ig_3)
        (outcomes
                (ig_33
                  (density 1.0)
                  (quality_distribution  90  1.0)
                  (duration_distribution 90  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label dn_3)
        (agent agent_A)
        (subtasks dn_31 dn_32)
        (supertasks exercicio)
        (qaf q_sum) )

(spec_method
        (label dn_31)
        (agent agent_A)
        (supertasks dn_3)
        (outcomes
                (dn_31
                  (density 1.0)
                  (quality_distribution   80  1.0)
                  (duration_distribution 100  1.0)
                  (cost_distribution  0   1.0) )))

(spec_method
        (label dn_32)
        (agent agent_A)
        (supertasks dn_3)
        (outcomes
                (dn_32
                  (density 1.0)
                  (quality_distribution   80  1.0)
                  (duration_distribution 100  1.0)
                  (cost_distribution  0  1.0) )))

(spec_task
        (label ms_3)
        (agent agent_A)
        (subtasks ms_31)
        (supertasks exercicio)
        (qaf q_sum) )

(spec_method
        (label ms_31)
        (agent agent_A)
        (supertasks ms_3)
        (outcomes
                (ms_31
                  (density 1.0)
                  (quality_distribution  90  1.0)
                  (duration_distribution 30  1.0)
                  (cost_distribution  0  1.0) )))

(spec_facilitates
        (label facil_1)
        (agent agent_A)
        (from mc_1)
        (to mc_2)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_2)
        (agent agent_A)
        (from mc_2)
        (to mc_3)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_3)
        (agent agent_A)
        (from zf_1)
        (to zf_2)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_4)
        (agent agent_A)
        (from zf_2)
        (to zf_3)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_5)
        (agent agent_A)
        (from se_1)
        (to se_2)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_6)
        (agent agent_A)
        (from se_2)
        (to se_3)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_7)
        (agent agent_A)
        (from ip_1)
        (to ip_2)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_8)
        (agent agent_A)
        (from ip_2)
        (to ip_3)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_9)
        (agent agent_A)
        (from af_1)
        (to af_2)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_10)
        (agent agent_A)
        (from af_2)
        (to af_3)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_11)
```

```
        (agent agent_A)
        (from ig_1)
        (to ig_2)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_12)
        (agent agent_A)
        (from ig_2)
        (to ig_3)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_13)
        (agent agent_A)
        (from dn_1)
        (to dn_2)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_14)
        (agent agent_A)
        (from dn_2)
        (to dn_3)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_15)
        (agent agent_A)
        (from ms_1)
        (to ms_2)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_16)
        (agent agent_A)
        (from ms_2)
        (to ms_3)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_17)
        (agent agent_A)
        (from mc_11)
        (to mc_12)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_18)
        (agent agent_A)
        (from mc_11)
        (to mc_13)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_19)
        (agent agent_A)
        (from mc_11)
        (to mc_14)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_20)
        (agent agent_A)
        (from mc_12)
        (to mc_14)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_21)
        (agent agent_A)
        (from mc_13)
        (to ig_14)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_22)
        (agent agent_A)
        (from mc_13)
        (to ip_123)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_23)
        (agent agent_A)
        (from zf_11)
        (to zf_12)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_24)
        (agent agent_A)
        (from zf_11)
        (to zf_13)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_25)
        (agent agent_A)
        (from zf_11)
        (to zf_14)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_26)
        (agent agent_A)
        (from zf_11)
        (to zf_15)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_27)
        (agent agent_A)
        (from zf_11)
        (to zf_16)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_28)
        (agent agent_A)
        (from zf_12)
```

```
        (to zf_13)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_29)
        (agent agent_A)
        (from se_11)
        (to se_12)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_30)
        (agent agent_A)
        (from se_11)
        (to se_13)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_31)
        (agent agent_A)
        (from se_12)
        (to se_121)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_32)
        (agent agent_A)
        (from se_12)
        (to se_122)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_33)
        (agent agent_A)
        (from ip_11)
        (to ip_12)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_34)
        (agent agent_A)
        (from ip_11)
        (to ip_13)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_35)
        (agent agent_A)
        (from ip_11)
        (to ip_14)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_36)
        (agent agent_A)
        (from ip_121)
        (to ip_123)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_37)
        (agent agent_A)
        (from ip_122)
        (to ip_123)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_38)
        (agent agent_A)
        (from ip_1)
        (to af_1)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_39)
        (agent agent_A)
        (from ip_2)
        (to af_2)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_40)
        (agent agent_A)
        (from ip_3)
        (to af_3)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_41)
        (agent agent_A)
        (from af_11)
        (to af_12)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_43)
        (agent agent_A)
        (from ig_11)
        (to ig_12)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_facilitates
        (label facil_44)
        (agent agent_A)
        (from ig_11)
        (to ig_13)
        (quality_power 100 1.0)
        (duration_power 0 1.0)
        (cost_power 0 1.0)
        (delay 0 1.0) )

(spec_enables
        (label enable_1)
        (agent agent_A)
        (from mc_1)
        (to mc_3)
        (delay 0 1.0) )

(spec_enables
        (label enable_2)
        (agent agent_A)
        (from zf_1)
        (to zf_3)
        (delay 0 1.0) )

(spec_enables
        (label enable_3)
```

```
        (agent agent_A)
        (from se_1)
        (to se_3)
        (delay 0 1.0) )

(spec_enables
        (label enable_4)
        (agent agent_A)
        (from ip_1)
        (to ip_3)
        (delay 0 1.0) )

(spec_enables
        (label enable_5)
        (agent agent_A)
        (from af_1)
        (to af_3)
        (delay 0 1.0) )

(spec_enables
        (label enable_6)
        (agent agent_A)
        (from ig_1)
        (to ig_3)
        (delay 0 1.0) )

(spec_enables
        (label enable_7)
        (agent agent_A)
        (from dn_1)
        (to dn_3)
        (delay 0 1.0) )

(spec_enables
        (label enable_8)
        (agent agent_A)
        (from ms_1)
        (to ms_3)
        (delay 0 1.0) )

(spec_enables
        (label enable_9)
        (agent agent_A)
        (from zf_12)
        (to zf_17)
        (delay 0 1.0) )

(spec_enables
        (label enable_10)
        (agent agent_A)
        (from zf_13)
        (to zf_17)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_11)
        (agent agent_A)
        (from zf_14)
        (to zf_17)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_12)
        (agent agent_A)
        (from zf_15)
        (to zf_17)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_13)
        (agent agent_A)
        (from zf_16)
        (to zf_17)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_14)
        (agent agent_A)
        (from zf_23)
        (to se_13)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_15)
        (agent agent_A)
        (from se_12)
        (to se_13)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_16)
        (agent agent_A)
        (from se_122)
        (to se_123)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_17)
        (agent agent_A)
        (from se_12111)
        (to se_12112)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_18)
        (agent agent_A)
        (from se_1221)
        (to se_1222)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_19)
        (agent agent_A)
        (from se_1221)
        (to af_12)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_20)
        (agent agent_A)
        (from ip_12)
        (to ip_14)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_21)
        (agent agent_A)
        (from ig_131)
        (to ig_132)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_22)
        (agent agent_A)
        (from ig_132)
        (to ig_133)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_23)
        (agent agent_A)
        (from ig_132)
        (to ig_14)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_24)
        (agent agent_A)
        (from dn_11)
        (to dn_12)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_25)
        (agent agent_A)
        (from dn_11)
        (to dn_13)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_26)
        (agent agent_A)
        (from ms_11)
        (to ms_12)
        (delay 0.0 1.0) )

(spec_enables
        (label enable_27)
        (agent agent_A)
        (from ms_11)
        (to ms_13)
        (delay 0.0 1.0) )
```

# Anexo 2 – Tabelas do BD previamente preenchidas

# Anexo 3 – Manual de instalação do sistema DIACOM

Este manual contém instruções sobre a instalação do DIACOM e alguns cuidados que devem ser tomados quando as estruturas de tarefas *TÆMS* forem criadas.

## 1 – Instalação

Os procedimentos da instalação do DIACOM serão descritos a seguir.

- instalar o JDK 1.3, ou mais atual:
  1. para fins de documentação foi utilizada a versão: 1.3.1_01
  2. assumimos que o mesmo está instalado no diretório /usr/java/jdk1.3

- instalar o suporte a JDBC para MySQL:
  1. foi instalado o mm.mysql versao 2.0.8 (instalado em /usr/local/mm.mysql-2.0.8)

- instalar o suporte a Servlet para o JDK:
  1. para fins de documentação foi instalada a versão: servlet-2_3-fcs
  2. o mesmo foi copiado para /usr/java/jdk1.3 (o diretorio javax)

- instalar tomcat + apache (sugerimos o uso dos RPMS de ambos os produtos):
  1. assumimos que o tomcat está instalado no diretório /var/tomcat;
  2. assumimos que o apache está instalado no diretório /etc/httpd;
  3. para fins de documentação estamos utilizando as seguintes versões:
     3.1. tomcat: 3.2.1-1;
     3.2. apache: 1.3.18 (Unix).

- instalar o MySQL (sugerimos o uso de RPMS):
  1. assumimos que o MySQL está instalado no diretório /var/lib/mysql;
  2. para fins de documentação estamos utilizando a versão 3.23.36;

- escolher portas, distintas, para os servidores:
  1. sugerimos 8080 para o apache e 8082 para o tomcat (essas serão as portas utilizadas nesta documentação);

- configurar o tomcat para operar na porta escolhida:
  1. editar o arquivo /var/tomcat/conf/server.xml e altera-lo para a seguinte configuração:

```
<Connector className="org.apache.tomcat.service.PoolTcpConnector">
<Parameter name="handler"
 value="org.apache.tomcat.service.http.HttpConnectionHandler"/>
<Parameter name="port" value="8082"/>
</Connector>
```

- configurar o apache para operar na porta 8080:
  1. editar o arquivo /etc/httpd/conf/httpd.conf: `Listen 8080`

- configurar o script de inicialização para adicionar o CLASSPATH adequado:
  1. editar o arquivo /etc/rc.d/init.d/tomcat
  2. alterá-lo para:

```
export PATH=$PATH:/usr/java/jdk1.3/bin:/usr/java/jre1.3/bin
export JAVA_HOME=/usr/java/jdk1.3
export TOMCAT_HOME=/var/tomcat
export CLASSPATH="/usr/java/jdk1.3:/usr/local/mm.mysql-2.0.8/
mm.mysql-2.0.8-bin.jar"
```

- configurar as classes para operarem nas portas adequadas, observando o endereço correto do servidor

- criar o banco *taems* no MySQL, de acordo com as especificações;

- configurar o usuário de acesso ao banco, especificando senha, permissões e hosts:
  1. as entradas devem ser semelhantes a:
     1.1. tabela user:

| Host | % |
|---|---|
| User | vanessal |
| Password | 18dd01667cf4766a |
| Select_priv | N |
| Insert_priv | N |
| Update_priv | N |
| Delete_priv | N |
| Create_priv | N |
| Drop_priv | N |
| Reload_priv | N |
| Shutdown_priv | N |
| Process_priv | N |
| File_priv | N |
| Grant_priv | N |
| References_priv | N |
| Index_priv | N |
| Alter_priv | N |

1.2. tabela db:

| Host | % |
|---|---|
| Db | taems |
| User | vanessal |
| Select_priv | Y |
| Insert_priv | Y |
| Update_priv | Y |
| Delete_priv | Y |
| Create_priv | Y |
| Drop_priv | Y |
| Grant_priv | Y |
| References_priv | Y |
| Index_priv | Y |
| Alter_priv | Y |

- atribuir as permissões adequados para acesso ao aplicativo DTC
- colocar as classes em /var/tomcat/webapps/taems/WEB-INF/classes
- colocar as páginas em /home/httpd/htdocs/inf5513
- adicionar o alias no apache:
  1. editar o arquivo /etc/httpd/conf/httpd.conf (adicionar ao final do arquivo):
  `Alias /inf5513 /home/httpd/htdocs/inf5513`
- verificar se as classes e as páginas html estão apontando para a porta e endereço certos.

## 2 – Observações sobre a criação das estruturas de tarefas *TÆMS*

Como não há uma ferramenta para editar as estruturas de tarefas *TÆMS*, estas são criadas através de uma descrição textual. O manual completo sobre como criar estas estruturas de forma textual, está disponível em <http://mas.cs.umass.edu/research/taems/white>. Porém, neste trabalho foram adotados algumas regras para que o sistema consiga interpretar as estruturas sem problemas, descritas a seguir.

A primeira linha do arquivo da descrição textual da tarefa deverá conter o nome do plano, ou seja, o nome do curso ao qual a estrutura de tarefas se refere. Este é o nome que irá aparecer na lista do segundo formulário do sistema, onde o usuário escolhe o curso que irá estudar. Esta informação deve ser escrita na forma de um comentário, por isso, antes do nome do plano, usa-se um ponto-e-vírgula (por exemplo: `;Computação Simbólica e Numérica`).

Outra observação importante, é que sempre depois da descrição de uma tarefa, de um método ou de um relacionamento, há pelo menos dois fecha-parênteses, que devem estar separados por um espaço (por exemplo, observe como deve ficar a última linha da descrição de uma tarefa: `(qaf q_sum) )`).

O arquivo que contém a estrutura de tarefas deve ser do tipo texto, gravado com extensão `ttaems`. Durante a digitação, sugere-se não usar a tecla <tab> ou qualquer outro tipo de tabulação (a indentação deve ser feita somente com a barra de espaços).

# Anexo 4 – Relação de publicações

- "Um Estudo sobre Diagnóstico: do Diagnóstico de Falhas em Dispositivos Físicos ao Diagnóstico Cognitivo Baseado em Modelo" – Vanessa Lindemann, Trabalho Individual 961 – PPGC – UFRGS, dezembro de 2000 – Porto Alegre/RS (47p.).

- "Um estudo sobre Diagnóstico Baseado em Modelo para análise do comportamento do Aluno em Sistemas de Ensino" – Vanessa Lindemann, publicado nos anais do I Congresso Brasileiro de Computação – CBComp 2001, promovido pela UNIVALI – Universidade do Vale do Itajaí e CTTMar – Centro de Ciências Tecnológicas da Terra e do Mar, de 20 a 24 de agosto de 2001 – Itajaí/SC (p.311-322).

- "Utilização de uma Ferramenta Independente do Domínio para Diagnóstico do Comportamento do Aluno" – Vanessa Lindemann e Ana L.C. Bazzan, publicado nos anais do WORKCOMP'2001 – Workshop de Computação, realizado em São José dos Campos – SP nos dias 08 e 09 de outubro de 2001, promovido pelo Instituto Tecnológico de Aeronáutica – Divisão de Ciência da Computação (p 1-7).

- "Utilização de uma Ferramenta Independente do Domínio para Diagnóstico do Comportamento do Aluno" – Vanessa Lindemann e Ana L.C. Bazzan, publicado nos anais do XIII Simpósio Brasileiro de informática na Educação, realizado no período de 21 a 23 de novembro de 2001, promovido pela Sociedade Brasileira de Computação – Comissão Especial de Informática na Educação (p.503-511).

- "A Domain-Independent Diagnosis Tool to Adapt Organizations in Learning Scenarios" – Ana L. C. Bazzan, Vanessa Lindemann e Victor Lesser, aceito no AAMAS 2002 – Autonomous Agents and Multi-Agent Systems, Bologna, Italy, que será realizado de 15 a 19 de julho em Bologna, Italy.

# Bibliografia


[ABR 80] ABRAMOVICI, M.; BREUER, M. Multiple Faults Diagnosis in Combinatorial Circuits Base on Effect-Cause Analysis. **IEEE Transactions on Computers**, New York, v.C-29, n.6, p.451-460, June 1980.

[APA 2001] APACHE Software Foundation. Disponível em: <http://www.apache.org>. Acesso em: out. 2001.

[BAZ 92] BAZZAN, A.L.C. **Um Sistema Baseado em Conhecimento para Fechamento de Balanço de Massa na Produção de Celulose**. 1992. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[BAZ 93] BAZZAN, A.L.C. Métodos de Cálculo de Diagnóstico de Falhas em Dispositivos Físicos. In: VICCARI, R.M. et al. **Diagnóstico Inteligente**. Porto Alegre: Instituto de Informática, UFRGS, 1993. p.12-36. Relatório de Pesquisa.

[BAZ 98] BAZZAN, A.L.C.; LESSER, V.; XUAN, P. **Adapting an Organization Design through Domain-Independent Diagnosis**. Amherst: University of Massachusetts, 1998. Technical Report.

[BER 94] BENJAMINS, R; JANSWEIJER, W. Toward a Competence Theory of Diagnosis. **IEEE Expert – Intelligent Systems & Their Applications**, Los Alamitos, v.9, n.5, p.43-52, October 1994.

[BER 95] BENJAMINS, R. Problem-Solving Methods for Diagnosis and their Role in Knowledge Acquisition. **International Journal of Expert Systems**, [S.l.], v.8, n.2, p. 93-120, 1995.

[BEB 99] BENYO, B.; LESSER, V.R. **Evolving Organizational Designs for Multi-Agent Systems**. Amherst: University of Massachusetts, 1999. (Technical Report 99-00).

[DAV 84] DAVIS, R. Diagnostic Reasoning Based on Structure and Behavior. **Artificial Intelligence**, Amsterdam, v.24, n.1-3, p.347-410, December 1984.

[DEC 94] DECKER, K.; LESSER, V. **Quantitative Modeling of Complex Environments**. Amherst: University of Massachusetts, 1994. (Technical Report 93-21).

[DEC 95] DECKER, K.; LESSER, V.R. Designing a Family of Coordination Algorithms. In: INTERNATIONAL CONFERENCE ON MULTI-AGENT SYSTEMS, 1., 1995. **Proceedings...** San Francisco, CA: AAAI Press, 1995.

[DEC 98] DECKER, K.; LI, J. Coordinated Hospital Patient Scheduling. In: INTERNATIONAL CONFERENCE ON MULTI-AGENT SYSTEMS, 3., ICMAS, 1998, Paris, France. **Proceedings…** Paris, France: IEEE Computer Society, 1998.

[DEI 99] DEITEL, H.M.; DEITEL, P.J. **Java: How to Program**. 3.ed. New Jersey: Prentice Hall, 1999.

[DEK 87] De KLEER, J.; WILLIAMS, B. Diagnosis Multiple Faults. **Artificial Intelligence**, Amsterdam, v.32, n.1, p.97-130, April 1987.

[FEA 97] FEATHER, S. **JavaScript em Exemplos**. São Paulo: Makron Books, 1997.

[FRA 93] FRANCO, S.M.L. **Representação e Diagnóstico de Falsas Concepções em um Tutor Inteligente**. 1993. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[GEN 84] GENESERETH, M. The Use of Design Descriptions in Automated Diagnosis. **Artificial Intelligence**, Amsterdam, v.24, n.1-3, p.411-436, December 1984.

[HOR 9-a] HORLING, B.; BENYO, B. LESSER, V. **Using Self-Diagnosis to Adapt Organizational Structures**. Amherst: University of Massachusetts, [199-]. Disponível em: <http://mas.cs.umass.edu/index. Shtml >. Acesso em: mar.2001.

[HOR 9-b] HORLING, B.; LESSER, V.; VICENT, R. **Multi-Agent System Simulation Framework**. Amherst: University of Massachusetts, [199-]. Disponível em: <http://mas.cs.umass.edu/index.shtml>. Acesso em: mar.2001.

[HOR 2000] HORLING, B. et al. Diagnosis as an Integral Part of Multi-Agent Adaptability. In: DARPA Information Survivability Conference And Exposition, 2000, South Carolina. **Proceedings...** South Carolina: IEEE Computer Society, 2000.

[HOR 2001] HORLING, B. et al. Distributed Sensor Network for Real-Time Tracking. In: AUTONOMOUS AGENT, 2001. **Proceedings...** [S.l.: s.n.], 2001.

[JAV 2001] JAVA TM 2 Plataform, Standard Edition, v.1.31 API Specification. Disponível em: <http://java.sun.com/j2se/1.3/docs/api/index.html>. Acesso em: set. 2001.

[KON 98] KONING, K.; BREDEWEG, B. Qualitative Reasoning in Tutoring Interactions. **Interactive Learning Environments**, Amsterdam, v.5, p.65-79, 1998.

[KON 2000] KONING, K. et al. Model-Based Reasoning About Learner Behaviour. **Artificial Intelligence**, Amsterdam, n.117, p.173-229, 2000.

[LEM 99] LEMAY, L.; CADENHEAD, R. **Aprenda em 21 Dias Java 2**. Rio de Janeiro: Campus, 1999.

[LES 98] LESSER, V. et al. **A Multi-Agent System for Intelligent Environment Control**. Amherst: University of Massachusetts, 1998. (Technical Report 98-40).

[LES 2000] LESSER, V. et al. **The Taems White Paper**. Disponível em: <http://mas.cs.umass.edu/research/taems/white>. Acesso em: jun. 2000.

[LIN 2000] LINDEMANN, V. **Um Estudo sobre Diagnóstico: do Diagnóstico de Falhas em Dispositivos Físicos ao Diagnóstico Cognitivo Baseado em Modelo**. 2000. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[MAS 2000] MASLAKOWSKI, M.; FIGUEIREDO, J. **Aprenda em 21 Dias MySQL**. Rio de Janeiro: Campus, 2000.

[MUL 2001] MULTI-AGENT Systems Laboratory. Disponível em: <http://mas. cs.umass.edu/index.shtml. Acesso em: jan. 2001.

[OHL 87] OHLSSON, S. Some Principles of Intelligent Tutoring. **Artificial Intelligence and Education**, Norwood, n.1, p.203-237, 1987.

[PIP 86] PIPITONE, F. The FIS Eletronics Troubleshooting System. **Computer**, New York, v.19, n.7, p.68-77, July 1986.

[REI 87] REITER, R. A Theory of Diagnosis from First Principles. **Artificial Intelligence**, Amsterdam, v.32, n.1, p.57-95, April 1987.

[ROT 66] ROTH, J.P. Diagnosis of Automata Failures: a Calculus and a Method. **IBM Journal of Research and Development**, New York, v.10, n.4, p.278-291, July 1966.

[SAV 97] SAVOLA, T.; WESTENBROEK, A.; HECK, J. **Usando HTML**: o guia de referência mais completo. Rio de Janeiro: Campus, 1997.

[SEL 93] SELF, J. Model-Based Cognitive Diagnosis. **User Modeling and User-Adapted Interaction**, Germany, v.3, p.89-106, 1993.

[SOU 2001] SOUTO, M.A.M. et al. Ferramentas de Suporte a Monitoração do Aluno em um Ambiente Inteligente de Ensino na Web. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, SBIE, 2001, Vitória/ES, Brasil. **Anais...** Vitória/ES, Brasil: Crediné Silva de Menezes, Davidson Cury, Orivaldo de Lira Tavares, 2001.

[TAR 2001] TAROUCO, L.M.R. **Teleducação - introdução**. Disponível em: <http://penta.ufrgs.br/edu/teleduc/teleduc1..htm>. Acesso em: mar. 2001.

[VIC 93] VICCARI, R.M. et al. **Diagnóstico Inteligente**. Porto Alegre: Instituto de Informática, UFRGS, 1993, p.12-36. Relatório de Pesquisa.

[VIC 96] VICCARI, R.M.; GIRAFFA, L.M.M. Sistemas Tutores Inteligentes: abordagem tradicional x abordagem de agentes. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, SBIA, 13., 1996. **Anais…** Curitiba: CEFET-PR, 1996.

[WAG 2001] WAGNER, T.; HORLING, B. **The Struggle for Reuse and Domain Independence: Research with TÆMS, DTC and JAF**. In: WORKSHOP ON INFRASTRUCTURE FOR AGENTS, MAS, AND SCALABLE MAS, AAAI, 2., 2001, Montreal, Canada. **Proceedings…** Montreal, Canadá: [s.n], 2001. Disponível em: <http://mas.cs.umass.edu/index.shtml>. Acesso em: ago. 2001.

[ZHA 99] ZHANG, X. et al. **Integrating High-Level and Detailed Agent Coordination into a Layered Architecture**. Amherst: University of Massachusetts, 1999. Disponível em: <http://mas.cs.umass.edu/index.shtml >. Acesso em: out. 2000.