UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA DA COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

A. I. A. T. D.

AMBIENTE INTEGRADO DE APOIO À TOMADA DE DECISÃO

por : Eng. Waldomiro P. D. C. Loyolla

orientador: Prof. Dr. Manuel de Jesus Mendes

/92

Este Exemplar corresponde à redação
final da Tese defendida por Waldomiro Loyolla
e aprovada pela Comissão Julgadora em 18.12.92

Orientador

Tese apresentada à Faculdade de Engenharia Elétrica FEE - UNICAMP como parte dos requisitos exigidos para a obtenção do título de DOUTOR EM ENGENHARIA.

Novembro de 1992

O meu louvor a

DEUS

pela vida,

pelos meus queridos,

pela bênção de aqui chegar.

Dedico este trabalho:
à minha querida esposa Sandra,
às minhas filhas Pollyana e Sharon,
aos meus pais Ziliah e Reduzina.

Este trabalho contou com o apoio financeiro da CAPES / PICD

AGRADECIMENTOS

Ao Prof. Mendes, pelo contínuo suporte e direção através deste desafio, mas acima de tudo por, além de Orientador, revelar-se como Amigo.

Ao Prof. Fernando Gomide, pela co-orientação do trabalho, pelas direções e discussões em todos os artigos, e pela grande amizade demonstrada.

Ao Prof. Rafael, pela direção nas discussões da especificação formal.

Ao amigo Alberto, pelos muitos dias de trabalho conjunto e pelo desenvolvimento do SMAC-SIM.

Ao amigo Brás, pelo companheirismo, ao longo do trabalho com as especificações formais.

Ao amigo Edwin, pela confiança e ajuda, enquanto chefe do departamento.

À UNESP, pelo afastamento que permitiu o desenvolvimento deste trabalho.

À UNICAMP, por haver dado a oportunidade de realizar este trabalho.

E a todos aqueles que colaboraram, direta ou indiretamente, para o desenvolvimento deste trabalho.

RESUMO

Este trabalho descreve o desenvolvimento e implementação de um Ambiente Integrado de Apoio à Tomada de Decisão. O mesmo destina-se à análise de sistemas a eventos discretos, oferecendo ferramentas para a realização de modelagem, simulações, previsões de comportamento e inferências lógicas, de maneira integrada e cooperativa. Apresenta, como principal característica, a integração funcional e operacional de um Simulador a Eventos Discretos com um Processador de Conhecimento, tornando possível a utilização integrada de conhecimento estático, proveniente do Processador de Conhecimento, e de conhecimento dinâmico, proveniente do Simulador.

A utilização integrada destes tipos de conhecimento fez-se possível devido ao desenvolvimento de um Modelo Abstrato de Informação, contribuição central deste trabalho. Destacam-se, como principais características deste modelo, o ser: configurável pelo usuário, determinando a forma de apresentação da informação; incremental, facilitando a inclusão de novas informações; flexível, permitindo seu uso para diferentes propósitos; hierárquico, permitindo a composição da informação em vários níveis.

A abordagem utilizada no desenvolvimento do modelo difere, em parte, da metodologia tradicional de modelagem orientada a objetos, visto combinar uma forma flexível de representação do conhecimento, composta por frames, regras e procedimentos, com as vantagens de uma programação simbólica e declarativa, mediante o uso da linguagem Prolog. Devido a isto, a representação da informação pode ser expressa tanto de maneira informal declarativa, quase coloquial, como de maneira formal, através da Lógica Temporal de Tempo Real Generalizada. Uma metodologia para a transformação da representação formal do conhecimento em declarativa também foi desenvolvida. Uma lógica de simulação foi especialmente desenvolvida, para usufruir das facilidades advindas do modelo de informação e da integração da manipulação do conhecimento, oferecida pelo Processador de Conhecimento.

A arquitetura do ambiente é modular, baseada em elementos funcionais projetados segundo o paradigma de objetos, permitindo uma adequação ao uso em configuração distribuída e multiusuário, podendo operar como um Servidor de Apoio à Tomada de Decisão.

A utilização do ambiente é amigável ao usuário, dispondo de sistemas tutores especialistas para a aquisição e análise de conhecimento que, por facilitar e orientar o usuário em sua operação, permitem seu uso diretamente pelos profissionais responsáveis pela tomada de decisão.

ABSTRACT

This work describes the developing and implementation of an Integrated Decision-Making Support Environment for Discrete-Event System analysis. The decision-making support is provided by a set of integrated and cooperative tools for modeling, simulation, inference, and system behavior forecast. Its main feature is to provide functional and operational integration for both a Discrete-Event Simulator and a Knowledge Processor, allowing the integration of dynamic knowledge, coming from simulations, with static knowledge coming from inferences of Knowledge Processor.

The integrated use of these two knowledge classes was only possible with the development of an Abstract Information Model, the central contribution of this work. The main characteristics of this model are: user configurability, determining the information features; incrementability, allowing simple new information inclusion; flexibility, using the same model for several purposes; hierarchic, using several levels for information description.

The approach herein proposed, for information model development, differs from traditional object-oriented methodologies because it combines symbolic and declarative programming capabilities, from Prolog, with flexible schemes of knowledge representation, combining frames, rules, and procedures structures. Thus, information can be presented both in an informal declarative (quasi-coloquial) way, and in a formal way, using Generalized Real Time Temporal Logic. A methodology for translating one representation to another has been developed. Also, a special dedicated Simulation Logic has been developed for using the Abstract Information Model by Simulator, without previous manipulation.

The environment architecture is modular, based on functional elements designed by using object paradigm. This allows to adapt the environment to multiuser, distributed configuration, working as a Decision-Making Support Server.

The environment is also user-friendly, with tutoring expert-systems helping knowledge acquisition and analysis. So, the environment can be used by technical or managerial personel, with little or no assistance.

ÍNDICE

PÁGINA

4 - ASPECTOS DE IMPLEMENTAÇÃO

5 - APLICAÇÕES

6 - CONCLUSÕES

7 - REFERÊNCIAS BIBLIOGRÁFICAS

CAPÍTULO 1 - INTRODUÇÃO

1.1 - MOTIVAÇÃO

A crescente competição no mercado mundial da manufatura tem requerido importantes desenvolvimentos na arte de tomada de decisão. No âmbito da Manufatura Integrada por Computador (CIM), a tomada de decisão requer a inferência sobre sistemas extremamente complexos, frequentemente compostos por equipamentos e processos de alto custo.

A grande complexidade desses sistemas integrados de manufatura tem motivado o desenvolvimento de diferentes técnicas de apoio à tomada de decisão, uma vez que métodos tradicionais de projeto e de análise de sistemas têm se mostrado inadequados ao estudo de suas complexas interações e comportamento dinâmico.

A simulação tem-se apresentado como uma importante ferramenta no apoio à tomada de decisão, no âmbito de sistemas integrados de manufatura [3, 25, 36, 91, 107, 151, 169]. Isto se deve, principalmente, à possibilidade de realização de uma avaliação abstrata antecipada do comportamento dinâmico de um sistema produtivo. Pode-se, assim, possibilitar a poupança de muito dinheiro, tempo e recursos em diversos setores, tais como no projeto, implantação, planejamento, operação e controle de sistemas automatizados da manufatura.

O desenvolvimento de Sistemas de Simulação [1, 59, 137, 154], inclusive com alguns deles sendo tradicionalmente denominados linguagens de simulação, tem levado à adoção da simulação como uma das mais importantes ferramentas de análise gerencial e operacional, no âmbito da manufatura automatizada. A disponibilidade de potentes interfaces homem-máquina com poderosos recursos gráficos, como animação, geração automática de código de simulação e modelagem gráfica por ícones [11, 39, 71, 105, 144, 149, 165], têm contribuído para difundir a utilização da simulação no ambiente fabril.

Apesar de disporem dessas facilidades, as linguagens de simulação apresentam sérias restrições à sua utilização diretamente por profissionais de projeto ou operação de sistemas flexíveis de manufatura. Essas dificuldades são motivadas, principalmente, por requererem do modelador uma considerável experiência em modelagem para simulação, além de conhecimento da sintaxe da linguagem escolhida. Assim o uso eficiente da simulação tem requerido consideráveis volumes de recursos humanos, financeiros, de equipamentos e de tempo, já que a formação de pessoal especializado em modelagem e simulação tem-se

apresentado como um processo demorado e dispendioso.

Dificuldades surgem também na modelagem de sistemas de médio e grande porte, onde existe a necessidade de adaptação do comportamento da manufatura à estratégia de abordagem do universo, adotada em cada linguagem de simulação. Deve-se isto ao fato que os atuais sistemas de simulação, como concebidos, são derivados das tradicionais linguagens de programação. Isto fez com que tanto suas estruturas de dados, como seu comportamento procedimental de execução apresentassem um caráter fortemente voltado para funções matemáticas. Assim a composição dos modelos de simulação deve seguir rigorosas estruturas de dados, frequentemente não apresentando a flexibilidade necessária à modelagem de sistemas muito complexos, ou heurística, em sua descrição.

Outro problema a ser considerado refere-se ao fato de que a estrutura empresarial moderna encontra-se consideravelmente compartimentada, distinguindo-se algumas áreas decisórias, como produção, estoque, finanças e marketing. No entanto, todas estas áreas devem trabalhar cooperativamente para atingir-se a meta global da manufatura, qual seja, o funcionamento de toda a estrutura produtiva de forma a ter-se uma produção com qualidade e custo que lhe permitam obter o lucro planejado [138]. Note-se aí que o requerido lucro, em termos corporativos, nem sempre corresponde à soma do máximo lucro requerido de cada área, sugerindo, então, estudos de sua interdependência em função das metas ou espectativas gerenciais [94].

Dada esta complexidade técnica e organizacional da manufatura atual, na maioria das vezes, um mesmo sistema de simulação não se mostra adequado a atender diferentes áreas de interesse, exigindo o uso de diversos sistemas de simulação e consequente necessidade de dispor-se de especialistas em cada sistema utilizado.

A isto agrega-se a necessidade de estabelecer-se uma linguagem comum para a comunicação entre os responsáveis pela geração de uma base de informações através da simulação e aqueles responsáveis pelo processo decisório propriamente dito. Frequentemente esta comunicação torna-se quase tão difícil quanto a descrição ou o próprio entendimento de todo o sistema produtivo, em toda sua complexidade.

Considerados estes fatores, pode-se notar o motivo pelo qual o processo de tomada de decisão, no ambiente da manufatura, tem sido tão compartimentado. Apesar do já grande e crescente uso do computador nesses ambientes, tem sido, principalmente, utilizado como um executor de complexas funções ou tarefas distintas. Sua operação, ainda, tem-se realizado com baixo nível de integra-

ção, quando se tratando do interrelacionamento entre diferente áreas organizacionais da manufatura.

O uso de princípios de Inteligência Artificial (AI) no ambiente da manufatura tem sido largamente proposto e aplicado, no sentido de simplificar-se o uso da simulação, tanto nas tarefas de modelagem como nas de análise [156, 157], permitindo um tratamento computacional do conhecimento heurístico, bem como outras simplificações.

Sistemas Especialistas baseados em conhecimento podem ser utilizados como geradores de informação, a partir da realização de inferências. No entanto, a formatação da informação a ser utilizada nas inferências, geralmente é restrita ao domínio de informações para o qual foi desenvolvido o sistema. Isto dificulta a utilização dessas informações de forma integrada com outros sistemas.

Muito ainda está por ser desenvolvido, no sentido de utilizar-se, em forma cooperativa, os resultados provenientes de sistemas simuladores e de sistemas especialistas, de forma prática e simples, no ambiente industrial.

1.2 - A TESE

O tema central da Tese diz respeito às seguintes partes:

> É possível desenvolver-se um Ambiente Integrado de Apoio à Tomada de Decisão, suportado por duas ferramentas básicas de análise: um simulador a eventos discretos, e um processador configurável de conhecimento.
>
> A operação destas ferramentas realiza-se cooperativamente, sobre um mesmo modelo de informação, a ser composto tanto de maneira declarativa informal, como de maneira formal.

O projeto de tal ambiente pode desenvolver-se de forma top-down, seguindo a metodologia sugerida pelo paradigma de objetos, tendo como principal meta a simplificação do processo de modelagem para simulação, considerado um dado público alvo.

Tem-se como público alvo, usuário deste sistema, os profissionais de projeto, implantação, planejamento, operação, controle e de diferentes níveis de gerência, em sistemas de manufatura integrada por computador.

A representação usada para os modelos de simulação deve ser fácilmente inteligível por diferentes usuários, em diferentes níveis funcionais do sistema de manufatura, quer a modelagem tenha sido realizada de maneira formal ou informal. Tal representação deve ser flexível o bastante de forma a permitir a visualização de diferentes níveis de detalhamento dos modelos em estudo. Espera-se, também, a possibilidade de se dispor de reusabilidade de modelos de sistemas de manufatura, bem como facilidade e simplicidade na realização de modificações dos modelos, a qualquer nível de detalhamento.

Para implementar-se tais características, deve-se subdividir a sequência lógica de trabalho em distintos objetos de análise, de forma que possam vir a ser utilizados de maneira configurável pelo usuário do sistema.

Adota-se uma abstração na descrição da manufatura que leva a uma modelagem modular e hierarquizada. Isto permite que um mesmo modelo para simulação seja usado, tanto para uma análise global e macroscópica, como para uma análise detalhista e microscópica, quando do estudo comportamental de uma mesma unidade de manufatura.

Como consequência da característica descritiva adotada para o modelo abstrato de simulação, é possível realizar a simulação, levando-se em consideração diferentes tipos de informações. Estas informações, tanto podem ser

técnicas, utilizadas a nível de análise operacional, como informações administrativas e comerciais, úteis a nível de análise financeira, comercial ou gerencial.

A arquitetura proposta para o ambiente apresenta um novo enfoque em relação à modelagem de sistemas flexíveis de manufatura, simplificando significativamente a operação cooperativa entre simulação e processamento de conhecimento.

A adoção de procedimentos de inteligência artificial permite o tratamento computacional do conhecimento heurístico contido na descrição de regras e fatos que caracterizam a condicionalidade da maior parte da heurística existente em sistemas de manufatura. Procura-se assim poder-se adquirir, apresentar e processar o conhecimento em forma quase coloquial.

A estruturação modular adotada, tanto para a arquitetura do simulador, como para o processador de conhecimento e para a modelagem de entidades da manufatura, permite a integração do sistema, e mais, que este trabalhe de forma distribuída, sobre conhecimento armazenado de forma também distribuída, o que o caracteriza como um servidor de apoio à tomada de decisão.

1.3 - Sequenciamento do Trabalho.

No capítulo 2, discutem-se as características dos Sistemas de Manufatura Integrada por Computador. Apresenta-se também um breve comentário sobre as ferramentas de apoio computacional para estes sistemas. Em particular, comenta-se a necessidade por integração e padronização da apresentação dos conhecimentos nas diferentes áreas da manufatura.

No capítulo 3, apresentam-se os requisitos impostos para o desenvolvimento do ambiente integrado proposto. A seguir, é discutido o Modelo Abstrato de Informação, a ser utilizado como forma de apresentação de conhecimento. Por fim, considera-se o desenvolvimento do modelo funcional do ambiente, descrevendo-se a funcionalidade e composição de cada elemento funcional.

No capítulo 4, são apresentados aspectos de implementação do ambiente. Discute-se a composição implementada para a representação declarativa e para a representação formal do Modelo Abstrato de Informação. Aspectos da transformação de especificações de sistemas de uma representação para a outra também são discutidos.

No capítulo 5, são apresentados alguns exemplos de modelagem de sistemas tanto de maneira formal, como declarativa, com o propósito de realização de simulações.

Finalmente, no capítulo 6, apresenta-se uma revisão dos principais resultados e um resumo da contribuição realizada com este trabalho de pesquisa. Conclui-se com uma discussão de futuros trabalhos a serem realizados como sequência deste.

CAPÍTULO 2 - A TOMADA DE DECISÃO E A MANUFATURA INTEGRADA POR COMPUTADOR

2.1 - A Manufatura Integrada por Computador

Diferentes fatores têm caracterizado o desenvolvimento dos ambientes de manufatura nos últimos tempos. Os grandes avanços tecnológicos em diferentes áreas do conhecimento, ocorridos em um passado recente, juntamente com as pressões sócio-econômicas, oriundas da nova ordem econômica mundial, apresentaram-se como alguns dos principais fatores responsáveis pelo atual nível de modernização atingido pela manufatura [5].

O advento do computador, como ferramenta frequente no ambiente de manufatura, provocou, inicialmente, o aparecimento de ilhas de automação, passando posteriormente a dar suporte à implantação de Sistemas Flexíveis de Manufatura (FMS), culminando por tornar-se no elemento catalisador do desenvolvimento do conceito da Manufatura Integrada por Computador (CIM) [52, 114, 141, 158].

A Manufatura Integrada por Computador representa muito mais do que uma expressão da vastidão de possibilidades de uso do computador no ambiente da manufatura. Ela corresponde, essencialmente, à formulação de um conceito representativo de um sistema produtivo. Neste sistema, todos os tipos de informações, materiais, atividades, equipamentos e processos compõem-se num todo, automatizado, otimizado e principalmente integrado pelo computador [114, 115].

De uma forma genérica, um Sistema de Manufatura Integrada por Computador (CIM) pode ser descrito como composto pelos seguintes elementos organizacionais [61]: Engenharia Auxiliada por Computador (CAE), Planejamento Auxiliado por Computador (CAP), Manufatura Auxiliada por Computador (CAM). Estes elementos são representados na figura 2.1.

Há que se notar o fato de que, na Manufatura Integrada por Computador, o fluxo de informações ocorre entre todos os elementos organizacionais, enquanto o fluxo de materiais desenvolve-se apenas internamente ao sistema de manufatura, culminando por gerar o produto acabado.

O ciclo produtivo inicia-se, geralmente, a partir da reunião de dois tipos de informações: solicitações de produto advindas do mercado e idealização de novos produtos a serem desenvolvidos. Estas informações são, inicialmente, tratadas pelo sistema de informações comerciais e depois enviadas para

o desenvolvimento pela engenharia de produto.

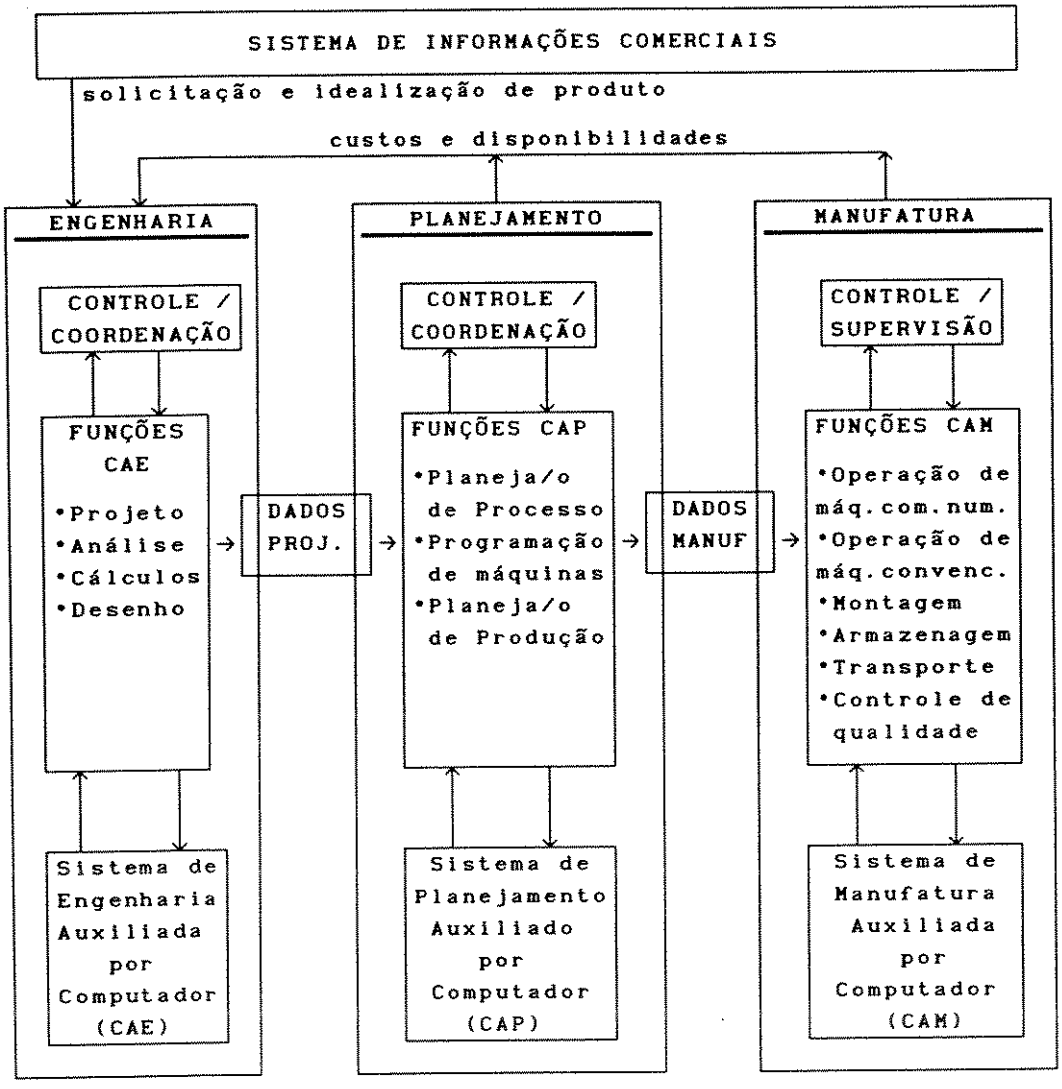


Fig. 2.1 - Elementos Organizacionais da Manufatura Integrada por Computador

SISTEMA DE ENGENHARIA AUXILIADA POR COMPUTADOR

Um sistema de Engenharia Auxiliada por Computador (CAE) consiste na utilização de hardware e software gráfico, de forma a criar imagens, manipular estas imagens, desenvolver cálculos e análises de projeto [50], além de proceder ao desenho de partes e peças nas mais variadas perspectivas.

O projeto de um produto corresponde ao desenvolvimento do conjunto dos projetos de suas partes componentes. Para cada uma destas partes determina-se

sua forma, dimensão, tolerância, etc. O projeto de cada parte desenvolve-se em quatro distintas fases: projeto preliminar, refinamento, análise e implementação [30].

No projeto preliminar, consideram-se fatores globais como forma, material, requisitos funcionais, etc., estabelecendo-se uma base inicial de informações a respeito do produto. No refinamento, tais fatores são detalhados. Importante papel, no projeto de partes e produtos, é desempenhado pela Tecnologia de Grupo, com as devidas caracterizações e agrupamentos de similaridades geométricas, de acordo com adequações a padrões previamente estabelecidos [58].

Na fase de análise, avalia-se o detalhamento de projeto realizado na fase de refinamento, em conjunto com as considerações de ferramentas e fixações necessárias. Nesta fase, alterações e aperfeiçoamentos são considerados, podendo levar a um reprojeto das partes. Na fase de implementação, o projeto final é documentado, provendo-se o necessário detalhamento.

SISTEMA DE PLANEJAMENTO AUXILIADO POR COMPUTADOR

Um Sistema de Planejamento Auxiliado por Computador (CAP) é utilizado para o desenvolvimento de: Planejamento de Processos, Programação de Máquinas e Planejamento de Produção.

a) Planejamento de Processos

O Planejamento de Processos determina as diferentes fases de produção das partes de um produto. Neste plano, são estabelecidos as rotas, operações, máquinas e ferramentas necessárias para o processamento de cada parte, de cada produto, de forma econômica, eficiente e com a qualidade requerida [22]. Para este planejamento, são considerados, ainda, fatores como custo de produção, tempo de manufatura, tempo inativo de máquina, de forma a minimizar os custos e maximizar o volume de produção e a qualidade.

A utilização da Tecnologia de Grupo durante o projeto de partes e produtos, em muito facilita o estabelecimento de um planejamento do processo, restringindo o universo de possibilidades a serem consideradas [140].

b) Programação de Máquinas

A Programação corresponde ao desenvolvimento de programas para o processamento de máquinas convencionais e de comando numérico, de sistemas de manu-

seio, armazenagem e de recuperação de materiais.

A programação de máquinas de comando numérico corresponde à elaboração de um conjunto de instruções para estas máquinas operatrizes. Estas instruções incluem a sequência de operação, passos de ferramentas, velocidade de processamento, etc. Esta programação pode ser provida pelo computador, a partir da base de informações de projeto de produto.

Um sistema de controle numérico por computador (CNC) supervisiona o controle lógico, o processamento de dados geométricos e a execução do programa de controle numérico da máquina. Um sistema de Controle Numérico Distribuído (DNC) supervisiona, de forma integrada, um conjunto de máquinas de comando numérico, com a ajuda de um computador [142].

A programação de sistemas de manuseio, armazenagem e recuperação de materiais, corresponde ao estabelecimento de rotinas de trabalho como: trajetórias para elementos de movimentação; sequências de operação para elementos de fixação; e determinação de formas e locais de armazenagem de materiais, partes, e ferramentas, a serem utilizados em conjunto com os elementos de fixação e movimentação.

c) Planejamento de Produção

O Planejamento de Produção corresponde ao estabelecimento de níveis e sequências de produção para um determinado intervalo de tempo [50]. Este planejamento desenvolve-se em duas fases: planejamento de necessidades de materiais (MRP) e planejamento de carga e sequenciamento de máquinas.

O planejamento de necessidades de materiais define as solicitações de materiais necessários para a produção de partes com um tempo prévio, de forma a assegurar sua disponibilidade à época de utilização. Este planejamento é realizado a partir de informações a respeito do tipo e quantidade das partes a serem produzidas e dos tipos e quantidades dos materiais especificados para produzi-las.

Baseando-se nas necessidades estimadas de materiais, pode-se estabelecer as necessidades de recursos de maquinários para produção como: horas-máquina, horas de trabalho, etc.

Integrando-se as informações sobre planejamento de necessidades de materiais, planejamento de necessidades de recursos de maquinários e o planejamento de sequenciamento global de produção, estabelece-se um sistema de Planejamento de Necessidades de Materiais em Malha Fechada (closed loop MRP). A

realimentação de informações, necessária à implantação deste sistema, permite um real acompanhamento e controle dos programas de produção.

A disponibilidade de integração de um conjunto mais abrangente de informações, como planejamento de necessidades de materiais, planejamento de necessidades de recursos de maquinários, planejamento de sequenciamento de produção e planejamento comercial, leva ao MRP II, denominado Planejamento de Recursos da Manufatura [68].

O tratamento conjunto e integrado destas informações exige um alto volume de processamento, além de certas decisões intermediárias, no sentido de restringir-se o número de variáveis a serem consideradas. É importante a restrição do número de variáveis a serem consideradas, porque, de outra forma seria praticamente impossível atingerem-se conclusões específicas, devido à explosão combinatorial de alternativas para o planejamento integrado.

O planejamento de carga de máquinas corresponde ao direcionamento de partes às diferentes máquinas e unidades de manufatura, de forma a distribuir a carga de produção. O planejamento de sequenciamento de máquinas determina a sequência em que as máquinas desenvolvem suas operações e o volume de trabalho direcionado a cada uma. Esta é uma tarefa consideravelmente complexa, onde se deve levar em consideração, de forma interdependente, fatores como: capacidade de processamento de cada máquina, tempo de processamento de cada parte em cada máquina, prazos, etc. Frequentemente o planejamento de sequenciamento deve ser reconsiderado, em face de contingências.

SISTEMA DE MANUFATURA AUXILIADA POR COMPUTADOR

Um Sistema de Manufatura Auxiliada por Computador (CAM) envolve as fases de operação, controle de operação, manutenção e controle de qualidade de: máquinas de comando numérico, máquinas convencionais, sistemas de armazenagem e transporte de materiais e ferramentas, sistemas de montagem.

a) Operação

Como operação, entende-se todo processamento produtivo de transformação ou montagem, realizada em chão de fábrica.

b) Controle de Operação

O controle compreende, inicialmente, o acompanhamento da operação do sistema de manufatura frente ao planejamento de produção e à programação. A

partir desse acompanhamento, define-se o conjunto de atitudes corretivas, frente a desvios sofridos em relação ao planejamento e à programação.

Este controle é desenvolvido sobre todas as partes envolvidas no processamento, em diferentes níveis hierárquicos como: máquina operatriz, célula flexível de manufatura (FMC), sistema flexível de manufatura (FMS) e mesmo a nível de fábrica. O acompanhamento, em todos os níveis, é importante, porque, com a utilização de um planejamento integrado de recursos, procura-se estabelecer uma meta ótima global e não apenas uma meta ótima localizada a qualquer nível.

c) Manutenção

A Manutenção envolve o diagnóstico e correção de mau funcionamento de quaisquer das partes que compõem o processamento.

A execução de manutenção em equipamento produtivo afeta os planos estabelecidos nos planejamentos realizados. As consequências de parada de equipamentos para a realização de manutenção podem ser desastrosas para os planejamentos de produção, se não consideradas previamente. Assim sendo, um rigoroso controle de manutenção deve ser mantido, fornecendo informações necessárias, quando da realização dos planejamentos.

Quebras de equipamentos podem ocorrer, mesmo que minimizada sua ocorrência por um programa eficiente de manutenção. Estas ocorrências devem ser estimadas e encaminhadas para devidas alterações nos planos já estabelecidos.

e) Controle de Qualidade

O controle de qualidade corresponde ao acompanhamento e avaliação dos resultados de produção, frente a medidas, tolerâncias e outros parâmetros estabelecidos em projeto.

O controle de qualidade, quando efetuado com a ajuda de computador, permite uma tomada de decisão muito mais rápida e eficiente. Isto se traduz em minimização de perdas e maximização de eficiência e lucro do sistema produtivo.

Informações provenientes de um controle de qualidade auxiliado por computador podem ser rápidamente utilizadas, tanto na atualização de programação de máquinas, como nos planejamentos de produção, ou até mesmo no projeto de partes e produtos.

Apesar de todas as possíveis vantagens, advindas de uma manufatura totalmente integrada por computador, esta não foi atingida de forma completa [115]. Isto se deve, principalmente, a dois motivos:

- maior facilidade de aplicar-se a automação e integração em pequenas partes do sistema produtivo, ao invés de proceder-se a uma integração completa,
- existência de grande número de problemas tecnológicos ainda por resolver, principalmente no que se refere à falta de desenvolvimento de padrões totais de informações, bem como padrões completos para transferência de dados que possam ser utilizados por todas as áreas da manufatura.

2.2 - As Ferramentas de apoio à Manufatura

A integração de tão grande diversidade e volume de informações existentes na Manufatura Integrada por Computador, leva à exigência de grandes sistemas computacionais de apoio. Estes sistemas computacionais auxiliam, não só na geração e manipulação das informações, como também na compreensão e avaliação destas.

O estudo setorizado da manufatura tem levado à geração de muitas ferramentas computacionais de apoio, para uso nas suas mais diferentes áreas.

Na área de engenharia auxiliada por computador, mais particularmente em projeto auxiliado por computador, inúmeros sistemas computacionais foram desenvolvidos. Muito comum tem sido a utilização de softwares de apoio ao projeto, dotados de processamento gráfico tridimensional, desenho, detalhamento e documentação de projeto, como o AUTOCAD.

O desenvolvimento de programas consideravelmente abrangentes, em várias das fases de projeto, também tem sido muito grande. Um que se enquadra, nesse caso, é o PROMPT [125], que apresenta um acompanhamento documental de todas as fases do projeto. Fornece uma atualização automática das características intrínsecas do projeto diretamente para uma biblioteca, criada especificamente para acompanhamento, utilizando-se de uma heurística própria para a análise de cada peça ou parte, ao longo do projeto.

Muitos outros sistemas computacionais foram criados, com propósitos mais específicos [63, 81, 121, 176], como projeto de transportadores, laminadores, manipuladores, etc.

Sistemas computacionais mais complexos foram desenvolvidos para o auxílio ao projeto de sistemas de manufatura, como o apresentado por Eloranta e outros em [33]. Nele, é apresentado o sistema CENT, que utiliza ferramentas de multimídia no desenvolvimento de reprojeto de partes e peças. O conhecimento proveniente de outras situações de projeto é armazenado em uma base, onde informações orientativas podem ser buscadas.

Outros sistemas computacionais são específicos para a fase de análise de projeto de células de manufatura, de forma a estimar seu desempenho em função de dados de projeto e de políticas de planejamento de recursos [74, 112].

No entanto, devido à grande multiplicidade destas ferramentas, estudos continuam sendo feitos para o desenvolvimento de tecnologia para a devida integração de software de apoio a projeto com todas as outras áreas da manufa-

tura integrada [17, 33].

Na área de planejamento de processos da manufatura a quantidade de software de apoio desenvolvido também tem sido muito grande [13, 24, 72, 85, 87, 126, 147], cada um detalhando o planejamento de processo para diferentes grupos de partes e peças.

Alguns dos sistemas desenvolvidos nesta área provêem uma razoável interligação com as informações de projeto, otimizando o desenvolvimento do planejamento [69, 133], enquanto outros permitem uma integração de informações do planejamento de processo com a área de processamento [13, 93, 166, 175].

De qualquer forma, o grande avanço, ainda a ser atingido, corresponde à interligação completa entre informações dos três elementos organizacionais da Manufatura Integrada por Computador, principalmente acrescentando-se a variável tempo como parâmetro a ser considerado para aprimoramento de certos processos [80].

É na área de Manufatura Auxiliada por Computador que se encontra a maior parte do software de apoio desenvolvido nos últimos tempos. Estes dedicam-se, geralmente, a realizar, de forma dedicada, tarefas inerentes às fases que compõem a manufatura auxiliada por computador.

Para a programação de máquinas, geralmente tem-se um software proprietário, para trabalho específico com uma determinada máquina ou célula de manufatura [144].

Quanto ao planejamento de produção, tem sido a área de manufatura onde se tem desenvolvido, no total, o maior número de software de apoio, talvez por apresentar um relacionamento muito estreito com a produtividade, e consequentemente com o lucro.

Além dos métodos tradicionais de planejamento, outros têm sido desenvolvidos, principalmente utilizando-se de princípios de Inteligência Artificial, de forma a torná-los o mais abrangente possível [35, 73]. Outros destinam-se ao planejamento de produção de determinados setores como: seleção de equipamentos para manuseio de materiais [106], para seleção de robôs [111], alocação de máquinas [118].

O uso de Tecnologia de Grupo para planejamento de produção também tem sido privilegiado por muitos programas, principalmente para o projeto, ou realocação estática e dinâmica de conjuntos de máquinas [21, 60, 77, 82, 89, 153, 174]. Em especial, o estudo de layout, para implantação e modernização de células e sistemas de manufatura, tem gerado um importante conjunto de publi-

cações, enfatizando diferentes aspectos da Tecnologia de Grupo e Análise Operacional [40, 88].

O sequenciamento de tarefas (scheduling), um dos principais problemas no planejamento de produção, tem experimentado considerável avanço com novos métodos computacionais, [23, 31, 86], inclusive servindo-se de alguns princípios de Inteligência Artificial [12, 16, 48, 83, 145, 160, 179].

Com relação ao processamento, manutenção e controle de qualidade, dificilmente são desenvolvidos métodos computacionais ou sistemas de acompanhamento que não sejam dedicados. Isto se deve ao práticamente infinito número de possibilidades de operação, quando se pensa em acompanhamento de processamento e controle de qualidade em Sistemas Flexíveis de Manufatura. Também quanto à manutenção, cada célula produtiva apresenta suas características, provocando uma análise dedicada de cada caso.

2.3 - A Simulação como Ferramenta de Apoio Gerencial

Apesar do grande volume de sistemas computacionais desenvolvidos para operação junto à Manufatura Integrada por Computador, a figura humana é indispensável. O aspecto decisório e gerencial, desempenhado pelo ser humano, não pode ser programado de forma genérica. Muito menos podem ser consideradas, de uma maneira analítica, todas as alternativas decisórias que surgem dinamicamente durante o projeto, planejamento e operação de sistemas de manufatura.

O conjunto das ferramentas de apoio, citadas acima, caracteriza-se por apresentar um apoio operacional em cada uma das diferentes fases da manufatura integrada. No entanto, uma ferramenta tem-se destacado como extremamente importante no apoio à análise gerencial e decisória, além de também prestar apoio operacional. Esta ferramenta é a Simulação, em suas mais diferentes formas de implementação.

Simulação, de uma forma genérica, tem sido uma ferramenta utilizada nas mais variadas áreas de estudo e pesquisa [26, 47, 155, 172]. De maneira mais particular, a simulação a eventos discretos tem sido usada, de forma crescente, como ferramenta de análise preditiva em Sistemas de Manufatura, do projeto à implementação e operação [64, 144, 149, 173].

O uso da simulação a eventos discretos não substitui as ferramentas anteriormente citadas. Ela, sim, apresenta-se como elemento de apoio à análise gerencial. De uma forma muito especial, a simulação pode capturar os detalhes necessários para uma avaliação da dinâmica desenvolvida por complexos sistemas integrados de manufatura. Esta característica é especialmente importante na análise de planejamento e operação de Sistemas Flexíveis de Manufatura. Isto, porque modelos analíticos dificilmente conseguem expressar adequadamente o grande número de detalhes e a complexidade decorrente da flexibilidade operacional.

De uma forma geral, a utilização da simulação em relação a Sistemas Flexíveis de Manufatura pode ser classificada entre [144]:

a) Estimar o desempenho total de um sistema proposto,

b) Refinar o processo decisório em um sistema já existente.

A estimativa de desempenho pode ser considerada em diferentes situações, de forma isolada ou em colaboração com outras ferramentas em várias situações, por exemplo, ao se projetar uma nova célula de manufatura, ou alterações em uma já implantada.

A simulação pode ser utilizada como única ferramenta para avaliar a eficiência produtiva do modelo do sistema de manufatura proposto, avaliar resultados econômicos do investimento necessário à sua implantação em função da provável produtividade a ser atingida.

Diferentes sistemas de projeto auxiliado por computador podem ser utilizados para estabelecer a configuração (layout) de um sistema de manufatura, selecionando as máquinas e ferramentas a serem utilizadas. A simulação contribui para o estabelecimento de uma metodologia de gerenciamento de ferramentas, frente às circunstâncias.

A simulação, como ferramenta para o refinamento do processo decisório em um sistema existente, pode ser utilizada para a complementação de resultados obtidos por métodos analíticos, como nos casos de estabelecimento de planos de movimentação de insumos e estratégias de controle de tráfego. Também pode ser utilizado na avaliação econômica e temporal de certas situações como: projeto de novas células de manufatura, análise de layout de células de manufatura frente a alterações, avaliação de planejamento de processos, adoção de uma política para planejamento de produção, adoção de metodologias de trabalho frente a contingências, etc.

A enorme gama de possibilidades de utilização da simulação a eventos discretos para sistemas de manufatura levou ao surgimento de inúmeros sistemas computacionais, com as mais variadas características operacionais, áreas de enfoque, metodologias de simulação e capacidades de interfaceamento homem-máquina.

Linguagens de Simulação

O uso de certos "pacotes" de simulação a eventos discretos tornou-se tão comum, que estes passaram a ser chamados de Linguagens de Simulação.

As características de cada uma destas linguagens de simulação advém, principalmente, da estratégia adotada para a seleção de eventos no tempo, e de como as entidades a serem simuladas são definidas [37, 78]. Estas características determinam a perspectiva pela qual a simulação é executada em cada linguagem. A perspectiva adotada em cada linguagem permite sua classificação entre [62]: sequenciamento de eventos, varredura de atividades e interação de processos.

A perspectiva de sequenciamento de eventos corresponde a um sistema modelado pela definição de eventos (chegada de uma tarefa, quebra de uma máquina,

etc.) que podem mudar o estado do sistema. Os eventos são escolhidos e processados sucessivamente no tempo. Qualquer outro condicionamento, que não o temporal, deve ser avaliado internamente à rotina estabelecida no modelo.

A perspectiva de varredura de atividades corresponde a um sistema formado por componentes (peças ou conjuntos) que se engajam em atividades (tornear, furar, etc.), dependendo de certas condições. Uma atividade corresponde, conceitualmente, a uma transição ocorrida no estado do sistema, durante um intervalo de tempo, podendo ser representada pela ocorrência de dois eventos definidos como início e fim da atividade.

A perspectiva de interação de processos apresenta características inerentes a ambas as perspectivas acima. Os componentes de um sistema (clientes, serviços, transações, etc.) progridem através de uma sequência de passos (processos). Cada passo pode corresponder à avaliação de condições e à ocorrência de uma ação, em que as condições estabelecem se a ação pode, ou não, ocorrer em determinado instante do tempo.

Entre as diversas linguagens de simulação destacam-se: General Purpose Simulation System - GPSS [59, 148], Simulation Analysis - SIMAN [132], Simulation Language for Alternative Modeling - SLAM [137].

A linguagem GPSS adota a perspectiva de interação de processos. É composta por blocos construtivos, particularmente direcionados para a modelagem de sistemas através de filas. Os modelos são compostos pela interligação de blocos, através dos quais fluem as transações. Cada bloco representa eventos que causam mudanças no estado do sistema, ou passagem do tempo de simulação. Os recursos são modelados, de uma maneira genérica, através de equipamentos e armazenagens. Um equipamento corresponde a um recurso que processa uma transação por vez, enquanto uma armazenagem pode processar simultaneamente, ou reter qualquer número de transações, até seu limite de capacidade.

A linguagem SIMAN permite a adoção das três classes de perspectivas acima citadas, ou mesmo uma combinação delas. A modelagem de um sistema corresponde à interligação de um conjunto de blocos especiais, com características que incluem declarações específicas, como para a descrição de esteiras, robôs, sistemas de armazenagem, e mesmo células de manufatura. Estes blocos são interligados em uma forma gráfica interativa, formando a malha que compõe o modelo de simulação.

A linguagem SLAM também permite a adoção das três classes de perspectivas citadas, ou mesmo combinação delas. Originalmente escrita em FORTRAN apresenta

considerável portabilidade, e possibilidades de anexação de blocos que modelem entidades particulares. A composição do sistema a modelar é obtida através de uma malha formada por "ramos" e "nós". Ramos são utilizados para representar intervalos de tempo decorridos para as entidades que fluem pela rede. Atrasos condicionais, tomadas de decisão, etc., são experimentados pelas entidades, tanto em nós, como em ramos da rede.

Apesar de seu largo uso, cada uma destas linguagens apresenta suas vantagens e desvantagens quanto às suas características e capacidades [1, 2]. Uma das maiores desvantagens a elas atribuídas é quanto à dificuldade na elaboração de modelos de simulação. Tais dificuldades são provocadas por certas particularidades como: perspectiva própria para a descrição dos elementos a serem simulados, sintaxe rígida, dificuldade na composição de modelos que descrevam, com fidelidade, certas situações reais motivadas pela flexibilidade de sistemas de manufatura.

Devido a estas dificuldades e à larga difusão da utilização destas linguagens, em ambientes industriais e acadêmicos, muitas pesquisas foram desenvolvidas no sentido de aprimorá-las. Entre outras, destacam-se a procura por uma forma de torná-las: de mais fácil utilização; ou mais completas para suprir situações não cobertas por sua estruturação original.

Aperfeiçoamentos às Linguagens de Simulação

Os aperfeiçoamentos anexados às linguagens de simulação podem ser classificados [152] em duas grandes categorias de sistemas de auxílio ao usuário: sistemas para especificação de modelos e sistemas para geração automática de códigos de simulação.

Os sistemas para especificação de modelos têm apresentado três vertentes de trabalho: Interfaces de Diálogo Interativo, Interface Gráfica Interativa, e Interface de Linguagem Natural.

As Interfaces de Diálogo Interativo correspondem a um conjunto de questões que são formuladas ao usuário. A sequência das questões é dependente das respostas obtidas. Ao final da sessão de diálogo, obtém-se uma especificação do modelo a ser simulado.

As Interfaces Gráficas Interativas correspondem a um menu de ícones, selecionáveis pelo usuário. Com estes ícones, constrói-se uma representação gráfica do sistema a ser simulado. Após isto os valores específicos do modelo são anexados às informações adquiridas graficamente.

As Interfaces de Linguagem Natural permitem ao usuário a entrada de informações através de texto em formato livre. A interface seleciona as partes importantes do texto e gera os modelos para simulação.

Os sistemas, para geração automática de código de simulação, realizam transformação automática de uma especificação interna do modelo a ser simulado para um código executável da linguagem de simulação escolhida. Duas principais vertentes têm-se desenvolvido neste sentido: geração de código diretamente a partir da representação interna do modelo e geração de uma biblioteca de macros a ser utilizada pelo usuário na composição do código de simulação.

A geração de código a partir da representação interna é transparente ao usuário, mas relativamente restritiva na sua abrangência. A geração de uma biblioteca de macros corresponde à criação de certos blocos de código de simulação, escritos na linguagem alvo para simulação. Estes blocos apresentam o conjunto de código específico para a simulação de certos componentes genéricos, encontrados com mais frequência, e podem ser utilizados pelo usuário na composição de seu código de simulação. Embora mais abrangente, a sua utilização requer do usuário um maior conhecimento da linguagem de simulação.

Muitos trabalhos foram desenvolvidos, adotando estas metodologias de auxílio ao usuário. Como exemplos, citam-se os de Eid e Poirier [32], o de Jones e Greene [71] e o de Sarker [146].

O primeiro descreve uma interface de diálogo para a orientação do usuário na geração de comandos em SLAM, quando da produção de código para simulação específica de certa oficina.

O segundo descreve a criação de uma interface para geração de modelos de simulação em GPSS a serem ligados e simulados, no ambiente SAS. O principal propósito é o de aproveitar-se do grande poder de análise estatística oferecido pelo SAS, na avaliação e tomada de conclusões sobre simulações.

O terceiro apresenta uma interface para auxílio na modelagem específica de sistemas produtivos, utilizando a metodologia "just-in-time" de produção.

Quanto ao desenvolvimento de macros, para utilização na criação de modelos genéricos para uma determinada linguagem de simulação, alguns trabalhos se destacam.

Schroer [149, 150] e Schroer e Tseng [151] apresentam o desenvolvimento de macros para a linguagem GPSS. Estas macros são utilizadas para a modelagem de operações unitárias, estações de montagem, células de manufatura, unidades de inspeção de qualidade e elementos de transferência de estoque. A utilização

destas macros é, ainda, dirigida por uma interface de diálogo interativo, escrita em Pascal, facilitando a composição do modelo.

Esta metodologia também é utilizada por Haddock em [53], quando descreve um conjunto de macros, escritas em FORTRAN para utilização com SIMAN. Estas macros encarregam-se da execução de certas análises estatísticas mais detalhadas, para a avaliação de performance de sistemas de manufatura simulados em SIMAN.

No entanto, ultimamente, os principais trabalhos desenvolvidos têm reunido ao menos uma das opções de auxílio na composição de modelos, com uma das opções de auxílio na geração automática de código de simulação.

Haddock [54] desenvolveu uma interface de diálogo denominada gerador de modelos de simulação. Esta interface é escrita em Basic, para especificação de modelos a serem usados em projeto e controle de Sistemas Flexíveis de Manufatura. Este gerador inicialmente orienta o usuário na composição de modelos para simulação em SIMAN. Posteriormente, encarrega-se da conversão dos dados do modelo em código executável para simulação, nesta linguagem.

Nesta linha também se enquadram os trabalhos de Mellichamp e Wahab [112, 113], e Murray e Sheppard [124]. Os primeiros, criando interfaces para a geração de modelos para simulação operacional e financeira de sistemas de manufatura usando manipuladores de conhecimento [65], e interfaces para análises estatísticas e formulação de conclusões sobre simulações realizadas. Os últimos, desenvolvendo várias interfaces: gráficas, de diálogo e de reconhecimento de linguagem, para especificação de modelos a serem simulados em SIMAN.

Ford e Schroer [39], Schroer e Tseng [152], apresentam o desenvolvimento de interfaces de linguagem natural, direcionadas à geração de modelos e códigos de simulação, para simulações a serem realizadas em SIMAN e GPSS, respectivamente. As interfaces são construídas em torno de um dicionário de palavras-chave, expressões e resultados de simulações, referentes às montagens eletrônicas. A criação do modelo é obtida a partir do reconhecimento destas palavras e expressões em um texto, descrevendo a célula de manufatura a ser modelada.

Standridge [168] apresenta um sistema denominado Extended Simulation System - TESS, onde foram criadas interfaces de entrada e de saída para simulações a serem realizadas em SLAM. Duas interfaces gráficas de entrada permitem a especificação do modelo a ser simulado. Uma das interfaces permite a especificação através de criação de uma malha gráfica esquemática, específica

para a linguagem SLAM. A outra permite a especificação do modelo através do preenchimento de formulários específicos. As interfaces de saída apresentam análises estatísticas e animações de simulações realizadas.

Apesar do considerável avanço obtido no desenvolvimento de interfaces para linguagens de simulação, estas sempre se apresentam restritas quanto às estruturas das linguagens para as quais se destinam, e quanto ao específico domínio dentro do universo da manufatura, para o qual foram desenvolvidas. Assim sendo, dificilmente, um sistema computacional deste tipo poderia ser utilizado, ou adaptado, para situações distintas daquelas para as quais foi projetado.

Considerando, agora, a flexibilidade dos atuais sistemas de manufatura conclui-se que, ou tem-se um uso restrito da interface de simulação disponível, ou utiliza-se de várias interfaces distintas para suprir as várias áreas de análise em que a simulação poderia ser utilizada.

Estas restrições quanto ao domínio de aplicabilidade dos sistemas baseados em interfaces para linguagens de simulação, têm sugerido estudos mais amplos, que cobrem novas formas de apoio e aperfeiçoamento à realização de simulações de sistemas complexos. Em especial, estudos a respeito de novas formas de representação e utilização de conhecimento para a simulação têm sido desenvolvidos com este propósito.

2.4 - A Simulação e os Sistemas Especialistas

Os conceitos fundamentais adotados em pesquisas sobre Inteligência Artificial, principalmente no campo denominado Sistemas Especialistas, apresentam uma considerável ênfase na representação do conhecimento. Em particular, fornecem técnicas de aquisição e utilização de conhecimento que podem ser eficazmente aproveitadas na especificação de sistemas com frequentes tomadas de decisão, como é o caso de sistemas complexos de manufatura [74, 122, 156].

Quanto à utilização de sistemas especialistas em conjunto com sistemas de simulação, conforme apresentado por O'Keefe [127], esta pode ser realizada por uma das seguintes maneiras:

- aninhando um sistema especialista em um sistema de simulação, ou vice-versa,
- um sistema de simulação consultando um sistema especialista, ou vice-versa,
- ambos os sistemas operam em paralelo, respondendo diretamente ao usuário, ou aninhados em um ambiente computacional mais amplo, que os contenha,
- um sistema especialista opera como uma interface inteligente entre o usuário e um sistema de simulação.

Dentre estas possibilidades, uma das mais aproveitadas tem sido a operação de sistemas especialistas como interface inteligente para sistemas de simulação. Esta foi a opção utilizada na maioria dos trabalhos em que foram criadas interfaces para as linguagens de simulação. Na realidade, esta tem sido praticamente a única opção de utilização de apoio de sistemas especialistas a sistemas de simulação, embora a implementação de outras possibilidades de integração entre sistemas especialistas e sistemas de simulação apontem para outras oportunidades interessantes [177].

Uma outra área, importante para o desenvolvimento de novos aperfeiçoamentos no uso da simulação, tem sido a pesquisa de novas formas de representação do conhecimento necessário à realização de simulações.

A utilização de alguns conceitos fundamentais da Programação Orientada a Objeto [29, 45, 143] pode facilitar a descrição do comportamento de sistemas complexos, como os de manufatura. Dentre estes conceitos, destacam-se o princípio de organização hierárquica em classes e o ocultamento das informações internas de um objeto em relação aos outros.

Importantes progressos foram feitos no desenvolvimento do uso de Sistemas Especialistas em conjunto com novas formas de representação de conhecimento,

voltados para a simulação. A maioria dos trabalhos desenvolvem metodologias para a modelagem de sistemas de manufatura, adequadas aos conceitos de hierarquização do modelo e aos conceitos impostos pelo paradigma de objetos. Com isto, procuram expressar a hierarquização decisória e funcional, além da própria flexibilidade, importantes características de sistemas flexíveis de manufatura.

Zeigler [180] discute uma especificação modular e hierárquica para a criação de modelos para simulação a eventos discretos. Estes modelos são construídos de acordo com um formalismo específico, chamado DEVS, implementados em um dialeto da linguagem Lisp.

Antonelli e outros [7] propõem uma dupla decomposição dos modelos de sistemas de manufatura. A primeira, denominada Decomposição Hierárquica, que subdivide um problema (sistema) a ser resolvido, em outros menores. A segunda, a Decomposição Procedimental, que apresenta uma descrição funcional de cada objeto descrito. A linguagem adotada para a decomposição procedimental é ADA. O modelo de simulação acaba por ser um programa escrito em ADA.

Bagrodia e outros [8] apresentam uma linguagem de simulação denominada MAY, baseada em FORTRAN. Com esta linguagem realiza-se uma modelagem de sistemas através de um conjunto de mensagens transferidas. Uma entidade é definida como um processo de transmissão de mensagens, enquanto eventos são as mensagens transferidas. Os segmentos de programa, representando os modelos especificados, podem ser ligados a alguma das linguagens e simulação de propósito geral.

Levas e Jayaraman [91] descrevem um ambiente de apoio ao projeto de células de manufatura, denominado WADE. Este ambiente reúne uma modelagem geométrica de elementos (GDP) com uma representação orientada a objetos. Como apoio, desenvolve simulações sobre estes modelos, utilizando-se de uma linguagem orientada a objetos, denominada AML/X. A modelagem de entidades é hierarquizada em termos de informações geométricas, cinemáticas, de comunicação e de características de processo.

Burns [18] propõe uma linguagem de especificação para simulação a eventos discretos. Esta linguagem guarda certas similaridades com ferramentas de Computer Aided Software Engineering, no sentido de que engloba o desenvolvimento do software para simulação com sua documentação associada. Cada modelo de entidade corresponde a segmentos de programa que podem ser reutilizados em outras simulações. As entidades são definidas como objetos, caracterizados

através de seus relacionamentos com outros objetos.

Golden [46] faz um conjunto de considerações a respeito de projeto de linguagens para simulação. Enfatiza alguns aspectos da Engenharia de Software aplicada a este tipo de linguagens, em particular a modularidade e estruturação dos modelos de sistemas a serem simulados. Em especial, discute certas características a respeito das estruturas de dados a serem utilizados para simulação discreta e contínua.

Fox e outros [41] apresentam um sistema de simulação baseado em conhecimento denominado KBS. Nele a descrição de modelos para simulação é formada por redes semânticas hierarquizadas. Cada objeto é descrito por um conjunto de atributos, uma estrutura de comportamento e relações com outros objetos.

A aquisição de modelos para simulação é realizada utilizando-se de uma interface gráfica onde, através de ícones, se estabelece uma rede que representa o relacionamento entre diferentes objetos. As interfaces são dependentes do contexto, podendo ser intercambiadas conforme o caso em estudo.

Um sistema especialista dedicado procede a uma análise automática dos resultados de simulação, de acordo com especificações do usuário. Um aperfeiçoamento deste sistema é o SimulationCraft [42], produto disponível comercialmente.

Moser [122] sugere uma integração de um Sistema Especialista, direcionado para questões financeiras e de gerenciamento corporativo, com um Sistema Simulador. O conhecimento a ser utilizado no Sistema Especialista é composto por regras e fatos. As informações referentes a considerações temporais são obtidas através do sistema simulador. Estas informações podem ser agregadas à base de conhecimento para futuras inferências. Este sistema, denominado EXSYS, é todo composto utilizando-se da linguagem FORTRAN.

Em todos os casos citados acima, a especificação dos sistemas de manufatura para realização de simulações foi expressa de maneira não formal. Estas especificações caracterizam-se por adotar sintaxe e semântica direcionadas para uma linguagem específica de simulação. Com isto carregam todas as ambiguidades, características do não-determinismo das linguagens a que se destinam.

Sabe-se, no entanto, que estes mesmos sistemas de manufatura podem ser especificados de uma maneira formal [20, 79].

A Lógica Temporal tem sido um formalismo utilizado no projeto e implemen-

tação de protocolos de comunicação [9, 110, 119], e de uma formal geral para especificação e verificação de processos concorrentes [55, 103, 108, 109, 134, 135, 164, 178]. Em particular, tem sido utilizada na verificação e síntese de controladores para sistemas dinâmicos a eventos discretos (DEDS) [129, 171].

Apesar das vantagens de uso de especificações formais, estas não têm sido usadas com frequência na descrição de sistemas de manufatura. Isto se deve, principalmente, a que a sequência natural de sua utilização corresponderia a um processo de verificação e prova formais das especificações. Além de complexos, estes passos envolvem muitas características subjetivas, relacionadas com procedimentos de prova de teoremas, atributos presentes em poucos profissionais.

Em outras áreas, como na Engenharia de Software, na qual a especificação formal de processos tem-se feito mais necessária, esta tem sido utilizada e desenvolvida. Em particular, salienta-se o caso de desenvolvimento de compiladores para transformação automática de especificações formais em código de linguagens de alto nível, como Lisp ou C [10, 131].

Avanços como estes sugerem a pesquisa no sentido de desenvolvimento de maneiras para a utilização de especificações formais, não só para a descrição de complexos sistemas de manufatura, mas também para a geração de modelos para simulações.

3 - UM AMBIENTE INTEGRADO DE APOIO À TOMADA DE DECISÃO

3.1 - Introdução

Como apresentado em capítulo anterior, a análise organizacional da Manufatura Integrada por Computador (CIM) mostra ser esta estruturada em três principais áreas quais sejam, Engenharia Auxiliada por Computador (CAE), Planejamento Auxiliado por Computador (CAP) e Manufatura Auxiliada por Computador (CAM) [61].

Embora a completa interligação dos diferentes setores da manufatura em redes de computadores seja uma emergente realidade, o real e eficiente compartilhamento de informações entre setores ainda não tem sido possível. Isto se deve ao fato de que a formatação das informações, quando de sua geração em cada área, não tem ainda atingido uma adequada padronização e integração. Provoca-se então certa multiplicidade de recursos (humanos, físicos e computacionais) e de informações, que frequentemente leva a disrupções na integridade das informações.

A análise de informações, com consequente tomada de decisão, realiza-se sobre informações compartilhadas por diferentes áreas, só que avaliadas em diferentes contextos, dependendo de cada nível hierárquico, físico ou funcional, em que se esteja processando a análise desejada.

Enquanto ferramenta de análise operacional e gerencial, a simulação deve utilizar-se de informações geradas em todos os níveis hierárquicos nos quais seu apoio seja necessário. No entanto, para que esta utilização seja eficiente, necessário é que haja uma padronização e integração das informações geradas nos diferentes níveis da manufatura. Assim este conjunto de informações estaria disponível para utilização, tanto em simulações como em outros métodos de análise, de forma não ambígua nem duplicada.

Este trabalho propõe a criação de um **Ambiente Integrado de Apoio à Tomada de Decisão - AIATD**. Entende-se por apoio à tomada de decisão como a apresentação de um conjunto de informações composto por simulações, previsões, análises de comportamento e inferências lógicas sobre o conhecimento detido a respeito de um sistema em estudo.

A operação do AIATD é suportada em duas ferramentas básicas: um simulador a eventos discretos e um processador configurável de conhecimento. O Simulador a eventos discretos opera como ferramenta geradora de conhecimento dinâmico. Este conhecimento dinâmico, quando acrescido ao conhecimento estático já deti-

do a respeito do sistema em estudo, forma uma base integrada de conhecimento. Esta base, para simplicidade de utilização, é composta de forma configurável pelo usuário. O Processador de Conhecimento opera sobre a base integrada, realizando armazenagem, recuperação de informações e inferências lógicas, também de forma configurável pelo usuário.

A configurabilidade de operação do AIATD, quanto à composição da informação a ser utilizada e quanto à forma de sua utilização pelo Simulador e pelo Processador de Conhecimento, busca torná-lo bastante flexível operacionalmente. Sua flexibilidade apresenta duas grandes vantagens operacionais:

1- possibilidade de uso do ambiente em distintos graus hierárquicos e em diferentes áreas organizacionais da manufatura;

2- possibilidade de um rápido aprendizado de sua utilização por parte dos profissionais diretamente interessados nos resultados de sua operação.

A possibilidade de uso do AIATD em diferentes graus hierárquicos e em diferentes áreas organizacionais, busca permitir a alocação distribuída da base de conhecimento, embora integrada em sua composição, que leva a um compartilhamento de recursos, com maior aproveitamento do hardware e do software. A alocação distribuída da base de conhecimento também leva a uma maior integridade do conhecimento, já que não ocorre duplicidade de informação.

A possibilidade de um rápido aprendizado da utilização do AIATD procura trazer duas importantes consequências: diminuição do custo de uso de software de apoio e maior rapidez e correção na obtenção de conclusões. A diminuição do custo advém do fato de poder-se eliminar a necessidade de colaboração de um especialista no uso do software de apoio. A maior rapidez e correção na obtenção dos resultados advém da operação do ambiente diretamente pelo interessado em seus resultados, eliminando profissionais intermediários.

Destacam-se, assim, como principais contribuições deste ambiente, à arte da tomada de decisão, os seguintes fatores:

- Simplificação do uso cooperativo da simulação e processadores de conhecimento;
- Integração do conhecimento relativo a diferentes níveis hierárquicos físicos e funcionais, e áreas organizacionais, em uma única base de conhecimento a ser utilizada para simulações e inferências lógicas;
- Flexibilidade de operação do ambiente, advinda da configurabilidade do uso das ferramentas de apoio;
- Eliminação da necessidade de intermediarios na geração e análise de conheci-

mento;

• Diminuição do custo de uso de software de apoio à tomada de decisão.

3.2 - Requisitos para o Ambiente

O Ambiente Integrado de Apoio à Tomada de Decisão - AIATD, tal como proposto, destina-se, inicialmente, a ser usado no universo da Manufatura Integrada por Computador (CIM), de forma distribuída, tendo como ferramentas de apoio um simulador a eventos discretos e um processador configurável de conhecimento.

Em função das ferramentas em que se encontra baseado o AIATD, pode-se estabelecer o universo sobre o qual este é aplicável, ou seja, estabelecer-se seu domínio. A princípio, um simulador a eventos discretos é aplicável a quaisquer sistemas, cujas ocorrências sejam discretas ou passíveis de discretização [47, 155]. Um processador de conhecimento tem capacidade de realizar inferências sobre informações apresentadas de acordo com uma sintaxe estabelecida [161]. Tendo-se como meta de aplicação do AIATD o universo da Manufatura Integrada por Computador, ficam estabelecidos, como componentes do Domínio das Informações, os sistemas de manufatura passíveis de serem descritos por eventos discretos.

A aplicação do AIATD sobre o Domínio das Informações obtém, como resultado, um outro conjunto de informações composto por: simulações, previsões de comportamento do sistema em estudo, inferências lógicas sobre simulações, previsões de comportamento e outros conhecimentos armazenados. Determina-se, assim, o Contra-Domínio das informações, ou seja, o universo de resultados obtidos pela operação do AIATD sobre o Domínio das Informações.

A representação da aplicação do AIATD é simbolizada pela figura 3.1.

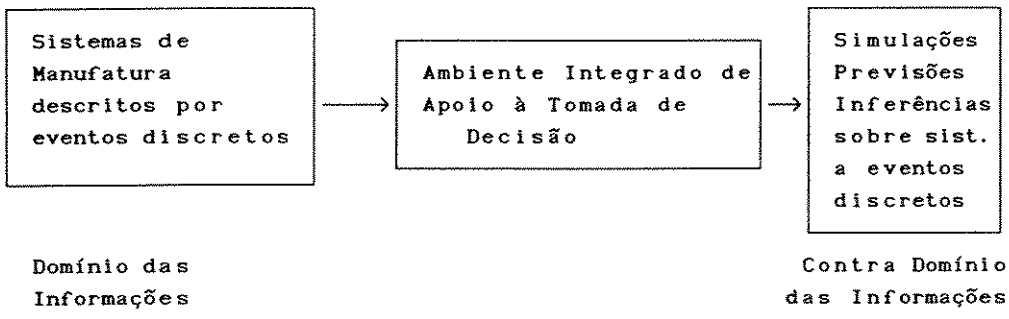


Fig.3.1 - Domínio e Contra Domínio do Ambiente

Consideradas as informações provenientes das três grandes áreas organizacionais da manufatura, CAE, CAP e CAM, sua tipificação sugere dois grandes

tipos de informações: Informações de Projeto e Informações de Manufatura [61]. As Informações de Projeto são partilhadas pelas áreas de Engenharia e Planejamento, enquanto as Informações de Manufatura são partilhadas pelas áreas de Planejamento e Manufatura.

A distinção entre Informações de Projeto e Informações de Manufatura também se apresenta quanto à temporalidade de seu uso, já que o processamento de Informações de Projeto tem limites e restrições de tempo bem menos rígidas que o processamento de Informações de Manufatura [80, 117]. A atual tendência de desenvolvimento de redes de comunicação de dados entre computadores, nos ambientes de manufatura, também aponta para uma distinção entre os protocolos que operam a nível de manufatura (chão de fábrica) e aqueles que operam a nível de projeto, planejamento e gerenciamento [43, 61, 123].

Este conjunto de distinções sugere que se apresente uma setorização das aplicações do AIATD. Esta setorização pode seguir a mesma divisão citada para a tipificação de informações, ou seja, Informações de Projeto e de Manufatura. Isto se deve ao fato de que há uma considerável distinção entre o conhecimento (dados e heurística) a ser processado tanto nas simulações como nas inferências lógicas a serem realizadas para as diferentes áreas da Manufatura Integrada por Computador [84, 177]. Esta distinção apresenta-se de várias formas, desde o volume de conhecimento a ser processado, até a frequência deste processamento, passando pela própria tipificação das informações.

Considera-se, então, uma distribuição setorizada para a aplicação do AIATD na Manufatura Integrada por Computador. Um setor de aplicação corresponde ao apoio às áreas de projeto, planejamento e gerenciamento, enquanto o outro corresponde à área da manufatura propriamente dita. Em cada um destes setores, propõe-se o estabelecimento de um módulo de Ambiente de Apoio à Tomada de Decisão. Para cada um deve-se compor uma base de conhecimento e uma base de dados para o sistema em estudo.

A setorização do processamento não impede que estas informações sejam trocadas, ou partilhadas, entre diferentes setores. Neste caso, requer-se, apenas, que as informações necessárias sejam requisitadas através da rede de comunicação. Representa-se esta setorização de processamento através da figura 3.2.

Chama-se Base de Conhecimento às informações estáticas, dinâmicas, e à heurística a respeito do sistema em estudo. Este conhecimento é adquirido de especialistas do setor e de interações com o sistema, sendo progressivamente

aperfeiçoado. O conhecimento a respeito do sistema pode encontrar-se centralizado ou distribuído em vários componentes ao longo da rede de comunicação.

A suposição que o conhecimento esteja distribuído exige que haja uma comunicação entre o AIATD e outros computadores armazenadores de informação. Esta comunicação requer do ambiente uma flexibilidade que permita a interconexão com equipamentos operando com diversos protocolos de comunicação. Devido a este fato, o AIATD deve apoiar-se em uma estrutura de comunicação modular e aberta, capaz de respeitar o modelo de referência OSI - Open System Interconnection [52, 94, 141].

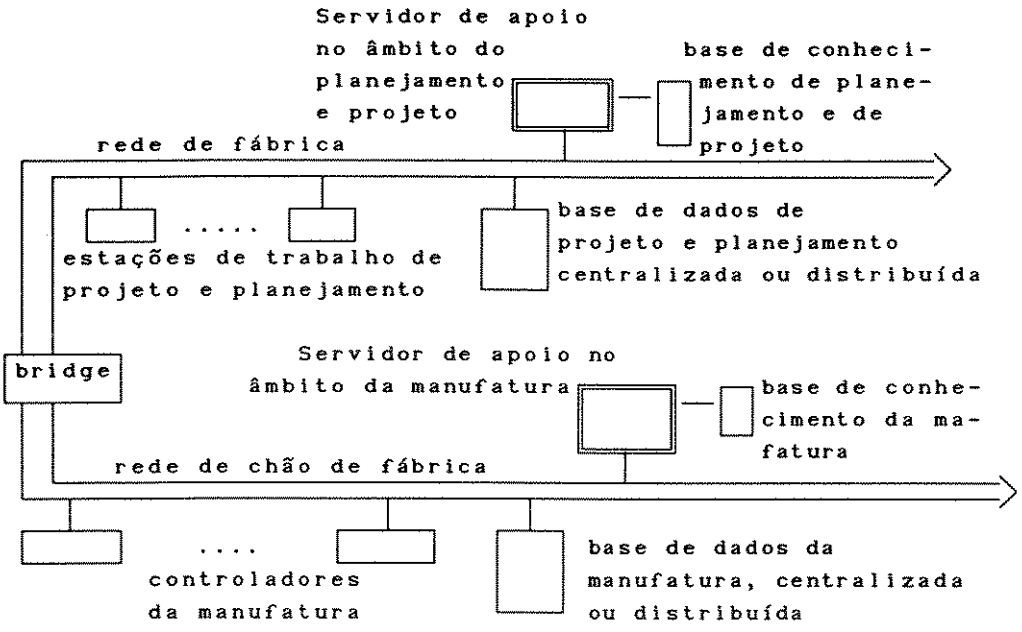


Fig. 3.2 - Servidores de Apoio em rede fabril de comunicação

Quanto à utilização do AIATD, uma das principais metas é que a representação do conhecimento seja facilmente realizada e compreendida pelos profissionais diretamente envolvidos com a tomada de decisão, nas mais diferentes áreas e níveis hierárquicos da manufatura. Isto permite que se prescinda de especialistas em aquisição de informações, simulação e realização de inferências lógicas, auxiliadas por computador [49].

Esta integração no processo de apoio à tomada de decisão permite que o profissional responsável direto pela tomada de decisão tenha uma visão global e um melhor conhecimento de todo o processo, desde a aquisição de informação até à decisão. Isto também permite que este profissional possa estabelecer uma

relação causa-efeito das informações, aperfeiçoando o tipo e a forma de aquisição das informações mais importantes para cada tipo de problema.

Para que se prescinda do especialista em aquisição de informações, alguns requisitos devem ser cumpridos [49]:

• a apresentação da informação deve ser realizada o mais próximo possível da forma coloquial, guardando uma relação direta com o conteúdo da informação.

• a informação adquirida deve ser armazenada de forma estruturada, guardando uma relação direta com os níveis hierárquicos, físico e funcional, a que se referem.

• a informação adquirida deve poder ser processada por qualquer uma das ferramentas de apoio que se queira utilizar, isto é, ela deve ser reutilizável.

A eliminação do especialista em simulação requer também que alguns importantes requisitos sejam cumpridos para a composição dos modelos de simulação e para a execução da simulação propriamente dita.

Quanto à composição do modelo de simulação, é desejável que [7, 18, 51, 75, 124, 156]:

• este seja composto de forma estruturada, separando-se a estrutura de dados do conjunto de dados que compõem o modelo.

• seja possível compor novos modelos, a partir de outros, através do detalhamento ou generalização das informações, isto é, que o modelo seja hierárquico.

• sua apresentação seja em forma declarativa.

Quanto à execução da simulação, é desejavel que esta se realize diretamente sobre modelos compostos de forma declarativa, executando uma computação simbólica, de forma a não haver necessidade de recomposição ou tratamento prévio de informações [75, 156].

Com estas características, facilita-se o trabalho de modelagem e realização de simulações, diretamente pelo interessado nestas atividades, ou seja, o responsável pela tomada de decisão.

Em relação ao processador de conhecimento, este deve ser capaz de manipular tanto conhecimento estático como conhecimento dinâmico. O conhecimento estático é aquele adquirido de especialistas. O conhecimento dinâmico é proveniente de simulações, ou de atualizações da base de dados da manufatura, devido à operação do sistema em estudo.

Isto requer que a formatação do conhecimento cumpra alguns requisitos como:

• ser flexível, a ponto de que o conhecimento disponível possa ser usado,

indistintamente, tanto pelo simulador como pelo processador de conhecimento;
• ser aplicável para a realização de análises variadas, em diferentes níveis hierárquicos e nas diferentes áreas da manufatura.

Tais requisitos exigem um redobrado cuidado com a determinação da estrutura de representação dos elementos da manufatura, principalmente por se levar em consideração que tanto a simulação como a inferência lógica possam ser realizadas sobre uma mesma base de conhecimento, com mesma tipificação de informações.

Para que se respeitem os requisitos acima, há que supor-se um modelo abstrato genérico para a representação da informação da manufatura. Também assume-se que simulação e inferência sejam funções de mesmo nível funcional, operando sobre partes de uma mesma base de conhecimento.

3.3 - O MODELO ABSTRATO DE INFORMAÇÃO

O Modelo Abstrato de Informação deve ser capaz de representar os sistemas que compõem o Domínio das Informações (sistemas a eventos discretos), em qualquer uma das áreas organizacionais da manufatura.

Cada uma destas áreas apresenta diferentes aspectos operacionais, trabalhando com diferentes conjuntos de informação. [75, 138, 157]. Por diferentes conjuntos de informação entendem-se as variadas formas de representação e reunião dos dados que descrevem, estrutural e funcionalmente, os sistemas de manufatura sob diferentes pontos de vista, tais como técnico, comercial, e gerencial.

As necessidades específicas na solução de problemas de cada área direcionam o tipo de informação a ser considerada e a forma em que esta deva ser utilizada [46, 70, 156]. Assim sendo, fatores importantes na concepção do modelo de informação a ser utilizado no ambiente proposto são:

- o domínio das informações;
- o conjunto disponível de ferramentas de análise;
- o contra-domínio das informações.

O domínio das informações pode ser considerado como o conjunto das informações a respeito de projeto, planejamento e controle de elementos ou sistemas, relacionados com cada uma das áreas da manufatura, sob os diferentes pontos de vista, conforme citado acima.

As ferramentas de análise disponíveis no ambiente são: um Simulador a eventos discretos e um Processador de Conhecimento.

O contra-domínio das informações corresponde ao conjunto de informações resultantes da operação das ferramentas de análise disponíveis no ambiente sobre o domínio de informações. Este conjunto é formado por análises tais como simulações, previsões de comportamento, inferências lógicas e interação entre simulações e inferencias lógicas.

Descrição de elementos da manufatura

A descrição de um mesmo sistema de manufatura pode ser realizada de diferentes formas, para uso quer em simulação, quer em inferências lógicas. Até mesmo um único elemento da manufatura pode ser descrito por diferentes conjuntos de informações, adequados para a realização de simulações ou execução de

inferências lógicas, dependendo dos interesses de quem gera ou de quem recebe as informações.

Considerando, então, apenas um sistema de manufatura, pode-se ter uma inconveniente multiplicidade de conjuntos de informações, passíveis de serem usados em cada uma das ferramentas de análise disponíveis no ambiente.

Por outro lado, tanto a realização de simulações como a execução de inferências lógicas, sobre sistemas de manufatura, têm como um de seus propósitos fundamentais justamente integrar diferentes elementos em um todo que evidencie seu interrelacionamento [156, 157]. Em termos do AIATD, esta integração deve ser obtida, quer através de previsões de comportamento, quer através de avaliações de relação causa-efeito, ou mesmo composição destes.

Devido à multiplicidade de possíveis conjuntos de informação que descrevam o domínio considerado, percebe-se uma grande dificuldade para a adoção de um conjunto de informação que represente, ao mesmo tempo, a abrangência do domínio de informação e ainda permita a realização dos interrelacionamentos desejados, através das ferramentas do AIATD.

Torna-se, então, necessária a criação de um modelo de informação que possa suprir tais necessidades, bem como apresentar uma estrutura que preencha os requisitos estabelecidos para a operacionalidade do AIATD.

PARADIGMAS PARA A COMPOSIÇÃO DO MODELO

Dada a complexidade da tarefa de criação do modelo de informação, é conveniente a adoção de técnicas que a estruturem e facilitem seu desenvolvimento.

Tanto a especificação como a modelagem de sistemas de manufatura guardam semelhança com a especificação de sistemas computacionais [46].

Modernos conceitos de especificação de sistemas apontam para algumas características desejáveis, de modo que a especificação seja:

- unificada;
- incremental;
- flexível;
- hierarquizada.

Ser unificada corresponde a permitir uma especificação que combine a descrição da estrutura e da funcionalidade do sistema, incremental permite uma fácil agregação de novas informações ao sistema, aperfeiçoando-o. Ser flexível corresponde a permitir o uso de uma mesma especificação em contextos distin-

tos, e hierarquizada para que a descrição do sistema possa ser de forma a representá-lo com diferentes graus de detalhamento, conforme necessário.

Dentre as muitas metodologias já desenvolvidas para a especificação e implementação de sistemas de computação [18, 46, 136], destaca-se a Programação Orientada a Objetos, pela sua adequação ao problema [15, 116].

O Paradigma de Objetos privilegia tanto o aspecto do interrelacionamento entre distintos elementos (objetos) como a adoção de altos níveis de abstração para a composição de um modelo de informação, isto porque este paradigma propicia, em comum com a abordagem de tipos abstratos de dados, um alto grau de coesão para cada objeto, além de um baixo grau de acoplamento entre diferentes objetos [170].

Esta conjunção de alto grau de coesão de um objeto com baixo grau de acoplamento entre objetos, é altamente desejável para a análise de sistemas de manufatura, uma vez que estes apresentam uma grande multiplicidade de elementos a serem considerados, sob diferentes pontos de vista.

A utilização deste paradigma, na definição do modelo de informação, permite a criação de modelos descritivos de entidades da manufatura (elementos componentes da manufatura) que sejam autoconsistentes, e que possam ser agregados uns aos outros, através da descrição de seus interrelacionamentos. A análise de novas configurações da manufatura pode ser realizada através da troca de um modelo descritivo por outro, mantendo-se o restante da configuração inicial e o contexto da análise.

O paradigma de objetos também privilegia a hierarquização de modelos descritivos de entidades, ao considerar, de forma clara, o mecanismo de herança, através de operações abstratas de generalização e agregação. Isto simplifica a representação hierarquizada de sistemas de manufatura, facilitando a realização de uma análise estruturada, através da avaliação de comportamento de entidades da manufatura descritas com diferentes graus de detalhamento. Abre-se, assim, a possibilidade de um bom controle da granularidade com que se deseja analisar um sistema de manufatura, além de permitir um desenvolvimento incremental de novos modelos descritivos. [7, 49, 156].

Outro aspecto importante do paradigma de objetos corresponde ao ocultamento, para outros objetos, das informações internas a cada um. Toda comunicação entre objetos desenvolve-se através de troca de mensagens [29, 45, 170], o que permite abstrair-se, de forma quase completa, das atividades desenvolvidas internamente a cada objeto. Ainda mais, em termos de descrição da interação

entre objetos, o único fator a ser considerado corresponde à determinação do tipo das mensagens a serem recebidas e emitidas em cada objeto. Quanto às realizações de cada objeto, utiliza-se a abordagem do tipo abstrato de dados, isto é, define-se um conjunto próprio de informação e um conjunto de operações sobre estas.

3.3-1 - Composição do Modelo de Informação

A especificação de elementos discretos da manufatura pode ser desenvolvida tanto com um aspecto formal como informal, cada um com suas próprias características, vantagens e desvantagens.

A especificação formal tem por meta a geração de um conjunto de informação tal que a descrição comportamental do elemento em estudo possa ser formalmente provada. As lógicas de primeira ordem, mais especificamente a Lógica Temporal tem-se apresentado como ferramenta bastante utilizada para a especificação comportamental de sistemas de manufatura [17, 38, 92, 128, 129, 130, 139, 171]. No entanto, a verificação e prova de tais especificações, além de apresentar um carater subjetivo, requer de seu responsável uma grande experiência na área de prova de teoremas [55, 109].

A especificação informal tem por característica a realização de uma descrição declarativa do elemento em estudo. Tem sido mais utilizada como forma de capturar o conhecimento humano, dentro de certo domínio bem determinado e geralmente restrito. Visa, de forma genérica, uma transferência de conhecimento que posteriormente sofrerá tratamentos próprios [76, 139, 177]. Devido à sua natureza especial, tais especificações não permitem uma verificação de sua completude e segurança, muito menos sua prova.

O modelo de informação desenvolvido caracteriza-se por apresentar uma representação genérica de sistemas de manufatura, utilizando-se de um alto nível de abstração, de forma a poder ser utilizado com ambas as formas de especificação acima citadas. Também representa tanto as características estruturais como as comportamentais daqueles sistemas, que devem ser tratadas computacionalmente pelas ferramentas de análise disponíveis no AIATD (simulador e processador de conhecimento).

As características estruturais descrevem a composição organizacional (frequentemente hierárquica) dos objetos que compõem o sistema. As

características comportamentais descrevem as ocorrências internas a cada objeto componente do sistema, relacionando-se íntimamente com a estrutura de dados adotada [7].

A especificação de Sistemas Computacionais implica no desenvolvimento destes mesmos tipos de descrição [136]. Devido a esta semelhança, pode-se estabelecer uma analogia entre a especificação de sistemas computacionais e de sistemas de manufatura, passando a utilizar das técnicas adotadas para os primeiros.

Para o desenvolvimento deste modelo de informação, requereu-se dele o cumprimento de alguns dos requisitos de especificação utilizados na metodologia de desenvolvimento automático de software [14]. Em particular, foram considerados os requisitos: relacional de entrada-saida, de hierarquia de sistema, descritivo de objeto e relacional de objeto. Por fim, para a integração das especificações, requer-se o cumprimento do requisito de completude.

Cada elemento genérico da manufatura, daqui por diante chamado de entidade, é considerado como um objeto abstrato a ser especificado. A necessidade de respeito aos requisitos comportamentais e estruturais direcionou a elaboração do modelo abstrato de informação. A elaboração deste modelo independe de qualquer linguagem adotada para realizá-lo. Isto se deve ao fato de ser a definição de requisitos um processo puramente declarativo que envolve caracterização de um objeto e não a descrição de processos.

a) O modelo abstrato genérico de informação

A concepção do modelo genérico pôde ser iniciada a partir da aplicação do requisito relacional de entrada-saída [14].

Requisito relacional de entrada-saída: "O comportamento de todo objeto (ou sistema) pode ser especificado através de uma referência lógica entre suas entradas e saídas."

Este requisito leva à idealização de uma entidade abstrata na qual exista um conjunto de entradas, levado ao conjunto de saídas, por intermédio de um conjunto de ocorrências internas à entidade.

A descrição das entradas e saídas pode representar, indistintamente, tanto canais de comunicação como estados característicos distintos e sucessivos da entidade.

A descrição das ocorrências corresponde à especificação do comportamento da entidade.

A agregação de um conjunto de identificadores à descrição do comportamento da entidade, pode tornar única sua especificação. Estes identificadores permitem distingui-la de outras entidades com o mesmo comportamento.

Uma representação gráfica do modelo básico de uma entidade é apresentada na figura 3.3.

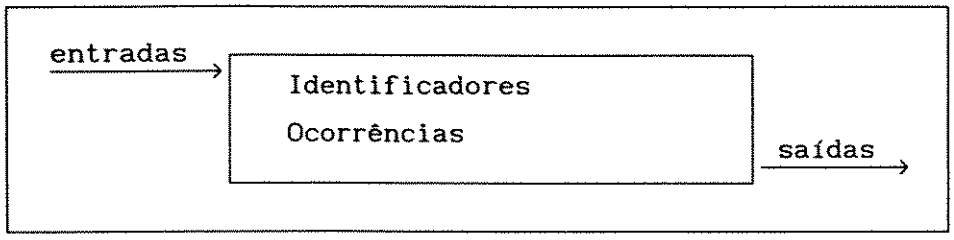


Fig. 3.3 - Modelo Abstrato Básico de uma Entidade

IDENTIFICADORES

Por identificadores de entidades, pode-se considerar um conjunto de informações que seja invariante para a entidade e um conjunto de informações que represente o estado da entidade em um dado instante.

O nome atribuído a uma entidade é um de seus principais identificadores. Outros identificadores invariantes, os mais variados possiveis, podem ser adicionados, contribuindo para a especificação da entidade.

A representação do estado de uma entidade pode ser obtida através do estabelecimento de alguns atributos aos quais com ele estejam relacionados. Estes atributos correspondem a variáveis às quais são agregados valores que podem ser alterados em função das ocorrências internas à entidade. Os atributos podem fazer parte do conjunto de identificadores da entidade, colaborando para uma identificação única desta, tanto em carater geral como a nível instantâneo. Assim, os nomes dos atributos correspondem a identificadores invariantes, enquanto os valores dos atributos representam o estado atual da entidade.

OCORRÊNCIAS

O conjunto de ocorrências internas à entidade pode ser especificado atra-

vés da descrição dos eventos que acontecem internamente a esta, e da ordenação temporal desses eventos. A descrição de um evento corresponde à apresentação de informações como seu nome e as consequências de sua ocorrência. A descrição da ordenação temporal dos eventos corresponde à indicação da forma como é sequenciada, no tempo, a ocorrência dos eventos.

Esta metodologia de representação de ocorrências é versátil o suficiente para descrever ocorrências internas à entidade por meio, tanto de eventos como de atividades [62]. Adotando-se eventos como elemento básico descritivo das ocorrências, a representação de atividades pode ser realizada através da indicação dos eventos início e fim de atividade.

Os eventos internos à entidade podem ser divididos em eventos físicos e eventos cognitivos [19].

Os eventos físicos correspondem àqueles relacionados com alguma ocorrência física instantânea, interna à entidade. O acontecimento de um evento físico pode provocar a alteração do estado da entidade, ou seja, pode provocar alterações nos seus atributos. A descrição da consequência da ocorrência de um evento físico pode ser realizada pela indicação de qual(is) atributo(s) da entidade deve(m) ser alterado(s) e de como deve ser efetuada esta alteração.

Ao método computacional responsável pela alteração dos atributos da entidade, consequência da ocorrência de um evento, denomina-se "Procedimento do evento". Assim, a cada evento físico corresponde um procedimento a ele agregado. Este procedimento indica quais os atributos da entidade devem ser alterados, e de que forma essa alteração deve ser processada, quando da ocorrência do evento. O conjunto de procedimentos relacionados aos eventos, pela sua composição, formam um outro tipo de informação, distinto dos identificadores.

Os eventos cognitivos são aqueles relacionados com a execução de inferências lógicas sobre conhecimento expresso em forma de regras. Inferências lógicas podem ser consideradas como atividades cognitivas, iniciadas e terminadas por eventos cognitivos. A conclusão de uma atividade cognitiva corresponde a uma tomada de decisão baseada em um conjunto de informações relativo à entidade, expresso por regras. Atividades cognitivas podem ser consideradas como de execução instantânea, e podem ser seguidas tanto por eventos cognitivos como físicos.

Pode-se citar, como exemplo de evento físico, o evento hipotético "fim da execução do furo 31". A conclusão de execução deste específico furo leva uma variável denominada "X", que represente o estado do sistema, ao valor 100.

Neste caso, um método computacional qualquer, que proceda à atribuição "X=100", corresponde ao procedimento do evento.

Um exemplo de evento congnitivo corresponde ao evento hipotético "habilita início do furo 31", ao qual relaciona-se uma regra que estipula "habilita início do furo 31 se peça na posição correta e ferramenta 5 na posição correta". A execução deste evento cognitivo dispara a inferência sobre a regra. O cumprimento das condições estabelecidas permitem a ocorrência do evento físico "início de execução do furo 31".

-Sequenciamento de ocorrências

O sequenciamento de ocorrências expressa a ordenação dos eventos da entidade. Este sequenciamento pode apresentar-se tanto de modo determinístico como condicional.

O sequenciamento determinístico indica que, após a ocorrência de um dado evento, outro determinado evento deva ocorrer, incondicionalmente. Esta sequência pode ser expressa por um conjunto ordenado de símbolos que represente a ordem em que os eventos devam ocorrer. O conjunto assim estabelecido pode ser incluído como mais um identificador da entidade.

O sequenciamento condicional relaciona a cada evento: um conjunto de outros eventos que o possam suceder e um conjunto de regras. A inferência sobre estas regras determina qual dentre os possíveis eventos sucessores deva ser o escolhido. A representação do sequenciamento condicional é expressa por um conjunto de símbolos que represente os eventos, por um indicador do conjunto de regras e pelo conjunto de regras. O conjunto de símbolos representativo dos eventos e o indicador do conjunto de regras podem ser considerados como mais um dos identificadores da entidade. O conjunto de regras, pela sua composição e tratamento diferenciado, são considerados como um outro tipo de informação, distinto dos identificadores e das regras.

TIPIFICAÇÃO DA INFORMAÇÃO NO MODELO ABSTRATO GENÉRICO

Em função do detalhamento dos identificadores e das ocorrências, reconheceu-se a necessidade de três tipos distintos de informação para a composição do modelo abstrato genérico descritivo de elementos da manufatura: identificadores, procedimentos e regras. Identificadores caracterizam e identificam, de forma única, cada entidade. Procedimentos expressam as alterações devido à ocorrência dos eventos físicos. Regras são utilizadas para expressar as condi-

cionalidades, sobre as quais são realizadas as inferências lógicas.

De uma maneira simplificada, pode-se representar a entidade abstrata como mostrado na figura 3.4.

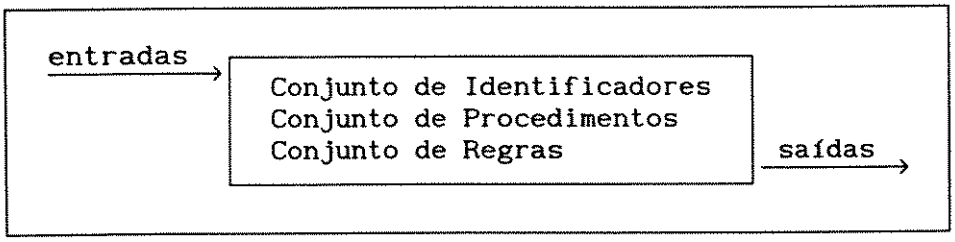


Fig. 3.4 - Modelo Abstrato genérico de uma Entidade

Através destes três tipos de dados, pode-se representar, de forma genérica, os elementos e atividades que compõem a manufatura [75, 149, 152].

b) O modelo abstrato genérico hierarquizado de informação

A composição do modelo abstrato genérico hierarquizado de informação faz-se através de considerações de hierarquização, hereditariedade e integridade das informações consideradas no modelo abstrato genérico de informação.

HIERARQUIZAÇÃO

A hierarquização da informação a ser considerada para elementos de manufatura pode ser classificada em dois tipos: hierarquia física e hierarquia funcional [7, 75].

Por hierarquização física, entende-se a visualização estratificada da manufatura, a nível de instalações físicas. Nesta visualização, uma companhia pode ser detalhada físicamente em subsequentes níveis hierárquicos como: Corporações, Divisões, Fábricas, Sistemas Flexíveis de Manufatura, Células flexíveis de manufatura e máquinas. Esta estratificação apresenta-se com caráter composicional, isto é, os elementos de nível superior são compostos físicamente pelos elementos de nível inferior, conforme representado pela figura 3.5.

A hierarquização funcional corresponde a uma visão estratificada da manufatura a nível de poder decisório. Neste caso, pode-se citar como elemento componente de cada nível hierárquico funcional, relacionado com a hierarquia

física acima citada, os seguintes: Presidente da Corporação, Diretores de Divisão, Gerente de Fábrica, Supervisor de Sistema de Manufatura, Controlador de Célula de Manufatura e Operador de Máquina. Esta estratificação corresponde a uma composição em forma de árvore, conforme representado pela figura 3.6.

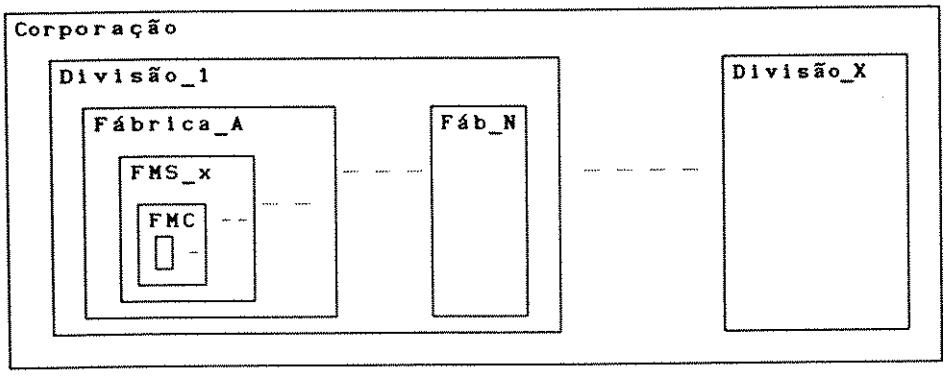


Fig. 3.5 - Representação de Hierarquia Física

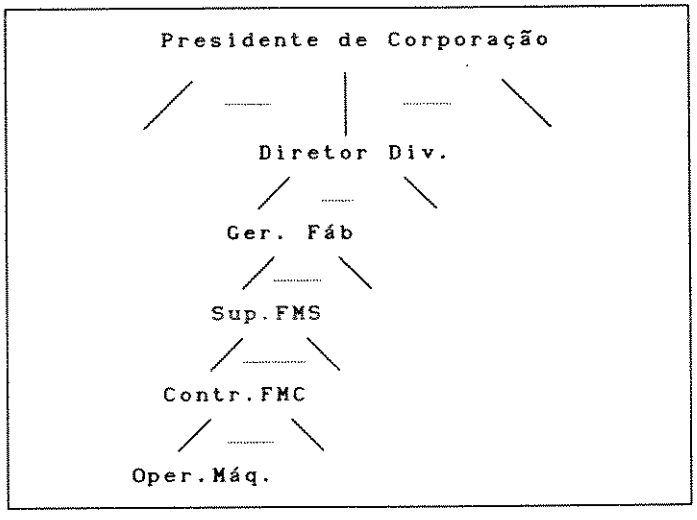


Fig. 3.6 - Representação de Hierarquia Funcional

O desenvolvimento de um modelo abstrato hierarquizado, para especificar entidades de manufatura, capaz de representar as diferentes formas de hierarquia, pôde ser iniciado a partir da aplicação do requisito de hierarquia de sistema [7, 14, 75] sobre o modelo abstrato genérico, anteriormente proposto. Requisito de hierarquia de sistema: "Todo objeto (sistema) pode ser considerado como um subsistema de algum outro de mais alto nível".

Como forma identificadora, pode-se denominar o objeto de mais alto nível hierárquico de "objeto-pai", e o de mais baixo nível de "objeto-filho".

Este requisito é útil para a elaboração sistemática de uma especificação hierarquizada de um sistema abstrato. Com ele, a especificação de um sistema torna-se um processo recursivo, que começa em um objeto básico, visto como uma "caixa preta".

Esta "caixa preta" possui um conjunto de entradas que, após o acontecimento de certas ocorrências, leva a um conjunto de saídas. Internamente ao objeto-pai, cada ocorrência pode ser detalhada em termos de um novo objeto-filho. Por sua vez, os objetos-filho que o compõem podem ser detalhados em termos de outros objetos de mais baixo nível hierárquico, e assim por diante. Cada entrada do objeto-pai deve ser assumida por um objeto-filho, que a transforma em uma saída interna. Esta saída será considerada como entrada de outro objeto-filho, e assim sucessivamente, até que um objeto-filho tenha por saída uma das saídas definidas para o objeto-pai.

Note-se que esta hierarquização apresenta um caráter de hierarquia funcional, já que cada ocorrência pode ser detalhada em novos objetos de um nível inferior. No entanto, dada a completa abstração do modelo genérico assumido, pode-se também usar o mesmo princípio de detalhamento para a representação de uma hierarquização física.

A representação da hierarquia física inicia-se por considerar que cada objeto pode ser decomposto em outros objetos de nível hierárquico inferior. O conjunto de ocorrências do objeto-pai é desempenhado globalmente pelo conjunto dos objetos-filho que o compõem. As entradas são partilhadas por estes objetos-filho, sendo levadas, de forma global, por todos eles, às saídas do objeto-pai.

Em ambos os casos de detalhamento hierárquico, existe a necessidade de estabelecer-se uma relação de transformação entre um nível hierárquico e outro. Esta necessidade é satisfeita ao serem indicadas as relações entre ocorrências do objeto-pai para com os objetos-filho, e as relações de entradas (e saídas) entre o objeto-pai e os objetos-filho.

No caso de uma especificação com hierarquização funcional, existe um detalhamento a nível de ocorrências. Cada entrada do objeto-pai é assumida por um objeto-filho. Cada objeto-filho responde pelo detalhamento de uma ocorrência do nível hierárquico superior, cumprindo o detalhamento comportamental desse nível. Ainda, cada saída do objeto-pai é desenvolvida por um objeto-

filho. Com relação ao interrelacionamento entre objetos-filho, a saída de um corresponde à entrada de outro, completando, assim, o sequenciamento de ocorrências do nível hierárquico superior. A figura 3.7 representa o modelo abstrato de informação de entidade, com especificação hierárquica funcional.

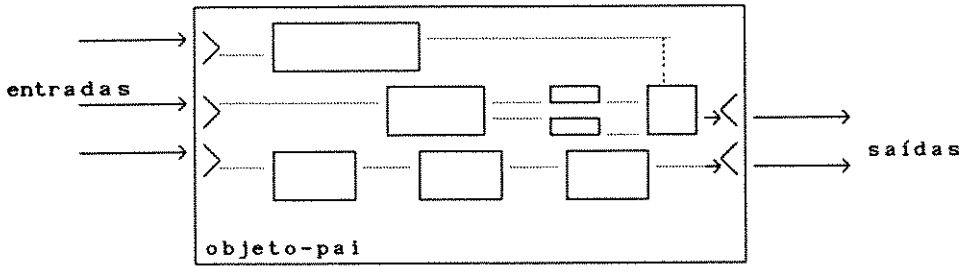


Fig. 3.7 - Modelo abstrato com hierarquia funcional

No caso de uma especificação com hierarquização física, existe uma nova composição de entradas-ocorrências-saídas para o nível hierárquico inferior. Cada entrada do nível superior pode ser partilhada por vários objetos-filho, bem como cada saída do nível superior pode ser composta pelas saídas de vários objetos-filho. As ocorrências do nível hierárquico superior não são simplesmente detalhadas, mas sim recompostas a partir das ocorrências do conjunto de objetos-filho que compõem o nível inferior. A figura 3.8 representa o modelo abstrato de informação de entidade, com especificação hierárquica física.

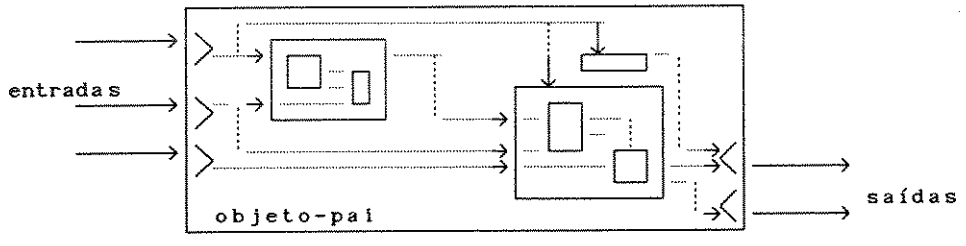


Fig. 3.8 - Modelo abstrato com hierarquia física

HEREDITARIEDADE

A especificação de hereditariedade de entradas e saídas é realizada de maneiras diferentes, para o caso de hierarquia funcional e física. No caso de hereditariedade por hierarquia funcional, o relacionamento de entrada-pai/entrada-filho é um indicativo de igualdade. No caso de hereditariedade por

hierarquia física, esta relação é expressa por uma indicação de partilha de uma entrada de mais alto nível com várias entradas de objetos-filho.

Para que haja integridade das especificações hierárquicas, quanto à hereditariedade, deve-se respeitar os requisitos estruturais denominados descritivo de objeto e relacional de objetos [14]:

Requisito descritivo de objeto: "Entradas e saídas de um subsistema devem ser definidas *a priori* em seu sistema-pai". Isto estabelece uma relação de segurança, garantindo que as entradas do objeto-pai sejam completamente levadas pelos objetos-filho, às saídas daquele. Garante-se, também, que as saídas (entradas) de cada objeto-filho estejam completamente relacionadas com as entradas (saídas) de outros objetos-filho de mesmo nível hierárquico. Com isto, nenhuma entrada, ou saída, de objetos-filho fica sem uma relação expressa de conexão, com o mesmo nível hierárquico ou com o superior.

Requisito relacional de objetos: "Todas as saídas de um objeto, ou sistema, devem ter um mapeamento entrada- saída". Com isto fica garantido que todas as ocorrências do objeto-pai sejam completamente representadas pelo conjunto de ocorrências dos objetos-filho.

INTEGRAÇÃO

A garantia de integração das especificações dos objetos-filho fica estabelecida ao impor-se o respeito ao requisito de completude [14].

Requisito de Completude: "Um sistema está completamente especificado se cada subsistema está completamente especificado".

O respeito a todos estes requisitos permite a composição de um sistema completamente especificado, através da especificação dos objetos-filho ou subsistemas.

A especificação hierárquica acima descrita pode ser utilizada indefinidamente, tanto no sentido de detalhamento como no de generalização, obtendo-se o nível desejado de especificação.

Em ambos os casos de especificação hierarquizada, os objetos-filho podem ser especificados segundo o mesmo modelo abstrato de entidade utilizado para a especificação do objeto-pai. Assim, o mesmo modelo de informação pode ser usado para a especificação de qualquer objeto, em qualquer nível, já que este modelo independe do nível de detalhamento assumido.

3.3-2 Especificação básica de entidades

O Ambiente Integrado de Apoio à Tomada de Decisão (AIATD) considera especificações de entidades expressas, tanto de maneira informal declarativa como de maneira formal.

Uma especificação formal, baseada em lógica temporal, apresenta um grande potencial para a descrição comportamental de sistemas [38, 108, 171]. Uma especificação informal declarativa pode ser, a princípio, utilizada, tanto para a descrição comportamental de sistemas como para a composição de bases de conhecimento sobre as quais sejam realizadas inferências lógicas, através de processamento simbólico [4, 27, 36, 157].

Especificações declarativas, quando passíveis de processamento simbólico, cumprem dois requisitos de interesse do AIATD:

a) podem ser mais facilmente realizadas por usuários com pouca experiência em modelagem para simulação, ou em criação de bases de conhecimento para inferências lógicas,

b) podem ser igualmente utilizadas em simulações, em inferências lógicas e, principalmente, no intercâmbio de conhecimento adquirido em simulações para uso em inferências lógicas ou vice-versa.

Devido à possibilidade de ser utilizada em ambas as ferramentas de análise, disponíveis no AIATD, a especificação informal declarativa foi a escolhida como tipo básico de especificação, a ser usada operacionalmente nas ferramentas do ambiente.

As especificações formais baseadas em Lógica Temporal foram escolhidas como ferramenta de especificação de modelos comportamentais formais para simulação de entidades da manufatura. Sua principal utilização, no ambiente, é vista como sendo a descrição concisa e formal do comportamento de entidades da manufatura, em situações em que estas necessitem de uma prova formal em relação ao cumprimento de requisitos de existência, segurança e completude de seu comportamento. Como um dos principais exemplos dessa situação, destaca-se a necessidade de prova formal de comportamento de controladores a eventos discretos, muito usados em ambientes de manufatura.

Optou-se, também, por uma utilização cooperativa entre os dois tipos de especificação, internamente ao ambiente. Isto deve ocorrer no caso em que se necessite de provas formais do comportamento das entidades sob desenvolvimento.

Este processo de operação cooperativa inicia-se com a especificação for-

mal da entidade, utilizando-se da Lógica Temporal de Tempo Real Generalizada [163], considerada adiante. Após isto, as especificações são transformadas em modelos informais, sobre os quais serão realizadas simulações. Devido ao caráter informal e declarativo das especificações utilizadas na simulação, estas podem ser facilmente acompanhadas pelo usuário, inclusive propondo alterações na especificação comportamental da entidade, visando aperfeiçoamentos ou correções. As simulações permitem que se suponha o cumprimento dos requisitos de existência, e de alguns requisitos de segurança, da especificação da entidade. Após as correções e novas simulações, as especificações são novamente convertidas para a maneira formal, procedendo-se à sua prova formal. Com o uso da simulação como análise intermediária, evita-se o dispêndio de tempo e esforço na tentativa de prova de especificações que se mostrariam não corretas.

Em qualquer situação, ambos os tipos de especificação devem adequar-se ao modelo abstrato hierarquizado de informação acima discutido. Isto quer dizer que se deve descrever cada entidade em termos de identificadores, procedimentos e regras. Como identificadores de cada entidade devem ser apresentadas as informações relativas a seu nome, nome e características de suas entradas e saídas, atributos que expressam seu estado, relação de seus eventos internos e sequenciamento dos eventos. Os procedimentos indicam as consequências da ocorrência de cada evento. As regras expressam as condicionantes relacionadas com a entidade. Estas condicionantes podem expressar tanto a possibilidade de ocorrência de um evento como as consequências lógicas de sua ocorrência.

A especificação de entidades da manufatura, na forma declarativa, como discutida no modelo abstrato hierárquico de informação, cria módulos independentes de conhecimento, para os quais são consideradas ligações hierárquicas. Uma representação de conhecimento através de "Frames" [95, 96, 101, 120] também adota a estruturação de módulos independentes de conhecimento.

Um "Frame" corresponde a uma estrutura de representação de conhecimento composta por Slots que são unidades elementares, onde são armazenadas as informações. Dentro dos slots, podem ser armazenados diferentes tipos de informação, ou outro slot chamado de sub-slot e até outro frame. Em geral, a estrutura baseada em frames é uma rede onde os nós representam conceitos (descritos por seus slots), e os arcos indicam o relacionamento entre estes conceitos. Aqui, apresenta-se embutido o sentido de hierarquia e herança, ou seja, os nós inferiores herdam as propriedades dos nós hierarquicamente superiores.

Com esta estrutura de armazenagem de conhecimento, pode-se obter uma

representação declarativa proposta no modelo abstrato hierárquico de informação.

Em um frame que represente uma entidade, cada slot pode estar relacionado com um entre os tipos de informação considerados: identificadores, procedimentos, ou regras. A possibilidade de armazenar-se diferentes tipos de informação nos slots facilita a modularidade da especificação de entidades, já que se podem criar módulos completos de conhecimento específico, a serem posteriormente combinados com outros, ou destes receberem comunicações.

A representação de uma estrutura hierárquica, como proposto no modelo abstrato hierárquico de informação, pode ser conseguido através da composição de árvores de frames. Aqui se apresenta uma das grandes vantagens do uso da representação do conhecimento através de frames, uma vez que se pode manter a modularidade de cada entidade ao representar sua composição hierárquica, independentemente do nível hierárquico representado.

Ao ser adotada a representação do conhecimento contido no modelo abstrato hierárquico de informação através de frames, passa-se a ter três estruturas abstratas de informação, a serem manipuladas no ambiente: frames, procedimentos e regras. Os identificadores serão a informação estática contida no frame, enquanto procedimentos e regras, com indicadores também armazenados nos frames, correspondem à representação da parte dinâmica do conhecimento contido nas especificações da entidade.

3.4 - MODELO FUNCIONAL DO AMBIENTE

De acordo com modernas tendências de especificação de sistemas, a concepção do modelo funcional do AIATD seguiu o paradigma de objetos, de maneira "top-down", ou seja, em um processo de detalhamento modular sucessivo [18, 19, 49, 75, 180].

O AIATD tem como função a realização de operações de apoio à tomada de decisão. Seu domínio de aplicação é composto pelo universo dos sistemas de manufatura, passíveis de serem descritos por eventos discretos. Seu contradomínio corresponde a um conjunto de simulações, previsões, análises de comportamento e inferências lógicas sobre o conhecimento detido a respeito do sistema em estudo.

A estruturação funcional do AIATD (Fig. 3.9) compõe-se de seis objetos funcionais primários, ou de mais alto nível: Interface de Comunicação, Gerenciador de Sistema, Aquisitor de Conhecimento, Simulador, Processador de Conhecimento e Base de Conhecimento. Cada um destes objetos funcionais primários decompõe-se em outros objetos funcionais de menor nível, denominados secundários e terciários, individualmente definidos e descritos abaixo.

3.4-1 - Objetos Funcionais Primários

O Ambiente Integrado de Apoio à Tomada de Decisão - AIATD - é composto pelos objetos funcionais: Interface de Comunicação, Gerenciador de Sistema, Aquisitor de Conhecimento, Simulador, Processador de Conhecimento e Base de Conhecimento. A interação destes objetos funcionais forma um todo, chamado de modelo funcional do AIATD, conforme representado na figura 3.9.

a) Interface de Comunicação

A Interface de Comunicação é o objeto responsável pela execução das atividades de conexão, manutenção e desconexão de comunicações entre o AIATD e seu universo de comunicação. Como universo de comunicação entende-se o conjunto de elementos geradores ou receptores de informações que interagem com o ambiente. Aí, enquadram-se tanto usuários humanos como equipamentos remotos.

A comunicação do AIATD com os elementos externos a ele deve realizar-se através de uma rede de comunicação de computadores à qual esteja conectado. A

Interface de Comunicação incumbe-se da compreensão sintática das mensagens provenientes da rede de comunicação, respeitados os padrões estabelecidos pelos protocolos de comunicação aceitos pelo ambiente. Também incumbe-se do tratamento sintático de mensagens a serem transmitidas do AIATD para a rede, realizando as operações requeridas pelos protocolos de comunicação.

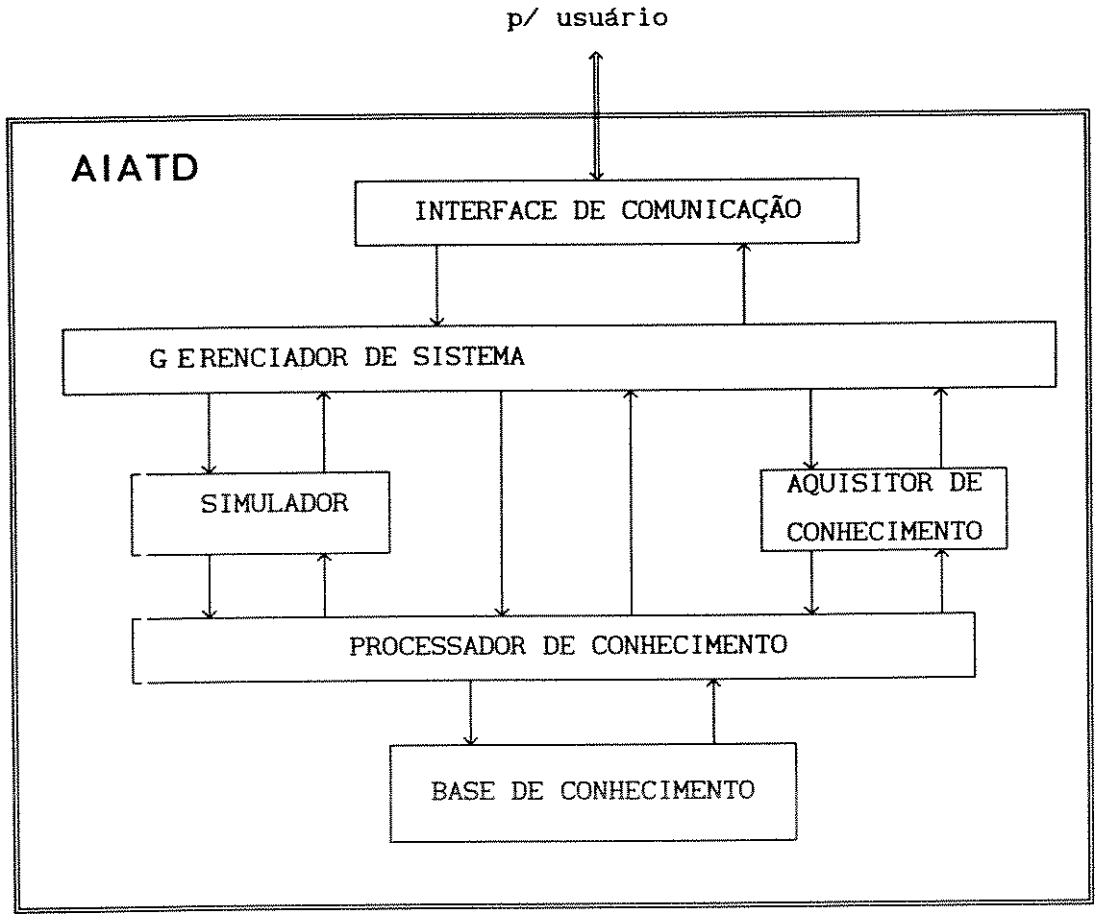


Fig. 3.9 - Modelo funcional do Ambiente Integrado de Apoio à Tomada de Decisão - AIATD

A comunicação destes elementos com o AIATD tanto pode ter caráter interativo bidirecional como caráter unidirecional. Por comunicação bidirecional, entende-se aquela em que um usuário humano interaja com o ambiente, tanto para gerar como para receber informações. Como comunicação unidirecional, entende-se aquela realizada com um elemento da rede que apenas envie informações de atualização para o ambiente, com, ou sem, solicitação explícita.

Internamente ao AIATD, esta interface comunica-se com o Gerenciador de

Sistema, transmitindo mensagens chegadas através da rede de comunicação, ou recebendo mensagens a serem transmitidas para a rede de comunicação de computadores.

b) Gerenciador de Sistema

O Gerenciador de Sistema é o objeto responsável pelo controle de funcionamento de todo o ambiente. Sua principal função é a ativação ordenada de cada objeto componente do ambiente.

A ativação de cada objeto é decidida e efetuada, a partir de comunicações recebidas de um usuário ou de algum dos outros objetos que compõem o ambiente.

Para o caso em que o AIATD opere como um servidor de informações, sendo acessado concorrentemente por vários usuários, este gerenciador deve manter e controlar cada acesso. Neste caso torna-se responsável pela criação de diretorias de acesso, bem como pelo controle do estado atual de cada acesso e do ambiente como um todo.

c) Simulador

O Simulador é o objeto funcional responsável pela execução de simulações a eventos discretos. Opera sobre modelos de simulação gerados de forma declarativa e armazenados na Base de Conhecimento. Os resultados podem ser apresentados tanto em tempo de execução como de forma pós-processada, todos, no entanto, de forma configurável pelo usuário. Como tipos possíveis de resultados tem-se: relatórios descritivos de análise de variáveis definidas pelo usuário; animação gráfica; conjuntos de informações passíveis de serem agregadas à Base de Conhecimento na qualidade de conhecimento dinâmico; e um histórico da simulação, listando todos os eventos ocorridos agregados à sua data de ocorrência.

Em função do modelo abstrato de informação adotado, o Simulador opera íntimamente relacionado com o Processador de Conhecimento, ordenando manipulações de conhecimento segundo uma sequência denominada Lógica de Simulação.

A Lógica de Simulação adotada permite classificar a operação do Simulador como: simulação discreta, orientada a eventos, com estratégia de ordenação de ocorrências do tipo de interação entre processos e atualização de relógio de simulação dirigido a eventos.

d) Aquisitor de Conhecimento

O Aquisitor de Conhecimento é o objeto funcional responsável por todo o trabalho de diálogo com o usuário, quando do processo de aquisição de conhecimento para a composição do conteúdo da Base de Conhecimento. Como aquisição de conhecimento, entende-se, tanto a entrada de conhecimento original como a alteração de conhecimento já detido pela base de conhecimento disponível.

A entrada ou alteração de conhecimento pode ser realizada através de uma especificação formal ou de forma declarativa. A especificação formal adotada segue o formalismo da GRTTL [163]. Este formalismo é usado apenas para a criação de modelos para simulação. A forma declarativa de entrada de conhecimento adota o modelo abstrato de informação descrito acima. Esta forma é usada tanto para a criação de modelos para simulação como para a criação de módulos de conhecimento a serem trabalhados pelo Processador de Conhecimento.

O Aquisitor de Conhecimento incumbe-se, ainda, de realizar: a montagem ou alteração da estrutura de conhecimento declarativo a ser usada; a ativação de métodos computacionais que avaliem questões de segurança e integridade do conhecimento a ser alterado; a conversão de modelos formais de simulação para modelos declarativos, a verificação de consistência e validade de modelos declarativos de simulação. Todo conhecimento adquirido através do Aquisitor é encaminhado ao Processador de Conhecimento, para que se desenvolva a devida armazenagem.

e) Processador de Conhecimento

O Processador de Conhecimento é o objeto funcional responsável por toda a manipulação do conhecimento detido pelo ambiente. Como manipulação de conhecimento, entendem-se os métodos relativos a armazenagem, recuperação e utilização de conhecimento.

Em face do modelo abstrato de informação adotado, esta manipulação é feita de forma tipificada segundo as classes Frames, Regras e Procedimentos.

A manipulação de Frames corresponde à criação e manutenção de estruturas de dados do tipo "Frame", execução de métodos computacionais para escrita, leitura, busca e verificação de consistência de informações agregadas a estas estruturas.

A manipulação de regras consiste em criar, utilizar e destruir conjuntos de regras de produção. Como criação de regras, entende-se a composição destas, em forma configurável pelo usuário. A configuração corresponde à possibilidade

de definição, pelo usuário, de: operadores sobre regras e seus respectivos níveis de prioridade; definição de métodos computacionais a serem usados na avaliação de regras; definição de relações entre os operadores, os métodos computacionais e os tipos de informação aos quais se aplicam.

Por utilização de regras, entende-se a execução das avaliações sintática e semântica das regras, realizando-se inferências lógicas sobre conjuntos de regras e dados contidos nos frames. Note-se que também os métodos de inferência lógica sobre as regras podem ser configurados pelo usuário.

A destruição de regras ou conjuntos de regras pode ocorrer por interação com o usuário, através da rede de comunicação, ou mesmo como consequência do resultado de certas inferências lógicas, executadas pelo Processador de Conhecimento.

A manipulação de Procedimentos consiste na criação, armazenagem, utilização e eliminação de métodos computacionais que operem sobre os frames, regras, ou que interajam com elementos externos ao Processador de Conhecimento.

O Processador de Conhecimento pode, alternativamente, operar com diferentes estruturas e metodologias de inferência. Uma Máquina de Inferência Generalizada, configurável pelo usuário, apresenta-se disponível, internamente ao Procesador de Conhecimento. Núcleos Dedicados de Inferência podem ser agregados ao Processador de Conhecimento e utilizados para a execução de inferências particulares, com estrutura e metodologia específicas e dedicadas.

f) Base de Conhecimento

A Base de Conhecimento é o objeto funcional responsável pela armazenagem de todo o conhecimento utilizado pelo AIATD.

Esta base trabalha com conhecimento estruturado de acordo com o modelo abstrato de informação descrito acima, ou seja, tem como tipos abstratos de dados: frames, regras e procedimentos. No entanto, a estruturação destes tipos de dados pode ser configurada pelo usuário, o que muito facilita a operação do ambiente.

3.4-2 - Objetos Funcionais Secundários

Dentre os objetos funcionais primários, acima descritos, apenas o Gerenciador de Sistema opera em um só nível hierárquico funcional. Todos os outros são compostos por objetos funcionais secundários, definidos e descritos abaixo.

A interação destes objetos funcionais secundários compõe a funcionalidade de cada objeto funcional primário.

3.4-2.1 - Interface de Comunicação

A Interface de Comunicação compõe-se de três objetos funcionais secundários: Gerenciador de Comunicações, Controlador de Comunicação Interativa e Controlador de Comunicação por Pacotes. O modelo funcional da Interface de Comunicação é apresentado na figura 3.10.

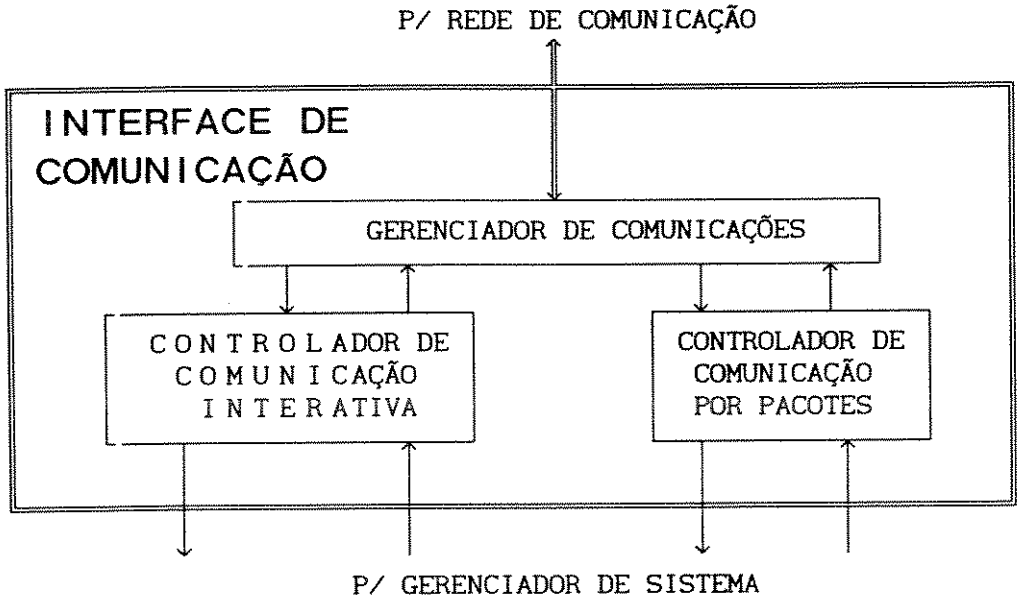


Fig. 3.10 - Modelo funcional da Interface de Comunicação

a) Gerenciador de Comunicações

O Gerenciador de Comunicações é o objeto funcional responsável pelo gerenciamento do fluxo de informações que chegam e saem do AIATD. Responsabiliza-se, também, pelo estabelecimento e manutenção da comunicação do AIATD com o mundo exterior ao ambiente, através da rede de comunicação de computadores. É

a este nível funcional que estão os serviços responsáveis pelas funções de comunicação com a Camada de Aplicação do Modelo de Referência para Interconexão de Sistemas Abertos RM-OSI/ISO [44, 52, 66, 67, 167].

Internamente ao Gerenciador de Comunicações se realiza um tratamento inicial das informações chegadas da rede de comunicação. Estas informações são filtradas e direcionadas ao Controlador de Comunicação adequado. As comunicações com usuários humanos são encaminhadas para o Controlador de Comunicação Interativa, enquanto as comunicações com elementos fornecedores de dados são encaminhadas ao Controlador de Comunicação por Pacotes.

b) Controlador de Comunicação Interativa

O Controlador de Comunicação Interativa é o objeto funcional responsável pelo controle de toda comunicação entre o AIATD e usuários humanos. Aqui são desenvolvidos todos os processos de formatação gráfica da comunicação com o usuário, realizando, também, o devido tratamento sintático de informações provenientes do usuário, de forma que estas possam ser devidamente processadas internamente.

c) Controlador de Comunicação por Pacotes

Este controlador é o objeto funcional responsável pela emissão de requisições a serem feitas para elementos da manufatura que estejam conectados ao AIATD com o propósito de fornecer dados em tempo real. Encarrega-se, também, do recebimento das informações por eles enviadas, sem a intervenção explícita do usuário ou do método computacional que venha a utilizar tal informação.

Comunicações deste tipo ocorrem quando o AIATD se encontra totalmente integrado à manufatura, sendo capaz de solicitar e receber atualizações a respeito do estado atual de diferentes unidades da manufatura. Esta comunicação desenvolve-se essencialmente por meio de envio de blocos de dados, contendo informações das unidades de manufatura para o AIATD, sendo estas, então, tratadas e encaminhadas para a base de conhecimento do ambiente.

3.4-2.2 - Gerenciador de Sistema

O Gerenciador de Sistema realiza todas as suas funções a um mesmo nível hierárquico funcional, controlando a operação de todos os outros objetos fun-

cionais a nível primário.

3.4-2.3 - Simulador

O Simulador realiza as simulações a eventos discretos. Este é composto por três objetos funcionais secundários: Interface do Simulador, Executor de Simulação e Analisador de Resultados. O modelo funcional do Simulador é apresentado na figura 3.11.

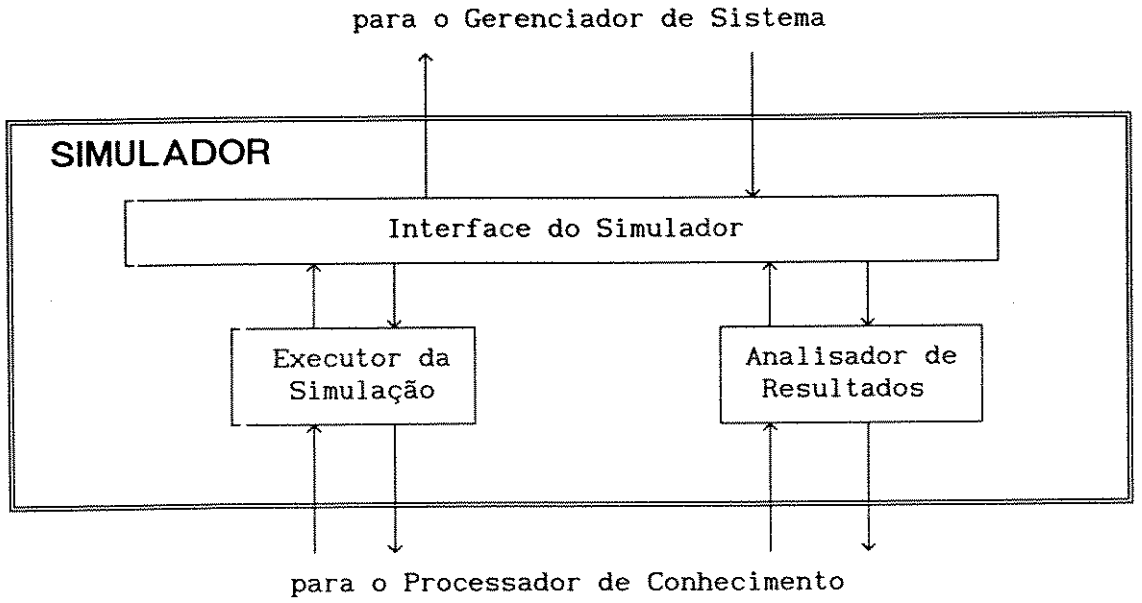


Fig. 3.11 - Modelo funcional do Simulador

a) Interface do Simulador

A Interface do Simulador é o objeto funcional responsável pela recepção das mensagens enviadas pelo Gerenciador de Sistema. Após o reconhecimento da mensagem, destina-a para o Executor de Simulação ou Analisador de Resultados. Também incumbe-se de receber o retorno de mensagens enviadas por estes, endereçando-as ao Gerenciador de Sistema. A atuação desta interface torna-se intensa no caso de utilização concorrente do Simulador, por vários usuários.

b) Executor da Simulação

O Executor de Simulação é o objeto funcional responsável pela execução de toda a Lógica de Simulação.

A Lógica de Simulação adotada é caracterizada, principalmente, por desenvolver uma simulação a eventos discretos, do tipo de Interação de Processos, adequada ao modelo abstrato de informação adotado.

A execução da simulação ocorre em três fases: ordenação de ocorrências, atualização de relógio e do histórico da simulação ("trace") e execução dos eventos.

A ordenação de ocorrências acontece em duas fases. Na primeira, trata-se cada entidade que componha o modelo de simulação, como um elemento independente. Nesta fase da ordenação, leva-se em conta apenas informações a respeito da própria entidade, não considerando sua interação com outras entidades. Na segunda fase, consideram-se as informações a respeito da interação entre todas as entidades que compõem o modelo de simulação. Nesta fase, confirma-se, ou não, uma ocorrência escolhida durante a primeira fase. Só após ela é que se compõe o conjunto de eventos passíveis de ocorrência para o dado instante de simulação, referente a todas as entidades do modelo de simulação.

Após a escolha do conjunto de eventos passíveis de ocorrência para um certo instante de simulação, procede-se à atualização do relógio de simulação para o valor do instante considerado e à inclusão da lista destes eventos em um histórico da simulação, agregando-se a eles o instante de execução (valor do tempo de simulação).

Por fim, procede-se à execução dos eventos escolhidos. Entende-se por execução dos eventos a atualização do estado atual do modelo de simulação, que corresponde à execução dos métodos computacionais agregados a cada um dos eventos escolhidos.

c) Analisador de Resultados

O Analisador de Resultados é o objeto funcional responsável por prover a devida análise do conjunto de resultados gerados pelo Simulador. Uma vez que o Simulador apresenta como saída um histórico da simulação (trace), é sobre essa sequência de eventos que o Analisador de Resultados atua.

Devido ao tipo de informações a que este objeto tem acesso, é possível desenvolver-se uma grande variedade de tipos de análise da simulação. Isto,

porque é possível programar-se os mais variados tipos de relacionamentos entre os eventos ocorridos na simulação e disponíveis no histórico da simulação. Assim a operação do Analisador de Resultados torna-se configurável, já que este atua de forma a apresentar as análises desejadas pelo usuário, em forma e conteúdo. Para facilidade de operação do usuário, este é dotado de rotinas pré-programadas para a realização de análises mais comuns e de maior uso. Entre estas encontram-se relatórios e alguns tipos básicos de animação.

3.4-2.4 - Aquisitor de Conhecimento

O Aquisitor de Conhecimento é o objeto funcional responsável por todo o trabalho de diálogo com o usuário, quando do processo de aquisição de conhecimento para a composição do conteúdo da Base de Conhecimento. É composto por cinco objetos funcionais: Gerenciador de Aquisição de Conhecimento, Aquisitor de Conhecimento Formal, Aquisitor de Conhecimento Declarativo, Conversor de Conhecimento Formal/Declarativo e Interface com Processador de Conhecimento. Seu modelo funcional é apresentado na figura 3.12.

a) Gerenciador de Aquisição de Conhecimento

O Gerenciador de Aquisição de Conhecimento é o objeto funcional responsável por controlar a operação dos outros objetos funcionais componentes do Aquisitor de Conhecimento, com exceção da Interface com o Processador de Conhecimento, que é ativada diretamente pelo objeto interessado em comunicação com o Processador de Conhecimento.

Sua operação é ativada através do Gerenciador de Sistema que o coloca em contato com um usuário. Após a definição de parâmetros de aquisição de conhecimento, este gerenciador abre uma sessão de aquisição de conhecimento ao ativar um dos dois aquisitores ou o conversor. A partir daí sua tarefa corresponde a coordenar as diversas comunicações entre os objetos aquisitores e os usuários.

b) Aquisitor de Conhecimento Formal

O Aquisitor de Conhecimento formal é o objeto funcional responsável por executar a aquisição de conhecimento, segundo o formalismo estabelecido pela

Lógica Temporal Generalizada de Tempo Real (GRTTL) [100, 163]. Este tipo de conhecimento é usado apenas para a criação e edição de modelos formais de entidades e sistemas a serem simulados. Tais modelos devem, posteriormente, ser convertidos para a forma declarativa aceita pelo Simulador.

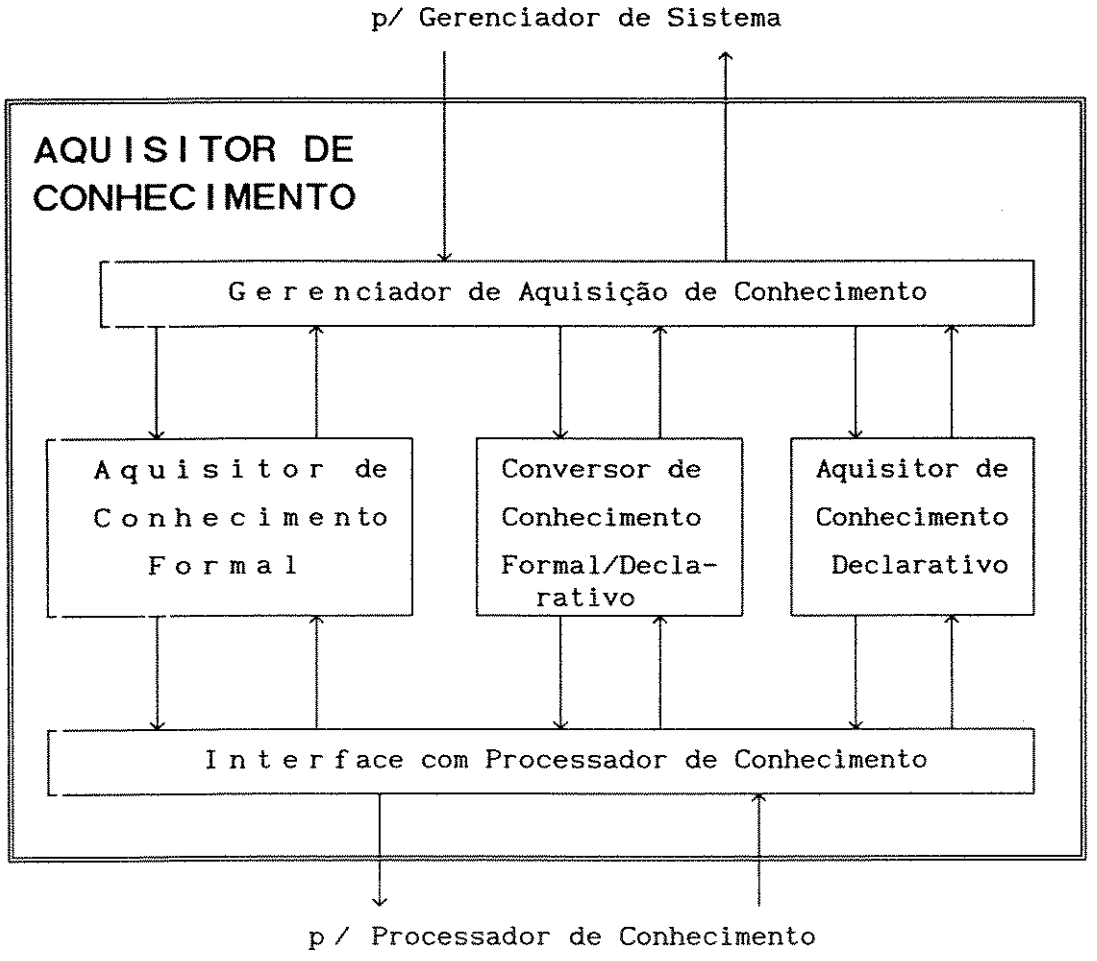


Fig. 3.12 - Modelo funcional do Aquisitor de Conhecimento

Através deste aquisitor, estabelece-se um diálogo com o usuário. Inicialmente o aquisitor requer do usuário a apresentação de parâmetros característicos do modelo a ser criado ou editado, além de senhas de segurança que para tal o habilitem. Após a verificação de questões de segurança, dá continuidade ao diálogo com o usuário, editando o modelo de simulação. Após a edição de cada expressão em GRTTL, o aquisitor procede a uma verificação de consistência da mesma, consultando uma base de informações relativas à lógica usada, armazenada na Base de Conhecimento. Ao término da edição do modelo de simulação,

este aquisitor encaminha-o à Base de Conhecimento para armazenagem.

c) Aquisitor de Conhecimento Declarativo

O Aquisitor de Conhecimento Declarativo é o objeto funcional responsável por executar a aquisição de conhecimento, segundo o modelo abstrato de informação acima descrito. Este tipo de conhecimento é usado para a criação e edição, tanto de modelos declarativos de entidades e sistemas a serem simulados como de módulos de conhecimento a serem trabalhados pelo Processador de Conhecimento.

Através deste aquisitor, estabelece-se outro tipo de diálogo com o usuário. Inicialmente o aquisitor requer do usuário a apresentação de parâmetros característicos do modelo de simulação ou do módulo de conhecimento a ser editado, além das senhas de segurança. Após a verificação de questões de segurança, dá continuidade ao diálogo com o usuário, requerendo a entrada da tipificação e estrutura das informações a serem trabalhadas, de acordo com o modelo de informação aceito pelo ambiente. Posteriormente procede à edição requerida.

Após a edição de cada entidade, ou módulo de conhecimento, o aquisitor procede a uma verificação de sua consistência, consultando uma base de informações relativas à estrutura e tipo de dados usados, armazenada na Base de Conhecimento. Ao término da edição do modelo de simulação, este aquisitor encaminha-o à Base de Conhecimento para armazenagem.

d) Conversor de Conhecimento Formal/Declarativo

O Conversor de Conhecimento é o objeto funcional responsável por realizar a conversão de modelos de simulação, gerados através do Aquisitor de Conhecimento Formal. Uma vez que o Simulador apenas executa simulações sobre modelos de simulação apresentados em forma declarativa, este conversor auxilia na transformação de modelos formais em modelos declarativos. O modelo gerado pelo conversor é encaminhado ao Procesador de Conhecimento para armazenagem.

A modularidade deste conversor permite que futuras alterações em uma ou outra forma de aquisição de conhecimento possam ser mais facilmente adaptadas para utilização com o Simulador e com o Processador de Conhecimento em operação no ambiente.

e) Interface com o Processador de Conhecimento

A Interface com o Processador de Conhecimento é o objeto funcional responsável por realizar uma adequação de formato do conhecimento adquirido pelos aquisitores.

Esta formatação é necessária uma vez que o Processador de Conhecimento tem uma formatação própria para o tratamento do conhecimento. No entanto esta formatação não se apresenta de forma amigável ao usuário. Tendo em vista este aspecto, adota-se uma formatação distinta para o trabalho dos aquisitores de conhecimento. A operação desta interface é justamente no sentido de realizar a conversão entre os dois formatos de conhecimento.

3.4-2.5 - Processador de Conhecimento

O Processador de Conhecimento é o objeto funcional responsável por toda a manipulação do conhecimento detido pelo ambiente. Compõe-se de quatro objetos funcionais secundários: Supervisor de Manipulação, Manipulador de Frames, Manipulador de Regras e Manipulador de Procedimentos. O modelo funcional do Processador de Conhecimento é apresentado na figura 3.13.

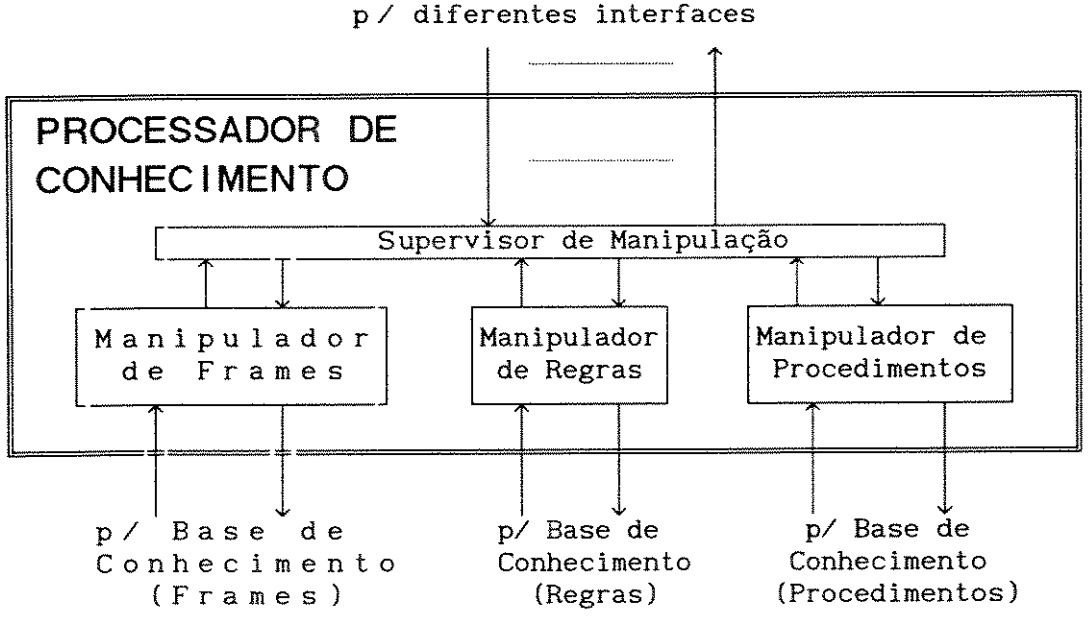


Fig. 3.13 - Modelo Funcional do Processador de Conhecimento

a) Supervisor de Manipulação

O Supervisor de Manipulação é o objeto funcional responsável pelo gerenciamento de toda a manipulação de conhecimento efetuada no AIATD.

É de sua responsabilidade, primeiramente, reconhecer todas as mensagens de solicitação de processamento de conhecimento, avaliando quem fez a solicitação e que tipo de processamento solicitou. Após isto avalia, para cada solicitação, os níveis de prioridade e as questões de segurança e integridade do conhecimento a ser processado. Só então é que as solicitações de processamento habilitadas são encaminhadas ao manipulador adequado, ativando-o. Por fim, a resposta de cada manipulação é recebida pelo Supervisor e endereçada a seu solicitante.

b) Manipulador de Frames

O Manipulador de Frames é o objeto funcional responsável pela realização de toda a manipulação de estruturas de frames e dos dados neles contidos.

Como manipulação de estrutura de frames, entende-se a criação, alteração, ou eliminação tanto do frame em sua totalidade como de suas partes, tais como classes, subclasses, slots, etc.

Como manipulação de dados, entende-se a escrita, leitura e busca de dados internamente aos frames.

Esta distinção na manipulação de estrutura de frames e dados tem por meta uma maior estruturação e eficiência da manipulação.

c) Manipulador de Regras

O Manipulador de Regras é o objeto funcional responsável pela realização de: montagem, avaliação sintática e inferência de todas as regras de produção do ambiente.

A montagem de regras corresponde, em uma primeira fase, ao reconhecimento de qual conjunto de regras deva ser utilizado ou criado. Posteriormente, procede-se à composição da regra propriamente dita, a partir do conjunto de informações provenientes da base de conhecimento, ou de um usuário. Note-se que a composição corresponde a colocar as regras em um formato adequado, para que possam ser inferidas. Este formato pode ser distinto daqueles em que a regra se encontra armazenada ou é introduzida no ambiente.

A avaliação sintática corresponde ao reconhecimento dos operadores usados

nas regras. Estes operadores tanto podem ser definidos pelo usuário como fazer parte de um conjunto armazenado internamente ao ambiente.

A inferência corresponde a uma avaliação semântica lógica, ordenada, das metas a serem provadas, sobre um conjunto de regras previamente determinado. Estas inferências sobre as regras podem ser realizadas através de um sequenciamento lógico padrão de avaliação de regras, ou através de sequenciamentos lógicos especiais, definidos pelo usuário.

d) Manipulador de Procedimentos

O Manipulador de Procedimentos é o objeto funcional responsável pela montagem e ativação de todos os procedimentos do ambiente, armazenados na base de procedimentos do ambiente.

A montagem do procedimento se faz necessária, uma vez que o conjunto de procedimentos a ser usado pelo ambiente pode não ser fixo, tendo-se a oportunidade de que o usuário configure os procedimentos que deseja usar.

A ativação de procedimentos corresponde à execução dos métodos computacionais a eles agregados.

Os procedimentos podem ser agrupados para operarem apenas sobre a estrutura de frames, sobre os dados contidos nos frames, sobre as regras, ou sobre conjunções variaveis destes tipos.

3.4-2.6 - Base de Conhecimento

A Base de Conhecimento é o objeto funcional responsável pela armazenagem de todo o conhecimento detido pelo AIATD. A armazenagem de conhecimento é executada sobre três bases distintas: Base de Dados, Base de Regras e Base de Procedimentos. Esta divisão é, fundamentalmente, consequência da tipificação adotada para o modelo abstrato de informação, descrito acima. Desta forma, garante-se uma maior estruturação e modularidade no tratamento do conhecimento e a possibilidade de operação paralela sobre diferentes tipos de dados. O modelo funcional da Base de Conhecimento é apresentado na figura 3.14.

a) Base de Frames

A Base de Frames contém informações sobre as estruturas de armazenagem de

dados e os dados relativos a cada entidade modelada. Entende-se por entidade modelada, aquela para a qual se definiu uma estrutura de dados que a descreva, e a esta estrutura se agregou um conjunto de informações. Contém ainda o conjunto de dados e identificadores de procedimentos relativos às condições de segurança e integridade da base.

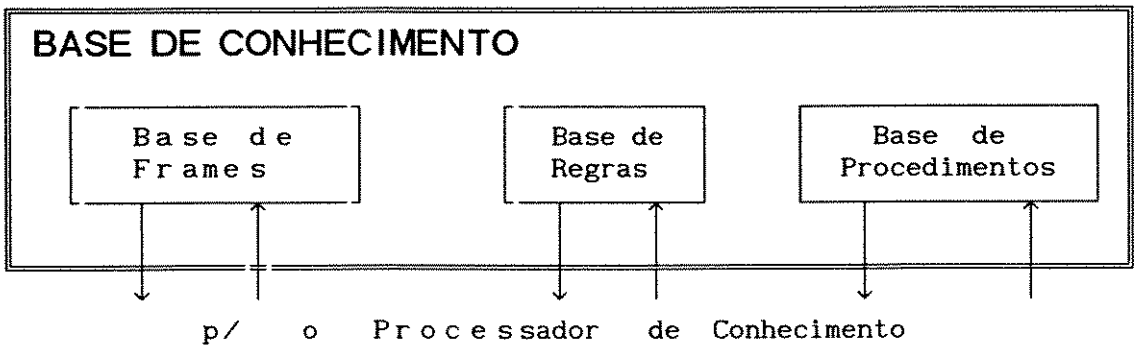


Fig. 3.14 - Modelo Funcional da Base de Conhecimento

b) Base de Regras

A Base de Regras armazena: a estrutura assumida para cada tipo de regra definida; os diferentes conjuntos de regras; os identificadores de dados e procedimentos associados a cada conjunto de regras; e os identificadores dos procedimentos e dados relativos às condições de segurança e integridade da base.

c) Base de Procedimentos

A Base de Procedimentos armazena: identificadores dos métodos básicos do ambiente (procedimentos básicos); identificadores dos métodos compostos do ambiente (procedimentos compostos); composição algorítmica dos métodos básicos; e identificadores dos procedimentos e dados relativos às condições de segurança e integridade da base.

Note-se que os procedimentos de segurança são ativados automaticamente sempre que qualquer manipulação de conhecimento seja solicitada. Encarregam-se da avaliação de níveis de prioridade de acesso, de condições de segurança, integridade e consistência do conhecimento a ser manipulado.

3.4-3 - Objetos Funcionais Terciários

Dentre os objetos funcionais secundários, acima descritos, apenas quatro deles são compostos por objetos funcionais terciários. São eles: Executor de Simulação (componente do Simulador), Analisador de Resultados (componente do Simulador), Aquisitor de Conhecimento Declarativo (componente do Aquisitor de Conhecimento), Manipulador de Regras (componente do Processador de Conhecimento).

3.4-3.1 - Executor de Simulação

A composição da Lógica de Simulação depende de fatores como: modelo de conhecimento adotado para a composição do modelo de simulação, estratégia adotada para a descrição de ordenação de ocorrências no modelo de simulação, estratégia adotada para a atualização do relógio de simulação [62].

Os objetos funcionais terciários componentes do Executor de Simulação são: Controlador de Simulação, Monitor de Entidades, Ordenadores Lógicos e Interface com Processador de Conhecimento. O modelo funcional do Executor de Simulação é apresentado na figura 3.15.

a) Controlador de Simulação

O controlador de simulação é o objeto funcional responsável por gerenciar a execução da simulação. Encarrega-se básicamente de três atividades: controlar o relógio de simulação, executar os eventos apresentados para ocorrência num dado instante e determinar o momento de encerrar a execução da simulação.

A execução dos eventos corresponde às seguintes fases:

- atualizar o relógio de simulação com o instante de ocorrência dos eventos.
- Inserir na lista de eventos ocorridos (histórico da simulação), aqueles escolhidos pelo Monitor de entidades para execução no instante dado.
- Disparar a execução de todos os métodos computacionais relacionados, com a execução dos eventos escolhidos para execução.
- Atualizar o estado do sistema sob simulação.

A determinação do instante de terminação da simulação é estabelecido a partir da avaliação da base de regras que contenha as condições de terminação. Esta avaliação ocorre a cada atualização do relógio de simulação.

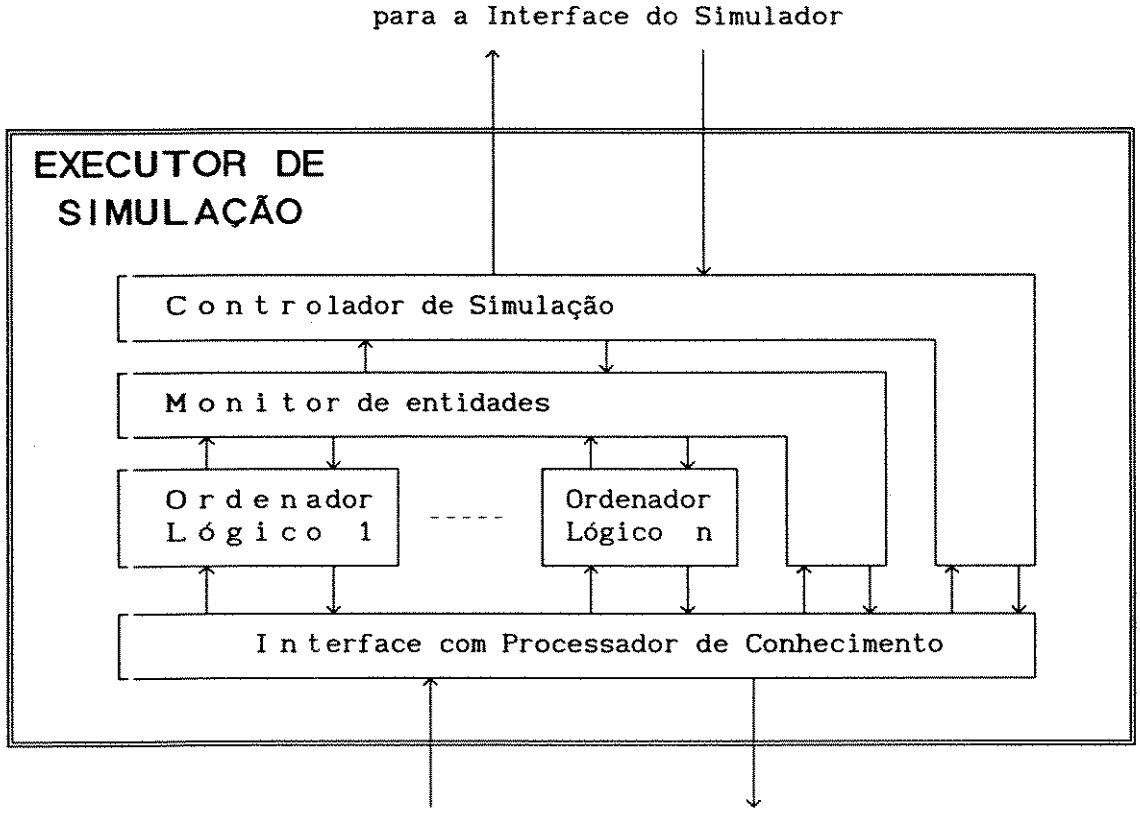


Fig. 3.15 - Modelo funcional do Executor de Simulação

b) Monitor de Entidades

O monitor de entidades executa um papel de árbitro. Esta arbitragem consiste em indicar ao Controlador de Simulação a execução dos eventos por ele avaliados como passíveis de ocorrência, naquele instante. Esta escolha é feita dentre aqueles eventos considerados como passíveis de ocorrência nas entidades participantes da corrente simulação.

A arbitragem é executada após a realização de inferências sobre o conjunto de regras que condicionam o interrelacionamento entre entidades participantes da simulação.

c) Ordenador Lógico

O Ordenador Lógico é o objeto funcional encarregado do tratamento de cada entidade participante da simulação, de forma independente. Sua função corres-

ponde à determinação de qual evento deva ser apresentado ao Monitor de Entidades para concorrer à execução, para cada entidade.

Determina-se, para cada entidade, o próximo evento passível de ocorrência, através de solicitação de inferência lógica ao Processador de Conhecimento. Esta inferência considera o modelo de simulação da entidade e seu estado atual.

No modelo de simulação da entidade infere-se sobre o sequenciamento de eventos e sobre as regras que condicionam a execução dos eventos, considerando o estado atual da entidade. Com base nessa inferência, escolhe-se apenas um evento para ocorrência, em cada entidade participante da simulação. Em todas estas inferências, não se considera qualquer informação das outras entidades ou o interrelacionamento entre elas, fatores que são avaliados pelo Monitor de Entidades.

O Ordenador Lógico atua da mesma forma para todas as entidades participantes da corrente simulação. Só após sua operação, sobre todas as entidades, é que é formada a lista dos eventos indicados para concorrer à execução. Esta lista é encaminhada ao Monitor de Entidades para consideração do interrelacionamento entre entidades.

d) Interface com Processador de Conhecimento

A Interface com o Processador de Conhecimento é o objeto funcional responsável por executar dois tipos de tarefas de formatação de conhecimento. A primeira delas consiste em formatar o conhecimento obtido a partir de consulta à Base de Conhecimento, para uso de cada um dos objetos que compõem o Executor de Simulação. O segundo tipo de tarefa, por ela realizado, consiste em formatar os resultados da simulação para que possam ser adequadamente endereçados à Base de Conhecimento.

3.4-3.2 - Analisador de Resultados

O Analisador de Resultados é o objeto funcional responsável por prover a devida análise do conjunto de resultados gerados pelo Simulador. É composto pelos seguintes objetos funcionais terciários: Formatador de Resultados, Processador Estatístico, Animador Gráfico e Interface com Processador de Conhecimento. O modelo funcional do Analisador de Resultados é apresentado na figura

3.16 abaixo.

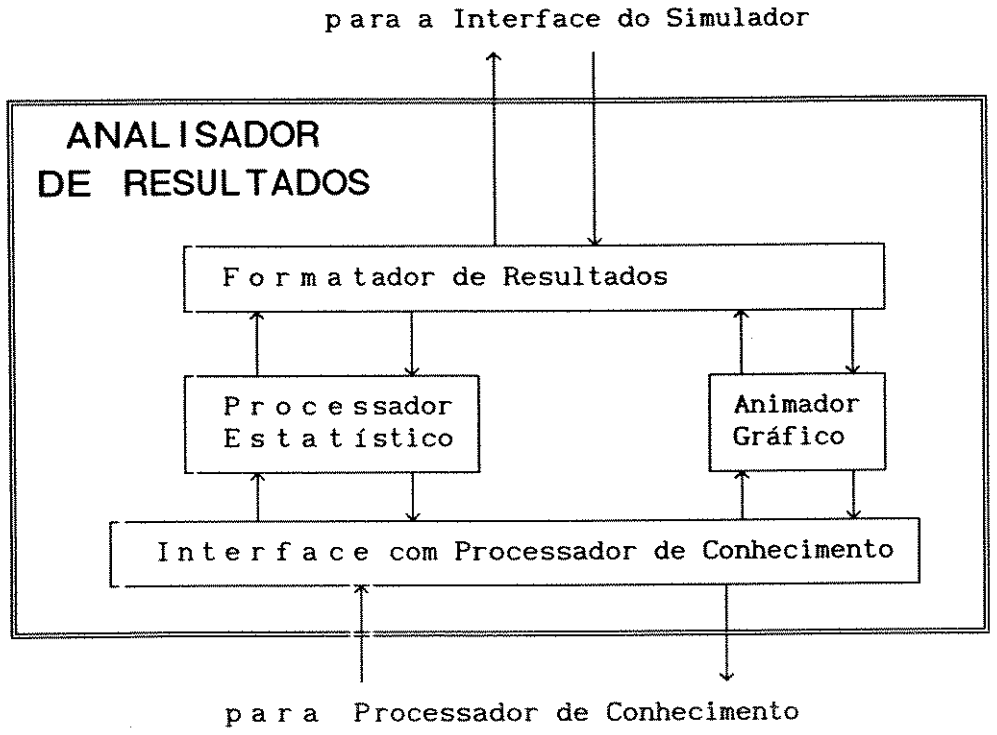


Fig. 3.16 - Modelo funcional do Analisador de Resultados

a) Formatador de Resultados

O Formatador de Resultados é o objeto funcional responsável por realizar a formatação dos resultados obtidos pelo Simulador.

A formatação dos resultados realiza-se em duas fases. A primeira desenvolve-se antes da simulação, quando, através de um diálogo com o usuário, são definidos os tipos e a estrutura dos resultados a serem emitidos como saída da simulação. A segunda desenvolve-se durante a simulação, ou após esta, quando o Formatador recebe os resultados gerados pelo Processador Estatístico, ou pelo Animador Gráfico. Neste caso, o Formatador coloca estes resultados na forma anteriormente especificada e os entrega à Interface do Simulador, para que sejam encaminhados ao usuário que os solicitou. É o responsável pelo estabelecimento de que tipo de análise é desejada pelo usuário. Encarrega-se, também, da formatação das saídas, conforme determinação do usuário, e de disparar os outros módulos componentes da análise de resultados, indicando-lhes o tipo de processamento desejado.

b) Processador Estatístico

O Processador Estatístico é o objeto funcional responsável por realizar toda a análise estatística solicitada pelo usuário. Através da Interface com o Processador de Conhecimento, solicita à Base de Conhecimento as informações necessárias para proceder às análises estatísticas, conforme determinado pelo usuário.

A avaliação estatística pode ser realizada de duas formas. Uma delas é feita diretamente sobre os dados da simulação, através de procedimentos definidos pelo usuário, ou através de procedimentos padrão, contidos em uma biblioteca de procedimentos. A outra corresponde à preparação dos dados para um pós-processamento mais elaborado, a ser desenvolvido por programas externos ao ambiente.

c) Animador Gráfico

O Animador Gráfico é o objeto funcional responsável pela apresentação de resultados na forma de animação gráfica. Pode operar de duas formas diferentes: animação em tempo de simulação e animação pós-processada.

A animação em tempo de execução realiza-se através do processamento de informações sobre a alteração do estado de entidades participantes da simulação. O envio de informações sobre estas alterações de estado para o Animador Gráfico é expressamente solicitada pelo usuário, ao compor o modelo de simulação.

A animação pós-processada realiza-se sobre as informações contidas no histórico (trace) da simulação. Devido à inexistência de requisitos de processamento em tempo real, a animação pós-processada pode ser mais elaborada graficamente.

d) Interface com Processador de Conhecimento.

A Interface com o Processador de Conhecimento é responsável por adequar a formatação das informações, provenientes da Base de Conhecimento, para que estas possam ser adequadamente usadas para análise estatística ou animação.

A adoção desta interface permite o desacoplamento do Processador de Conhecimento em relação aos métodos utilizados pelo usuário, para o tratamento

de resultados. Contribui, também, para uma maior modularidade da arquitetura do ambiente.

3.4-3.3 - Aquisitor de Conhecimento Declarativo

Dos componentes do Aquisitor de Conhecimento, apenas o Aquisitor de Conhecimento Declarativo é composto por objetos funcionais terciários.

O Aquisitor de Conhecimento Declarativo é o objeto funcional responsável por executar a aquisição de conhecimento segundo o modelo abstrato de informação acima descrito. É composto por três objetos funcionais terciários: Editor de Frames, Editor de Regras e Editor de Procedimentos. Seu modelo funcional é apresentado na figura 3.17.

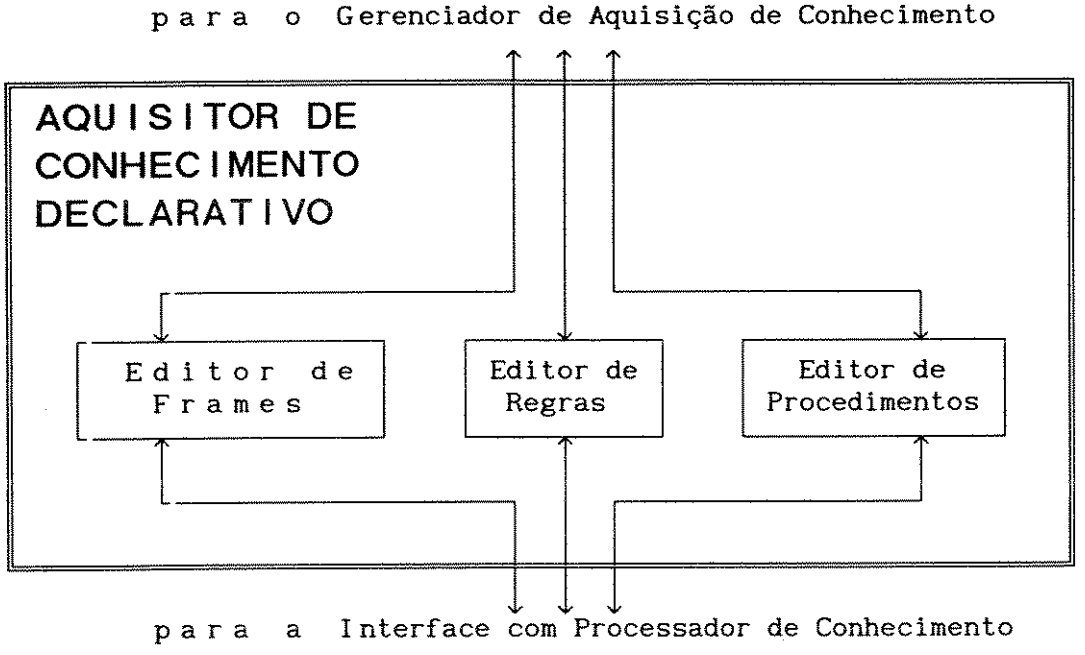


Fig. 3.17 - Modelo Funcional do Aquisitor de Conhecimento Declarativo

a) Editor de Frames

O Editor de Frames é o objeto funcional responsável pela edição dos frames a serem tratados no AIATD. Como edição de frame, entende-se tanto a composição da estrutura de cada frame como também a edição dos dados a serem armazenados nesses frames.

A edição dos frames é desenvolvida de modo amigável ao usuário. Um tutor de edição de frames orienta o desenvolvimento de estruturas de frames para diferentes aplicações. Como principal aplicação, tem-se a edição dos frames de entidades para simulação.

A modularização na composição do editor de frames permite que se altere o tutor de edição, auxiliando a edição de outros tipos de frame.

Durante a edição de frames, pode-se intercalar edição de dados com edição de estrutura do frame.

Ao final de cada edição, um verificador de consistência e integridade analisa a composição do frame e o endereça para armazenagem, com indicadores de consistência e integridade.

b) Editor de Regras

O Editor de Regras é o objeto funcional responsável pela edição das regras a serem usadas no AIATD.

Por edição de regras, entende-se a composição de operadores que sejam usados na edição das regras e a composição da regra.

A composição de operadores de regras corresponde à definição do operador, definindo-se o símbolo do operador, seu tipo de associatividade e sua prioridade de aplicação em relação aos outros operadores. Após a composição do operador, um analisador faz a avaliação de consistência do operador em função dos outros já constantes do sistema. Uma vez definidos, tais operadores podem ser utilizados na composição das regras.

Uma regra de produção genérica é composta pela associação de operadores a métodos computacionais previamente definidos, chamados procedimentos. A composição de regras pode ser realizada para diferentes tipos de regras de produção. Alguns tipos já se encontram definidos internamente, sendo acessados por um tutor de composição de regras.

Ao final de cada edição, um verificador de consistência e integridade analisa a composição das regras e as endereça para armazenagem, com indicadores de consistência e integridade.

c) Editor de Procedimentos

O Editor de Procedimentos é o objeto funcional responsável pela edição de

todos os procedimentos a serem usados no AIATD. Como procedimentos, entendem-se os métodos computacionais utilizados internamente ao AIATD, operando sobre informações. Procedimentos são utilizados para a verificação da habilitação de um usuário operar o ambiente, para a manipulação de conhecimento, para a alteração do estado de um sistema sob simulação, etc. Mais globalmente, os procedimentos são os responsáveis por toda e qualquer alteração do estado das informações internas ao AIATD.

O Editor de Procedimentos encarrega-se de duas principais funções: composição de novos procedimentos e composição de procedimentos referentes aos eventos de entidades.

Como composição de novos procedimentos, entende-se a definição e inserção no ambiente, de novos métodos computacionais. Estes novos procedimentos tanto podem ser gerados por completo como compostos a partir da combinação de procedimentos básicos arquivados em uma biblioteca de procedimentos.

Como composição das aplicações dos procedimentos, entende-se a indicação de quais informações, ou conjunto de variáveis serão afetadas pelo procedimento.

Ao final de cada edição, um verificador de consistência e integridade analisa tanto a composição dos procedimentos quanto a segurança das informações a serem motivo de alteração por parte do procedimento. Após tal verificação, os endereça para armazenagem, com indicadores de consistência, integridade e segurança.

3.4-3.4 - Manipulador de Regras

Dos componentes do Processador de Conhecimento, apenas o Manipulador de Regras é composto por objetos funcionais terciários.

O Manipulador de Regras é o objeto funcional responsável pela realização de: montagem, avaliação sintática e inferência de todas as regras de produção utilizadas no AIATD. É composto por três objetos funcionais terciários: Montador e Analisador Sintático de Regras, Máquina de Inferência Generalizada e Núcleo Dedicado de Inferência. Seu modelo funcional é apresentado na figura 3.18.

a) Máquina de Inferência Generalizada

A Máquina de Inferência Generalizada é o objeto responsável por realizar

inferências sobre as regras de produção, no AIATD.

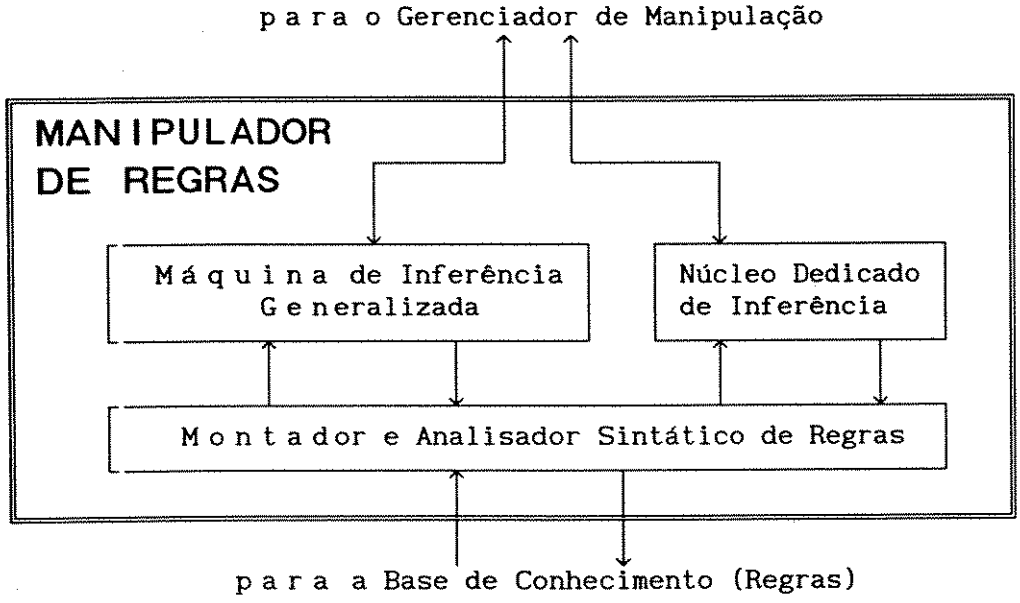


Fig. 3.18 - Modelo funcional do Manipulador de Regras

Trata-se de uma máquina de inferência operando sobre regras de produção com encadeamento reverso. Pode-se trabalhar com conjuntos cambiáveis de regras, sendo que cada conjunto pode ter seus próprios operadores. Durante o processo de inferência, esta máquina de inferência opera sobre uma memória de trabalho, considerando, além das regras, fatos (verdade incondicional), fatos temporários (resultados de inferências) e procedimentos (utilizados na composição das regras). A sequência de inferência obedece a uma ordem de prioridade, considerando primeiramente fatos, posteriormente fatos temporários, regras, procedimentos e, em casos especialmente definidos, informações solicitadas ao usuário.

Durante a operação, esta máquina de inferência pode ser acessada pelo usuário de três diferentes modos:

inclusão - inserindo novas regras e fatos;

ajuda - requerendo-se a apresentação de toda a sequência de inferência em curso;

consulta - requerendo inferência sobre um conjunto definido de regras para considerações intermediarias do usuário.

b) Núcleo Dedicado de Inferência

O Núcleo Dedicado de Inferência é o objeto funcional responsável pela realização de sequências especiais de inferência.

É definido e criado pelo usuário, nos casos em que a sequência lógica de inferência, oferecida pela Máquina Generalizada de Inferência, não seja adequada aos requisitos do usuário.

Comporta-se como um anexo do ambiente, permitindo personalização de inferência em casos especiais.

c) Montador e Analisador Sintático de Regras

O Montador e Analisador Sintático de Regras é o objeto funcional responsável pela montagem das regras, colocando-as em forma adequada a ser tratada por uma máquina de inferência.

Sua operação é requerida devido à flexibilidade do AIATD, que apresenta a possibilidade de tratar-se as regras de diferentes formas, construídas com operadores internos ou definidos pelo usuário.

A montagem consiste no recebimento das regras a partir da Base de Conhecimento e sua colocação em um formato que possa ser utilizado pela máquina de inferência de uso corrente. Para a Máquina de Inferência Generalizada, o formato utilizado é o de regras de produção apresentadas da seguinte forma: indicador, *se*, condição, *então*, ação. O indicador caracteriza o conjunto de regras em utilização e a regra dentro deste; *se* e *então* são operadores internamente definidos. Como "Condição", considera-se uma condição, ou um conjunto de condições, separadas por operadores de conjunção. "Ação" corresponde a uma única meta a ser provada pela inferência. Tanto as condições como a ação podem ser compostas por procedimentos, operando sobre tipos variáveis de informação, a serem especificados pelo usuário, durante a construção da regra.

A análise sintática da regra consiste em reconhecer todos os operadores e suas prioridades, procedimentos e tipos de informação, ordenando-os de forma a serem reconhecidos e tratados corretamente pela máquina de inferência. Realiza-se, por fim, uma verificação da consistência sintática dessa ordenação.

CAPÍTULO 4 - ASPECTOS DE IMPLEMENTAÇÃO

4.1 - INTRODUÇÃO

A implementação do Ambiente Integrado de Apoio à Tomada de Decisão (AIATD) seguiu o paradigma de objetos, através de um processo de desenvolvimento modular, com detalhamento sucessivo (top-down). Consideraram-se, como pontos básicos da implementação, o modelo funcional do AIATD e o modelo abstrato hierárquico de informação, discutidos anteriormente.

A linguagem computacional adotada para implementação tanto do ambiente, como um todo, como da estrutura de armazenagem do conhecimento e do próprio conhecimento foi o Prolog [28]. Isto, em virtude das suas bem conhecidas propriedades [36, 157, 159] e principalmente devido à possibilidade da combinação de programação declarativa e processamento simbólico com a simplicidade de representação e manipulação de conhecimento através da estrutura de "frames" [120].

Estabeleceu-se uma composição estrutural dos frames para armazenagem de conhecimento e dos procedimentos e regras, para tratamento de conhecimento, configurável pelo usuário, devido aos seguintes fatores:

a) o domínio das informações considerado apresenta-se muito amplo;

b) cada usuário de um ambiente, do tipo do proposto, tem diferentes interesses específicos, dentro do domínio das informações;

c) o ambiente, como um todo, tem como meta uma operação simplificada e amigável ao usuário, permitindo a este compor e manipular o conhecimento de uma forma personalizada.

Grande parte destas facilidades advém da flexibilidade imposta à operacionalidade dos elementos Aquisitor e Processador de Conhecimento (elementos funcionais do ambiente), através da linguagem Prolog. Desta forma, aumenta-se a flexibilidade na composição do conhecimento a respeito de entidades da manufatura, permitindo-se diferentes implementações para o modelo abstrato hierárquico de informação e para a utilização do conhecimento, configurado de acordo com este modelo adotado.

Apresenta-se, a seguir, uma implementação do modelo abstrato hierárquico de informação e do Ambiente Integrado de Apoio à Tomada de Decisão. Estas implementações são adequadas para o ambiente, cujas ferramentas básicas de

análise são um Simulador a eventos discretos e um Processador de Conhecimento.

Outras implementações, adequadas a diferentes ferramentas de análise e a diferentes domínios de informação podem ser desenvolvidas, devido à já citada flexibilidade operacional imposta aos elementos Aquisitor de Conhecimento e Processador de Conhecimento.

4.2 - IMPLEMENTAÇÃO DO MODELO DE INFORMAÇÃO

O modelo abstrato hierárquico de informação desenvolvido pode ser implementado tanto através de uma composição declarativa de conhecimento [96, 98] como através de uma composição formal, baseada na Lógica Temporal de Tempo Real Generalizada (GRTTL) [95, 100, 103, 163].

Primeiramente, apresenta-se a composição declarativa do modelo de informação, assumida como especificação básica para as ferramentas de análise do ambiente e adequada a entidades de manufatura. A seguir, apresenta-se a GRTTL, formalismo adotado para a especificação formal de entidades da manufatura. Apresenta-se, também, uma metodologia para a transformação de especificações formais expressas em GRTTL em especificações declarativas, de forma que tais especificações possam ser tratadas na forma padrão, utilizada pelas ferramentas de análise do ambiente.

4.2-1 - Implementação do Modelo Declarativo de Informação

O modelo declarativo de informação é usado para representar tanto entidades como sistemas de manufatura. Tal representação é usada com vista à execução de simulações a eventos discretos e de processamento de conhecimento por meio de inferências lógicas.

Todo o universo da manufatura pode ser considerado como composto por entidades a serem descritas pelo modelo declarativo de informação. Sistemas correspondem a integrações e interrelacionamento entre entidades.

4.2-1.1 - ENTIDADES

Do ponto de vista operacional do ambiente, e tendo como alvo a simulação a eventos discretos, uma entidade de manufatura pode ser classificada como ativa ou passiva [19, 62]. Entidades ativas são aquelas que exercem efeitos sobre as passivas. Em relação a simulações, a especificação de um sistema baseado no encadeamento de entidades ativas leva a uma caracterização de simulação orientada a equipamentos (facilities). A especificação de um sistema baseado no encadeamento de entidades passivas leva a uma simulação classificada como orientada a transações (transactions).

O modelo abstrato declarativo, desenvolvido para a especificação de entidades da manufatura, presta-se a ambos os tipos de entidades. Cada entidade é especificada através das três classes de informação do modelo abstrato, quais sejam:

Frame- conjunto de dados descritor da entidade;

Regras- conjunto de regras associadas à entidade e que condicionam certas ocorrências;

Procedimentos- conjunto de métodos computacionais associados aos eventos da entidade.

a) FRAME DE ENTIDADE

O frame de entidade é dividido em partes distintas, denominadas slots. Em cada slot, é apresentado um tipo de informação referente à entidade. Cada slot também pode apresentar-se dividido em sub-slots, apresentando diferentes instâncias do tipo de informação do slot.

Na condição de elemento aglutinador da especificação da entidade, o frame apresenta indicadores que o relacionam com os procedimentos e com as regras associadas à entidade. Assim, embora a composição do frame que especifica uma entidade seja configurável pelo usuário, a informação contida em cada slot tem um dentre os seguintes atributos:

valor (value): que representa a existência de um dado relacionado com o slot;

procedimento (if_needed): que representa a existência de um procedimento relacionado com o slot;

regra (rule): que representa a existência de um conjunto de regras relacionado com o slot.

Na especificação de uma entidade, alguns slots são considerados como de apresentação exigível. Os slots considerados como exigíveis são aqueles que indicam o nome da entidade, seu encadeamento hierárquico, suas ocorrências internas e suas entradas e saídas. Dependendo do interesse do responsável pela especificação de uma entidade, alguns destes slots podem não conter informações, com uma única exceção ao slot referente ao nome da entidade. Outros slots e sub-slots podem ser definidos pelo usuário no ato da especificação da entidade.

Um frame padrão para a especificação de uma entidade é apresentado na figura 4.1. Sua composição visa facilitar a execução de simulações e outros

aspectos operacionais das ferramentas de análise do ambiente, de forma que estas possam operar diretamente sobre informações contidas nos frames.

```
nome       : <valor> → nome da entidade
versão     : <valor> → número da versão corrente
data_hora  : <valor> → data e hora da versão corrente
tipo       : <valor> → nome da classe da entidade
membros    : <valor> → lista de entidades vinculadas
portas     :
             porta1 : <valor> → descrição da porta
                ⋮
             portan : <valor> → descrição da porta
bandeiras  :
             band1 : <valor> → descrição da bandeira
                ⋮
             bandn : <valor> → descrição da bandeira
informações : <valor> → informações adicionais
eventos    :
             evento1 : <valor> → descrição do evento
                ⋮
             eventon : <valor> → descrição do evento
rotinas    :
             rotina1 : <valor> → descrição da rotina 1
                ⋮
             rotinan : <valor> → descrição da rotina n
             evento_rotina : <valor> → evento e rotina
                                       para execução
```

Fig. 4.1 - Exemplo de frame de entidade

Os cinco primeiros slots do frame de entidade são considerados como um cabeçalho identificador, denominados slots identificadores. A especificação da entidade ainda é composta pelos slots descritores dos elementos de comunicação física e de controle, representados por portas e bandeiras; slot de informações gerais, configurável pelo usuário especificador; e pelos slots descritores de ocorrências, representados por eventos e rotinas.

a1) - IDENTIFICADORES

Os identificadores de uma entidade apresentados abaixo são: nome, versão, data_hora, tipo e membros.

NOME

O slot "nome", com atributo <valor>, indica o nome identificador da entidade. No ato da especificação, a unicidade deste nome é verificada como medida de segurança.

VERSÃO

O slot "versão", com atributo <valor>, identifica a versão corrente da especificação da entidade. O valor da versão é automaticamente atualizado a cada alteração da especificação. A critério do usuário especificador, versões anteriores podem ser arquivadas para efeito de segurança.

DATA_HORA

O slot "data_hora", com atributo <valor>, indica a data e hora de conclusão da versão corrente da especificação da entidade.

TIPO

O slot "tipo", com atributo <valor>, apresenta o nome do frame que descreva sua entidade-mãe, ou seja, a entidade de nível hierárquico imediatamente superior, à qual esteja vinculada. Assim, pode-se especificar classes de entidades, apenas alterando-se o conteúdo de atributos que não devam ser herdados pelas entidades de nível hierárquico inferior.

MEMBROS

O slot "membros", com atributo <valor>, indica a lista de entidades-filha, hierárquicamente relacionadas com esta e localizadas a um nível imediatamente inferior. Esta lista pode ser automaticamente corrigida toda vez que, durante a especificação de uma entidade, o nome desta seja citado no slot "tipo".

a2) - ELEMENTOS DE COMUNICAÇÃO FÍSICA E DE CONTROLE

As comunicações físicas e de controle, a serem realizadas com as entida-

des, são representadas através dos slots portas e bandeiras.

PORTAS

O slot portas apresenta a especificação das entradas e saídas de elementos de comunicação física de uma entidade.

Como elementos de comunicação física, entendem-se os corpos físicos, ou mensagens de comunicação, que chegam e partem da entidade descrita. As mensagens de comunicação, enviadas ou recebidas nas entradas e saídas da entidade, representam, indistintamente, tanto mensagens de comunicação física como mensagens de controle.

Este slot subdivide-se em vários sub-slots, cada um representando uma entrada ou saída da entidade.

Cada sub-slot recebe uma identificação do tipo "porta_x", que o individualiza dentro do slot "portas". Sua especificação ocorre através de um atributo <valor>, com uma composição de informações que descrevam a porta por ele representada. A composição de informações adotada apresenta tanto uma descrição da porta como também o acompanhamento de seu estado, quando a entidade estiver sob simulação.

A especificação básica adotada para a descrição de uma porta genérica "porta_x" é apresentada na figura 4.2. Nela são apresentadas informações que, além de a descrever como um elemento de comunicação, representam seu estado atual durante uma simulação e indicam como este pode variar. A descrição da porta utiliza-se de argumentos denominados : tipo, nº de lotes, tamanho do lote, valor atual, valor para decisão e instrumento, abaixo discutidos.

```
portas :
    ⋮
 porta_x : <valor> ⟶ [ tipo, nº de lotes, tamanho do lote,
                       valor atual, valor para decisão,
                       instrumento ]
    ⋮
```

Fig. 4.2 - Especificação básica de uma porta.

Porta_x - tipo

O argumento "tipo", primeiro na especificação da porta, indica o tipo de grandeza que por ela pode transitar. Optou-se por permitir apenas dois tipos de grandezas: valor numérico e fila de valores alfanuméricos. No caso de valor numérico, a porta comporta-se como um totalizador (buffer), não havendo possibilidade de distinção entre os elementos que transitam por ela. Como exemplos típicos de uma porta deste tipo, pode-se considerar um estoque de peças, indistintas, em espera para serem processadas por uma máquina ferramenta, ou um buffer de uma estação repetidora de uma rede de computadores, operando com mensagens do mesmo tamanho (indistintas). O valor alfanumérico a ser atribuído a esta porta pode ser processado, lido, ou modificado, de acordo com procedimentos a serem estabelecidos pelo responsável pela especificação da porta.

No caso em que o tipo de grandeza é uma fila de valores alfanuméricos, a porta comporta-se como um armazenador de grandezas distintas. Como exemplo típico de uma porta deste tipo, pode-se citar aquela que represente tanto o estoque de peças distintas a serem processadas por uma célula flexível de manufatura como um buffer de uma estação repetidora em uma rede de computadores que opere com mensagens distintas, com diferentes prioridades.

Porta_x - Número de lotes e Tamanho de lote

Os argumentos "número de lotes" e "tamanho do lote" são usados no caso em que pela porta transitam grandezas indistintas. Esse trânsito pode ser caracterizado por transferência de pacotes com um ou mais lotes. A composição entre o número de lotes, em cada transição, e o tamanho de cada lote compõe o total de cada transição na porta.

Porta_x - Valor atual

O argumento "valor atual" indica o conteúdo presente de uma porta. Dada a flexibilidade da representação permitida pela linguagem de implementação adotada, o mesmo argumento pode ser usado para ambos os tipos de grandeza a serem considerados em uma porta valor alfanumérico e fila de valores alfanuméricos. Em ambos os casos, este argumento corresponde a uma lista. Quando deva apresentar um valor alfanumérico, a lista é composta de apenas um elemento. No caso de representação de fila alfanumérica, a lista pode ser vazia ou composta

de um ou mais elementos.

As operações, a serem realizadas sobre o argumento "valor atual", são dependentes do tipo de grandeza por ele representado. Operações de leitura, comparação com um valor e escrita (alteração de valor) podem ser realizadas sobre este argumento, quando preenchido com um valor alfanumérico. Quando este argumento representar uma fila alfanumérica, podem ser realizadas operações de: leitura da lista; comparação de um valor com a cabeça, cauda ou algum elemento da lista; concatenação de um elemento como cauda ou cabeça da lista; eliminação da cauda ou da cabeça da lista. Antes da execução de qualquer operação, são realizados procedimentos de segurança que verificam o tipo de grandeza representado (1º argumento da porta), de forma a manter a consistência do argumento.

Porta_x - Valor para decisão

O argumento "valor para decisão" é utilizado como elemento de controle da porta. Assume a mesma tipificação de dados que o argumento "valor atual", conforme especificado pelo argumento "tipo". Sobre este argumento podem ser realizadas operações de leitura ou escrita, sendo que seu valor é utilizado como informação para operações em que se considere o argumento "valor atual". Um exemplo típico de utilização deste valor diz respeito ao estabelecimento de valores limite para a emissão de cartões de controle na metodologia "kan-ban".

Porta_x - Instrumento

O argumento "instrumento" é utilizado, principalmente, para o acompanhamento do valor atual da porta, quando de uma simulação da entidade em questão. O preenchimento deste argumento com um nome válido no ambiente, faz com que seja criado um frame de sistema com este nome, ao início de uma simulação. Ao longo da simulação, sempre que houver alteração do valor atual desta porta, o novo valor atual e o instante serão registrados no frame com o nome do instrumento, no slot referente à entidade, sub-slot referente à porta.

BANDEIRAS

O slot "bandeiras" apresenta a especificação de entradas e saídas de

sinais de controle em uma entidade. Os sinais de controle considerados são do tipo liga-desliga (on-off) e gerados internamente à entidade. Podem ser passados como mensagens para outras entidades, quando da realização de simulações. Este slot divide-se em sub-slots, cada um representando um sinal de controle.

Cada sub-slot "band_x" recebe uma identificação que o individualiza dentro do slot bandeiras. Sua caracterização ocorre através de um atributo <valor> e da informação que descreva seu estado atual.

A configuração básica assumida para a descrição de uma bandeira é através de um sub-slot do slot bandeiras, como indicado na figura 4.3.

O argumento estado atual de cada bandeira assume um dentre os valores "on" e "off".

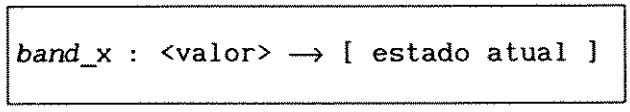


Fig. 4.3 - Especificação básica de uma bandeira.

a3) - INFORMAÇÕES

O slot "informações" apresenta qualquer conjunto de especificações de interesse do usuário especificador, assumindo um carater completamente configurável. Este slot pode ser subdividido em sub-slots definidos pelo usuário. Por sua vez, cada sub-slot pode também ser dividido em novos sub-slots, formando uma árvore de informações com encadeamento determinado pelo usuário especificador.

Cada sub-slot folha (sub-slot armazenador de informações) pode receber um dentre os atributos válidos no ambiente: <valor>, <regra>, <procedimento>.

A composição de informações a serem contidas no sub-slot folha é configurada pelo usuário. No caso de atributo <regra>, o usuário deve declarar o nome da base de regra a ser considerada.

No caso de atributo <procedimento>, o usuário deve declarar o nome do procedimento a ser considerado, explicitando-o posteriormente. A explicitação de um procedimento corresponde a escolher e encadear métodos computacionais previamente gerados, ou compor novos métodos computacionais. Em ambos os casos, deve-se explicitar sobre quais objetos o procedimento deva operar.

No caso em que o atributo do slot "informações" for <valor>, o usuário

deve especificar o número de argumentos que o compõem e seu identificador (mnemônico e posição relativa).

Para uma simplificação na definição dos métodos computacionais que operem sobre as informações contidas nestes sub-slots, estabeleceu-se que cada argumento seja composto por uma lista de valores alfanuméricos. Pode-se, assim, sem maiores definições, proceder-se a operações como leitura, comparação, escrita, concatenação de listas e eliminação de elementos, em cada argumento de um sub-slot com atributo <valor>.

a4) - OCORRÊNCIAS

As ocorrências internas à entidade são apresentadas através de descrições contidas nos slots "eventos" e "rotinas".

EVENTOS

O slot "eventos" apresenta a especificação de todos os eventos passíveis de ocorrerem internamente a uma entidade. Este slot subdivide-se em sub-slots, um para cada evento passível de ocorrência na entidade. Cada sub-slot "evento_x" apresenta, sob o atributo <valor>, informações relativas às consequências da ocorrência do evento e informações relativas à temporalidade imposta para a execução do próximo evento. Uma especificação básica de um evento é mostrada na figura 4.4.

Evento_x - nº do evento

O argumento "número do evento" estabelece uma relação unívoca entre o nome do evento e seu número indicativo. Este número serve para uma identificação mais suscinta do evento, em questões de sua identificação, durante simulações.

Evento_x - Nome do procedimento associado

O argumento "nome do procedimento associado" indica um nome para o procedimento associado com a execução deste evento, responsável pelas consequências

da execução do evento. Geralmente atribui-se o mesmo nome do evento a seu procedimento associado. Este procedimento corresponde à composição de um ou mais métodos computacionais básicos que operam sobre argumentos de alguns slots, como portas e bandeiras, alterando o estado da entidade. O procedimento associado ao evento é editado pelo usuário que escolhe ou cria os métodos computacionais básicos e indica sobre quais objetos (argumentos) estes devem operar.

evento_x : <valor> → [nº do evento, nome do procedimento associado, argumento opcional 1, argumento opcional 2, habilitação da base de regras, intervalo para próximo evento, tipo de distribuição, parâmetro 1 de distribuição, parâmetro 2 de distribuição, instante mínimo para próximo evento]

Figura 4.4 - Especificação básica de um evento.

Evento_x - Argumentos opcionais 1 e 2

Os argumentos "opcional 1" e "opcional2" são variáveis a serem opcionalmente instanciados pelo usuário especificador da entidade. Servem como informações auxiliares na identificação de um evento. Um exemplo típico de utilização destes argumentos é no caso em que o evento indica o início ou conclusão de uma operação de transporte. O argumento opcional 1 pode ser utilizado para indicação do ponto inicial do movimento, enquanto o argumento opcional 2 pode ser utilizado para indicação do ponto final do movimento.

Evento_x - Habilitação da base de regras

O argumento "habilitação para base de regras" estabelece a indicação da possível existência de regras que condicionem a ocorrência do evento. Este argumento pode assumir valores "hab" e "des".

O valor "hab" indica habilitação de consulta à base de regras, sendo

considerado em duas situações. A primeira, em tempo de modelagem da entidade, indica que deve ser editada uma ou mais regras que condicionem a ocorrência do evento. A segunda, em tempo de simulação, indica que deve haver uma consulta à base de regras da entidade. Esta consulta é para a verificação do cumprimento de condicionantes que restrinjam a ocorrência do evento, naquele instante. As regras consultadas são aquelas editadas em tempo de modelagem.

O valor "des" indica, em tempo de modelagem, que não há a necessidade de edição de regras que apresentem a execução do evento como meta. Neste caso, a execução do evento é considerada como fato, inserido de forma automática na base de regras.

A adoção deste argumento, na descrição de cada evento, traz duas vantagens, uma a tempo de modelagem e outra a tempo de simulação.

Em tempo de modelagem, a consulta a este argumento auxilia na verificação de consistência do modelo. Quando o valor do argumento se encontra em "hab" estabelece-se a necessidade de edição de regras que tenham a execução de tal evento como meta. Caso o usuário modelador venha a esquecer desta necessidade, o verificador de consistência da modelagem o alerta para tal.

Em tempo de simulação, a verificação deste argumento evita desnecessárias consultas à base de regras, para procurar condicionantes inexistentes referentes à ocorrência desse evento. Com isto pode-se agilizar o desenrolar da simulação.

Evento_x - Intervalo para próximo evento

O argumento "intervalo para próximo evento" estabelece uma temporalidade relativa a um evento, que restringe a ocorrência do próximo evento. A atribuição de um valor a este argumento, com unidade a ser estabelecida pelo usuário, condiciona temporalmente o instante, a partir do qual algum outro evento possa ocorrer. A inclusão desta temporalidade, na especificação de um evento, permite que, em uma especificação de ocorrências direcionada a eventos [62], possa-se expressar a temporalidade de forma compatível com uma especificação de ocorrências orientada a atividades.

Evento_x - Tipo de distribuição e Parâmetros 1 e 2

A atribuição da temporalidade, acima citada, pode ser realizada com cará-

ter tanto determinístico como probabilístico. No caso de uma atribuição determinística, este valor é considerado invariante. Em uma simulação, o instante, a partir do qual o próximo evento possa ocorrer, é determinado pela adição deste valor ao instante de ocorrência do evento em questão. O valor calculado é então atribuído ao último argumento do evento, denominado "instante do próximo evento".

No caso em que a atribuição do intervalo para próximo evento é probabilística, há que ser estimado um novo valor para esse intervalo, em função de um tipo de distribuição de valores e de um valor referência, indicado no argumento "intervalo para próximo evento". Os argumentos "tipo de distribuição", "parâmetro 1 de distribuição" e "parâmetro 2 de distribuição" permitem o estabelecimento de um valor probabilístico a ser atribuído ao intervalo estimado para o próximo evento.

Dependendo do tipo de distribuição atribuído, não existe necessidade de uso de todos os parâmetros de distribuição, sendo que aqueles não utilizados permanecem como variáveis a serem instanciadas.

Evento_x - Instante mínimo para próximo evento

O instante mínimo determinado para a ocorrência do próximo evento é calculado em tempo de simulação e atribuído ao argumento "instante mínimo para próximo evento".

Quando da ocorrência de um evento em uma simulação, a determinação do instante, a partir do qual o próximo evento possa ocorrer, é realizada da seguinte forma: a seu instante de ocorrência é acrescentado o valor do intervalo estimado para ocorrência do próximo evento. Este intervalo é calculado por geração de um número aleatório, levando-se em conta o dado atribuído para o intervalo do próximo evento, o tipo de distribuição, e os parâmetros de distribuição necessários.

ROTINAS

O slot "rotinas" estabelece a sequencialidade de ocorrência dos eventos da entidade. Este slot subdivide-se em sub-slots de dois tipos: rotina_x e evento_rotina. Podem existir vários sub-slots do tipo "rotina_x", cada um representando uma sequencialidade alternativa de ocorrências, denominada gene-

ricamente rotina_x. Um único sub-slot "evento_rotina" indica qual o evento, pertencente a qual rotina, foi determinado como próxima ocorrência da entidade.

Um exemplo de especificação básica do slot rotinas é apresentado na figura 4.5.

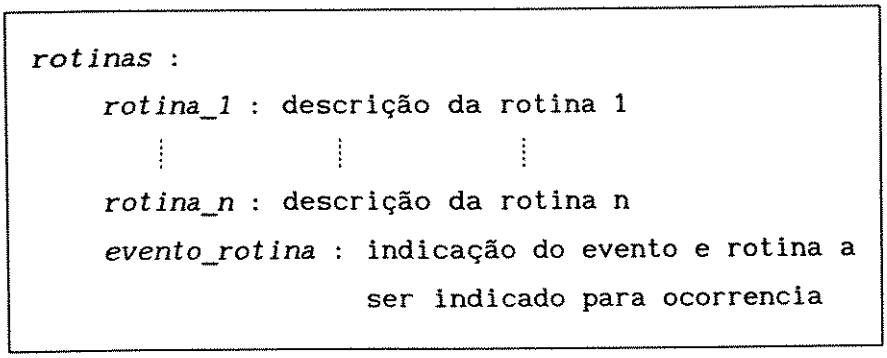


Fig. 4.5 - Composição básica das rotinas

Cada rotina expressa em um sub-slot "rotina_x" é definida de maneira a representar um sequenciamento particular de eventos, de forma cognitiva. São consideradas informações relativas à temporalidade das ocorrências, em conjunto com condicionantes referentes ao estado da própria entidade. Informações referentes a outras entidades que com ela estejam relacionadas, compondo um sistema sob análise, também podem ser consideradas. Isto é feito por meio da transmissão de mensagens através de portas e bandeiras.

A possibilidade de consideração de vários sequenciamentos, cada qual com suas particularidades, capacita o ambiente à descrição do comportamento, altamente flexível, apresentado por modernos sistemas de manufatura.

A sequencialidade de ocorrência dos eventos, expressa pela descrição da rotina, tanto pode apresentar-se através de uma evolução compulsória, como através de uma evolução ramificada.

Uma rotina, com evolução compulsória de eventos, corresponde ao caso em que não exista possibilidade de alterações em uma sequência de ocorrências previamente estabelecida para a entidade. Neste caso, o responsável pela especificação da entidade apenas indica qual evento deva ocorrer após outro. A temporalidade para a ocorrência de cada evento da sequência é estimada em função de informações a respeito do evento anteriormente ocorrido e de informações do estado atual tanto da própria entidade como de outras que com ela

estejam relacionadas.

Uma rotina com evolução ramificada corresponde a uma rotina flexivel, caso em que, após a ocorrência de um certo evento, vários outros possam ocorrer. A determinação de qual evento deva ser o próximo a ocorrer leva em conta informações a respeito do estado atual da própria entidade e de outras com ela relacionadas.

Um exemplo da sequencialidade compulsória é apresentado na figura 4.6. Aí, indica-se que, após o evento 1, deve ocorrer o 2 e, após este, o evento 3.

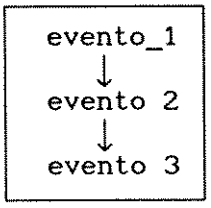


Fig. 4.6 - Exemplo de sequencialidade compulsória

Um exemplo de sequencialidade ramificada é apresentado na figura 4.7. Nela, indica-se que, após o evento 1, podem acontecer os eventos 2, 3, ou 4 e também que, após a ocorrência do evento 4, podem ocorrer os eventos 2, 3, ou 5.

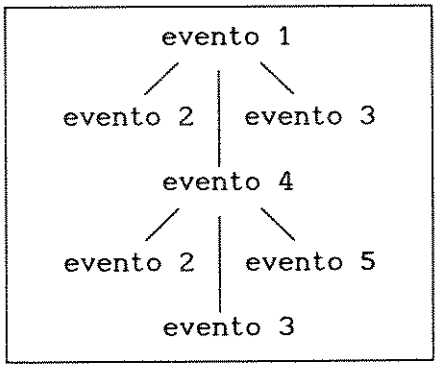


Fig. 4.7 - Exemplo de sequencialidade ramificada

Uma vez determinado qual deva ser o próximo evento a ocorrer, a temporalidade desta ocorrência também é determinada a partir de informações a respeito do evento ocorrido anteriormente e do estado tanto da própria entidade como de outras com ela relacionadas.

Todo este conjunto de informações é detalhado através de sub-slots que compõem a especificação de uma rotina qualquer. Estes sub-slots são chamados de sequência, decisão_fluxo e regras_entidade. A sequência apresenta uma descrição do sequenciamento dos eventos. A decisão de fluxo apresenta uma indicação da base de regras para decisão de fluxo na rotina. As regras de entidade indicam a base de regras da entidade na rotina. A descrição destes sub-slots é apresentada a seguir. Um exemplo de uma composição básica de uma rotina genérica X é apresentado na figura 4.8.

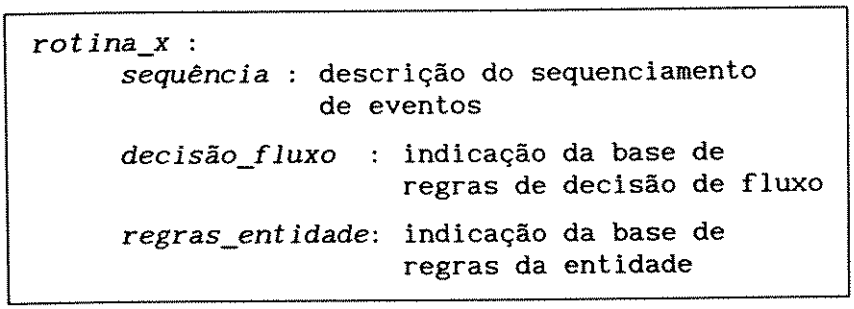


Fig. 4.8 - Composição básica da rotina genérica X

Rotina_x - Sequência

O sub-slot "sequência" expressa a sequencialidade dos eventos, de forma independente de outras condicionantes. Sob o atributo <valor>, apresenta-se, neste slot, uma lista com a ordenação dos eventos. Esta lista é composta por números que representam cada evento e por símbolos especiais, como mostra a figura 4.9.

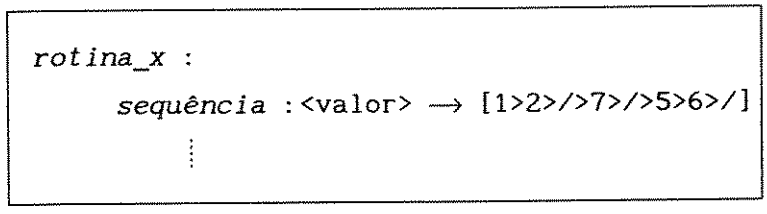


Fig. 4.9 - Especificação básica de uma sequência

O símbolo ">" representa um separador indicativo do sentido do sequenciamento. Quando este símbolo é seguido por um número representativo de um evento, estabelece-se um sequenciamento compulsório. Um exemplo desse caso é apre-

sentado pelas indicações "1 > 2" e "5 > 6" na figura 4.9. Nela, o sequenciamento expressa que, uma vez ocorrido o evento nº 1, o próximo evento a ocorrer é o de nº 2; e também que, uma vez ocorrido o evento nº 5, o próximo evento a ocorrer é o de nº 6. Em ambos os casos, essa sequência é observada incondicionalmente. A temporalidade para estas ocorrências, entretanto, depende de outros fatores, especificados na descrição dos eventos.

O símbolo "/" indica que a determinação do próximo evento a ocorrer depende da realização de uma inferência. Esta inferência é realizada sobre uma base de conhecimento, indicada pelo usuário especificador da entidade. Nesta base de conhecimento, estão contidos o conhecimento estático e métodos computacionais para a avaliação do conhecimento dinâmico.

No exemplo de sequência, apresentado na figura 4.9, as informações dadas por "2>/" e "7>/" indicam a tomada de uma decisão de fluxo, após a ocorrência dos eventos de nº 2 e 7, respectivamente. Cada tomada de decisão de fluxo, em relação ao desenvolvimento da rotina de ocorrências, é realizada através de inferências sobre uma base de conhecimento, indicada no sub-slot "decisão_fluxo", discutido a seguir. Como consequência desta inferência, pode-se chegar a qualquer um dos eventos da sequência como o próximo evento a ocorrer.

A ocorrência do símbolo "/" no início da representação de uma sequência provoca a realização de uma inferência para a determinação de qual deva ser o primeiro evento a ocorrer. A ocorrência deste símbolo, ao final da representação de uma sequência, indica a necessidade de realização de uma inferência para a avaliação de um possível termino do sequenciamento, ou de qual evento deva dar prosseguimento ao sequenciamento. Consegue-se, assim, uma simplificação e unificação na representação de sequências repetitivas e de uma só ocorrência.

Rotina_x - Decisão de fluxo

O sub-slot "decisão_fluxo" indica, sob atributo <regra>, o nome de uma base de conhecimento. Esta é a base consultada, quando da necessidade de realização de uma inferência para a determinação do próximo evento a ocorrer. A cada rotina especificada para a entidade corresponde uma base. Ela contém o conhecimento necessário à tomada de decisão, sobre qual o correto sequenciamento de fluxo a ser adotado.

O ponto de partida para a realização de cada inferência corresponde à informação de qual tenha sido o último evento ocorrido na entidade. A seguir,

são considerados outros fatores pertinentes, que são compostos pelo usuário especificador na base de conhecimento.

Um exemplo da especificação básica do sub-slot "decisão_fluxo" é apresentada na figura 4.10. Nela, indica-se que, para a tomada de decisão de fluxo na rotina X, deve ser consultada a base de conhecimento denominada decisão_fluxo_X.

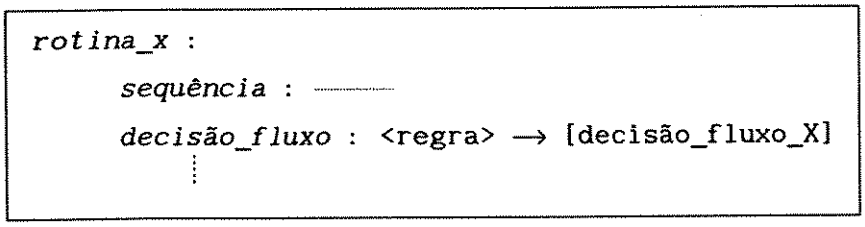


Fig. 4.10 - Especificação Básica da decisão de fluxo

Rotina_x - Regras de entidade

O sub-slot "regras_entidade", sob atributo <regra>, indica o nome de uma base de conhecimento. Esta é composta por informações sobre as condicionantes de ocorrência dos eventos da entidade, na determinada rotina. Nela, estão expressas, distintamente, as condicionantes para a ocorrência de cada evento. Um exemplo da especificação básica deste sub-slot é apresentada na figura 4.11.

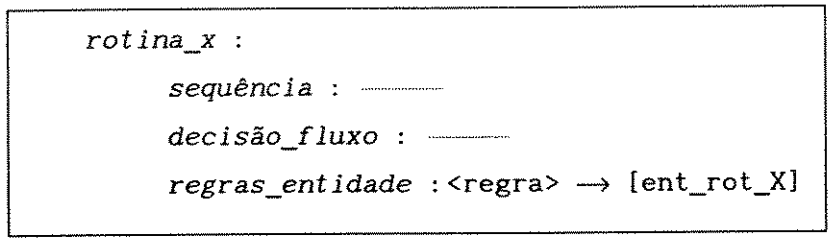


Fig. 4.11 - Especificação Básica das regras de entidade

As inferências realizadas sobre a base regras_entidade recebem, como ponto de partida, a indicação de qual deva ser o próximo evento a ocorrer. Para cada evento indicado, são realizadas inferências que avaliam as condições para sua ocorrência, levando em conta o estado presente da entidade.

Estas inferências são realizadas em tempo de simulação, avaliando, em um dado instante da simulação, se a entidade tem condições de apresentar algum

evento para ocorrência. O não cumprimento às condicionantes, estabelecidas para a ocorrência do evento escolhido, faz com que a entidade não apresente evento para ocorrência, naquele instante.

EVENTO_ROTINA

Concluindo a especificação de cada rotina, tem-se o sub-slot "evento_rotina". Este aparece uma única vez no slot rotina. Sob atributo <valor>, apresenta o evento escolhido como a próxima ocorrência da entidade, acrescido do nome da rotina a que pertence.

A escolha do evento a ser armazenado, neste sub-slot, pode ocorrer, em tempo de simulação, de duas maneiras. Uma, por simples verificação do próximo evento a ocorrer, quando de sequenciamento compulsório. Outra, por inferência sobre a base de conhecimento para decisão de fluxo indicada na rotina. Há que se salientar que a escolha do próximo evento a ocorrer não o habilita para imediata ocorrência, durante a simulação. Esta escolha apenas o indica para ser avaliado frente às condicionantes constantes da base de regras da entidade, indicada no sub-slot regras_entidade da rotina a que pertença o evento.

Um exemplo da especificação básica do sub-slot evento_rotina é apresentado na figura 4.12.

Neste exemplo, é indicado que o evento escolhido para ser a próxima ocorrência da entidade é o evento de nº Z, pertencente à rotina X.

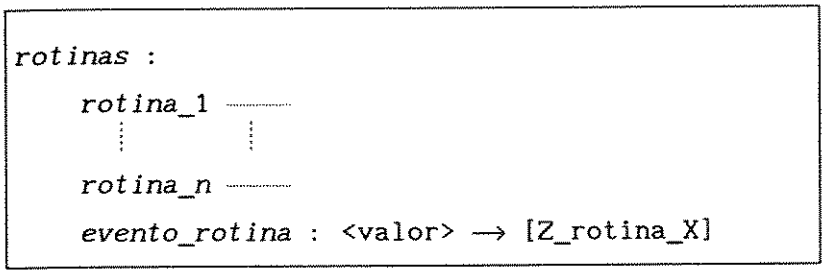


Fig. 4.12 - Especificação Básica do evento a ocorrer

b) - PROCEDIMENTOS

A cada evento da entidade, pode-se associar um procedimento. Este proce-

dimento é o responsável pela alteração do estado da entidade, devido à ocorrência do evento.

Um procedimento pode ser editado pelo usuário especificador da entidade, a partir de procedimentos básicos disponíveis em uma biblioteca de procedimentos. O usuário também pode editar e incluir novos procedimentos na biblioteca, deles fazendo uso quando necessário.

A biblioteca de procedimentos básicos é dividida em duas partes: biblioteca de procedimentos que operam sobre frames e biblioteca de procedimentos que operam sobre bases de regras.

Procedimentos que operam sobre frames

Os procedimentos básicos que operam sobre frames são aqueles que atuam sobre dados contidos em frames, tanto de entidades como de sistemas. Mais comumente estes procedimentos operam sobre o valor atual de portas ou sobre o estado atual de bandeiras.

Um exemplo de procedimento que opere sobre frames é o "altera_porta". Este procedimento, quando disparado, encarrega-se de alterar o valor atual de uma porta que aceite valores numéricos. Seus argumentos são: "entidade", "porta", "tipo", "ação".

Os argumentos "entidade" e "porta" especificam o frame e o slot sobre os quais o procedimento deve atuar.

O argumento "tipo" é usado para a confirmação do tipo de porta sobre a qual o procedimento deva operar. Com isto, evita-se que operações sobre valores numérico sejam efetuadas sobre portas que operem com filas de valores alfanuméricos e vice-versa. Promove-se, assim, uma verificação da consistência da operação indicada para o procedimento.

O argumento "ação" especifica qual a ação a ser efetuada sobre a porta indicada. Os possíveis valores para este argumento são: "[+]", "[-]", "[w,valor]".

No caso de atribuir-se "[+]" ao argumento "ação", o procedimento encarrega-se de incrementar o valor atual da porta especificada. O incremento a ser efetuado é calculado ao serem multiplicados os valores constantes dos atributos "nº de lotes" e "tamanho do lote", apresentados na especificação da porta em questão.

O decremento do valor da porta é realizado ao apresentar-se "[-]" como

valor do atributo ação.

A opção "[w,valor]" corresponde à ação de alterar-se o valor atual da porta, substituindo-o pelo valor indicado.

No exemplo da figura 4.13, apresenta-se uma típica especificação de atuação do procedimento "altera_porta". Nele, indica-se que o valor atual da porta "porta_5", da entidade "ent_1", deve ser alterado para "5000". Esta porta é do tipo que aceita valores numéricos.

```
altera_porta(ent_1,porta_5,num,[w,5000]).
```

Fig. 4.13 - Exemplo de procedimento sobre frames

Procedimentos que operam sobre bases de regras

Os procedimentos que operam sobre bases de regras são aqueles utilizados para auxiliar na realização das inferências expressas pelas regras. Encarregam-se da realização de avaliações sobre valores e estados atuais de entidades e sistemas. O resultado destas avaliações são, então, considerados em inferências.

Um exemplo de procedimento que opere sobre base de regras é o "compara_portas". Este procedimento encarrega-se de comparar o valor atual de duas diferentes portas. Seus argumentos são: "entidade 1", "porta 1", "tipo 1", "entidade 2", "porta 2", "tipo 2", "comparação".

Os argumentos "entidade 1", "porta 1", "entidade 2" e "porta 2" indicam quais os frames de entidade e qual a porta, em cada frame, deve ser consultada.

Os argumentos "tipo 1" e "tipo 2" são utilizados para uma verificação da consistência do tipo de valor da porta indicada.

O argumento "comparação" é utilizado para a indicação do tipo de comparação que é desejada. Pode assumir os seguintes valores: "igual", "maior", "menor", "maior_igual", ou "menor_igual".

Este procedimento realiza a comparação desejada entre as portas indicadas, gerando informação que é agregada à base de conhecimento. Esta informação pode ser de dois diferentes tipos: afirmação, ou negação, da comparação solicitada.

A afirmação agregada à base de conhecimento é do tipo "porta 1 eh igual

porta 2", enquanto a negação agregada é do tipo "porta 1 não_eh igual porta 2".

A informação agregada à base de conhecimento pode, então, ser avaliada como fato temporário, dentro do escopo de alguma regra.

No exemplo da figura 4.14, apresenta-se uma típica especificação do procedimento compara_portas. Nele, indica-se que se deve comparar o valor atual da porta "entrada_5" da entidade "ent_1" com a porta "saída_9" da entidade "ent_4", verificando se são iguais. Ambas as portas são do tipo numérico.

```
compara_portas(ent_1,entrada_5,ent_4,saída_9,igual)
```

Fig. 4.14 - Exemplo de procedimento sobre base de regras

Note-se que, uma vez concluída a inferência em que tal comparação foi solicitada, este conhecimento dinâmico que foi agregado à base de conhecimento é apagado. Isto é para que esta informação não continue sendo considerada e usada, como verdadeira, após o instante de sua validade.

c) - REGRAS

As regras compõem o terceiro tipo de informação do modelo abstrato de informação.

O tipo considerado de regras é o de regras de produção, ou seja, do tipo de construção: SE <condição> ENTÃO <ação>.

Para o simulador do ambiente, são consideradas apenas as inferências baseadas em encadeamento reverso (backward chaining). Neste caso, a parte <ação> é sempre formada por apenas uma meta, apresentada como meta de entrada da inferência.

A parte <condição> pode conter um ou mais objetivos a serem provados. No caso de mais de um objetivo, utiliza-se de um conector de conjunção, representado pelo operador "E" (and).

Para uma maior simplicidade do entendimento de um conjunto de regras, não se utiliza da disjunção em forma de operador. Esta é implementada através de duas regras com mesma meta (parte <ação>) e condições distintas. A precedência

da disjunção é imposta pela ordem de apresentação das regras na base considerada, mesma ordem assumida para a inferência.

Cada um dos objetivos da parte <condição> pode ser composto por um procedimento, ou por termos separados por operadores previamente definidos.

No caso dos procedimentos, para uma maior simplicidade em seu entendimento e facilidade de utilização, estes podem ser descritos de forma coloquial. Uma tabela, anexada à base de regras, indica a transformação necessária para converter a expressão coloquial em uma chamada ao procedimento, com seus devidos argumentos.

Os procedimentos geram conhecimento dinâmico que, quando anexado a uma base de conhecimento, corresponde a fatos temporários, passíveis de utilização para inferência.

Os operadores também podem ser definidos com apresentação coloquial, se assim devidamente definidos.

Todos os operadores básicos, previamente criados no ambiente, são definidos tanto para forma afirmativa como para a negativa. Isto permite uma considerável flexibilidade na construção de regras. No entanto, novos operadores podem ser definidos por usuários do ambiente, os quais, agregados ao conjunto de regras que os utilizam, permitem uma maior clareza, expressividade e simplicidade na composição de regras.

A apropriada reunião de termos, conectados por operadores, formam as condições, ou metas, que compõem as regras. Cada uma destas metas devem ser provadas frente ao conhecimento detido na base de conhecimento diponível.

O conhecimento disponível é formado por fatos (metas iquestionavelmente verdadeiras), regras e procedimentos.

Um exemplo de regra é apresentado na figura 4.15. Nesta regra, tem-se uma condição representada por procedimento, expresso entre aspas e outra formada por termos reunidos por operador.

A primeira condição é naturalmente satisfeita, ao executar-se o procedimento "'verifica_valor entrada_5-ent1'". Este procedimento avalia o valor atual da porta "entrada_5" da entidade "ent1".

A outra condição é representada por uma meta a ser provada, expressa pelos termos "valor_atual" e "5", conectados pelo operador "eh", representando a condição de que o valor atual avaliado pelo procedimento seja igual a 5.

Para o caso de também esta condição ser satisfeita, isto é, ser tida como verdadeira, após inferência sobre a base de conhecimento, prova-se a meta de

entrada. Esta prova é consequência da prova das metas parciais, que compõem a regra como um todo. Com isto, para a regra da figura 4.15, admite-se a execução do evento "evento_5" da entidade "ent1", operando em sua rotina "rot_2".

```
SE
  'verifica_valor entrada_5-ent1'
E
 valor_atual eh 5
ENTÃO [executar,evento_5,ent1,rot_2].
```

Fig. 4.15 - Exemplo básico de uma regra

No contexto de simulação do ambiente, quatro modelos de regras são utilizados: regras de decisão de fluxo, regras de entidade, regras de sistema e regras de condições de término.

Regras de decisão de fluxo são aquelas utilizadas para inferir-se qual o próximo evento a ser realizado, em uma sequência em que vários eventos possam ocorrer.

Regras de entidade são aquelas utilizadas para o estabelecimento de condições para a ocorrência de eventos da entidade.

Regras de condições de sistema expressam as condicionantes a serem cumpridas pelos eventos, levando-se em conta o estado atual do sistema sob simulação.

Regras de condições de término expressam as condicionantes a serem cumpridas para que se conclua uma simulação.

Em todos estes modelos de regras, a composição básica adotada é a mesma expressa no exemplo da figura 4.15.

4.2-1.2 - SISTEMAS

No contexto atual de especificação de entidades da manufatura, sistemas correspondem a um conjunto de entidades que se interrelacionam. Pelo paradigma de objetos adotado, cada entidade constitui-se em um objeto independente. Assim o interrelacionamento entre entidades, durante a simulação, é obtido por meio de troca de mensagens através dos elementos de comunicação e controle das entidades (portas e bandeiras).

A composição de um sistema implica nas seguintes fases: escolha das entidades a compô-lo, estabelecimento dos canais de comunicação entre as entidades, estabelecimento das condicionantes de interrelacionamento entre as entidades e estabelecimento das condições iniciais do sistema.

A escolha das entidades a comporem o sistema é feita dentre o elenco das entidades já especificadas. Nesta escolha, indica-se o nível de detalhamento hierárquico desejado para cada entidade componente do sistema.

O estabelecimento dos canais de comunicação, entre as entidades, é realizado através da indicação de quais portas e bandeiras de uma entidade se relacionam com as de outra entidade.

As condicionantes de interrelacionamento entre as entidades são expressas em uma base de conhecimento, sobre a qual se possa inferir durante uma simulação.

As condições iniciais do sistema a ser simulado são estabelecidas para todas as entidades que compõem o sistema. Isto se faz através da indicação do valor atual de cada porta e de cada bandeira destas entidades.

Todas estas informações, relativas à especificação do sistema, são expressas, conjuntamente, através de um "frame" denominado Status. Este "frame", além de todas estas informações, também contém algumas sobre o desenrolar da simulação.

Um exemplo da composição básica do "frame" de status é apresentada na figura 4.16.

A composição de cada slot é apresentada a seguir.

Nome

O slot "nome", sob atributo <valor>, indica que se trata do frame de status.

Versão

O slot "versão" apresenta, sob atributo <valor>, o número da presente versão de composição do sistema, com mesmo conjunto de entidades.

Data_hora

O slot "data_hora", sob atributo <valor>, indica a data e hora do término da presente versão de composição do sistema.

Tipo

O slot "tipo", sob atributo <valor>, apresenta o nome ambiente. Isto devido ao fato de este frame ser de utilização exclusiva do ambiente, para questões de simulação.

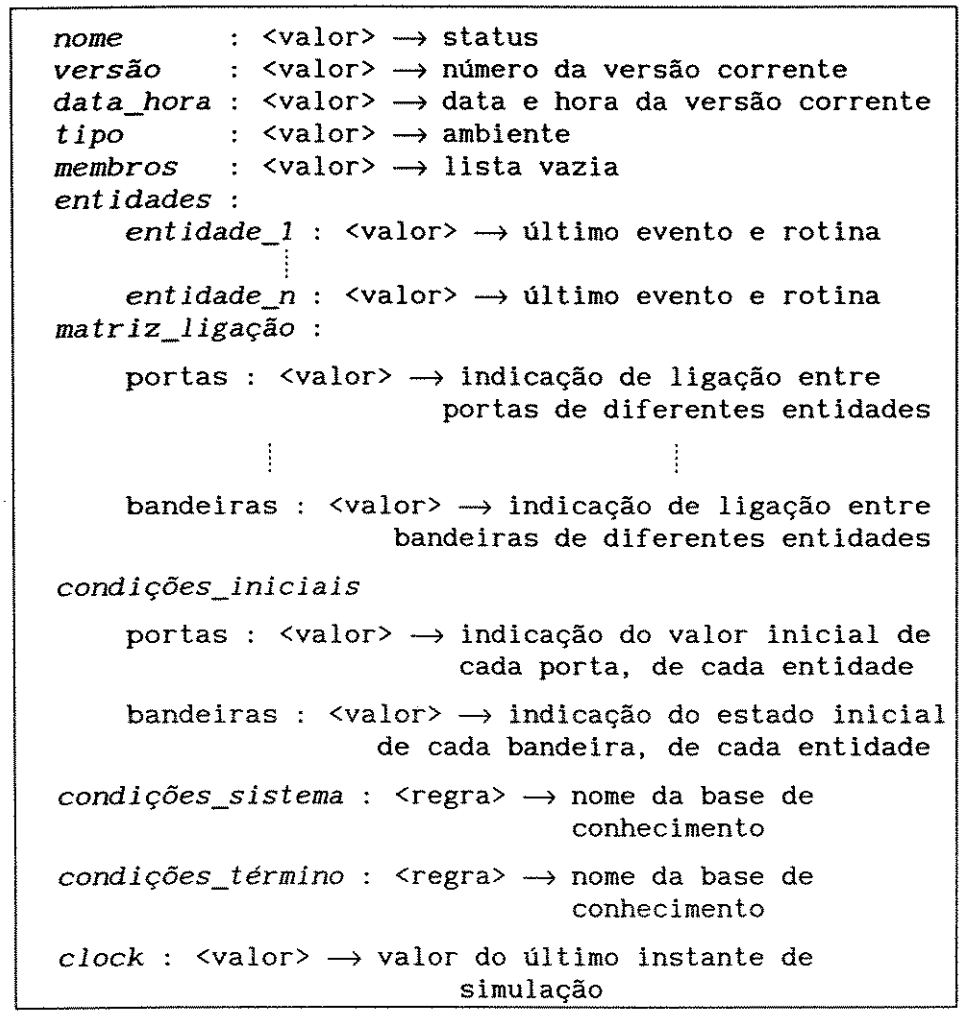


Fig. 4.16 - Composição básica do frame de status

Membros

O slot "membros" apresenta-se como uma lista vazia. Isto porque não se consideram subdivisões hierárquicas das informações de status do sistema.

Entidades

O slot "entidades" subdivide-se em diversos sub-slots. Cada sub-slot representa uma das entidades componentes do sistema sob especificação.

Cada entidade_x, descrita em um dos sub-slots, apresenta, sob atributo <valor>, uma lista composta de um único elemento. Este elemento é formado pelo número do último evento ocorrido na entidade, pelo separador "_" (underscore) e pela indicação da rotina a que este evento se referia.

Matriz de Ligação

O slot "matriz_ligação" apresenta o interrelacionamento entre portas e bandeiras de diferentes entidades. Subdivide-se em dois sub-slots denominados "portas" e "bandeiras".

O sub-slot "portas" apresenta, sob atributo <valor>, uma lista com elementos que expressam a conexão entre portas de diferentes entidades. Cada elemento da lista é composto por dois termos separados por um sinal de igualdade. Cada termo é composto pelo nome de uma porta e pelo nome da entidade a que pertence, separados pelo sinal "_" (underscore). Com isto, estabelece-se uma conexão lógica entre as portas assim relacionadas através do sinal de igualdade.

Uma vez estabelecida a conexão entre estas portas, a informação atribuída a uma delas é automaticamente atribuída à outra, durante uma simulação.

O sub-slot "bandeiras", também sob atributo <valor>, apresenta uma lista com elementos que expressam a conexão entre bandeiras de diferentes entidades. A composição de cada elemento desta lista segue a mesma regra adotada para a composição dos elementos da lista descrita para as portas.

Também para as bandeiras, durante uma simulação, o valor atribuído a uma é automaticamente atribuído à outra bandeira que com ela esteja relacionada.

Durante a edição de ambos estes sub-slots referentes ao slot "matriz_ligação", são realizados testes de consistência da informação. Neles, verifica-se a coerência do nome de cada porta, frente às entidades descritas no slot "entidades" e frente às especificações de cada entidade.

A figura 4.17 apresenta um exemplo de uma especificação básica do slot "matriz_ligação". Nela, estabelece-se que a porta X da entidade K encontra-se relacionada com a porta Y da entidade L. Também que a bandeira Z da entidade M está relacionada com a bandeira W da entidade N.

```
matriz_ligação :
        portas : <valor> ⟶ [portaX_entK = portaY_entL]
        bandeiras : <valor> ⟶ [bandZ_entM = bandW_entN]
```

Fig. 4.17 - Especificação básica da Matriz de Ligação

Condições Iniciais

O slot "condições_iniciais" apresenta o valor inicial de simulação, para cada uma das portas e bandeiras das entidades ativas na simulação, conforme indicadas no slot "entidades". Este slot subdivide-se em dois sub-slots denominados "portas" e "bandeiras".

O sub-slot "portas" apresenta, sob atributo <valor>, uma lista. Esta lista é composta por tantos elementos quantas forem as portas definidas no conjunto de entidades do sistema sob simulação. Cada elemento é composto por dois termos, separados por um sinal de igualdade. O primeiro termo é formado pelo nome da porta, seguido do nome da entidade, separados pelo símbolo "_" (underscore). O segundo termo recebe o valor inicial de simulação, a ser atribuído à porta.

O sub-slot "bandeiras" apresenta, sob atributo <valor>, uma lista. Esta lista é composta por tantos elementos quantas forem as bandeiras definidas no conjunto de entidades que formam o sistema sob simulação. Tal como para o sub-slot portas, cada elemento da lista é formado por dois termos, separados pelo sinal de igualdade. O primeiro termo é formado pelo nome da bandeira, seguido pelo nome da entidade, separados pelo símbolo "_". Ao segundo termo atribui-se o estado inicial da bandeira, seja este "set" ou "reset".

Durante a edição de ambos estes sub-slots, testes de consistência são realizados, para a avaliação do tipo de valor atribuído a cada porta ou bandeira.

Condições de Sistema

O slot "condições_sistema" apresenta, sob atributo <regra>, o nome de uma base de conhecimento. Esta base é composta de informações que expressam as

condicionantes globais do sistema sob simulação. Por condicionantes globais, entende-se o conjunto de condições que vinculem a ocorrência de certos eventos em uma entidade, devido ao estado de outras entidades.

Esta base é consultada, em tempo de simulação, para a avaliação da possibilidade de ocorrência do conjunto de eventos, para tal indicado, em um instante da simulação.

Condições de Término

O slot "condições_término" apresenta, sob atributo <regra>, o nome de uma base de conhecimento. Esta base é composta de informações que expressam os requisitos a serem cumpridos antes da conclusão de uma certa simulação. Ela é consultada, em tempo de simulação, após cada alteração do estado presente do sistema.

Clock

O slot "clock" apresenta, sob atributo <valor>, o tempo de simulação decorrido. Este valor é atualizado, de forma discreta, quando da ocorrência de um evento (ou vários), em um dado instante de simulação. A atualização se processa ao atribuir a este slot o valor do instante de ocorrência do evento.

4.2-2 - Implementação do Modelo Formal de Informação

O modelo utilizado para a especificação formal de entidades é o proposto na Lógica Temporal de Tempo Real Generalizada (GRTTL) [163].

No contexto deste trabalho, a GRTTL é usada para a especificação formal de entidades da manufatura, enquanto separadas em planta e controlador.

A lógica temporal de Tempo Real Generalizada (GRTTL) é um formalismo criado a partir da lógica temporal, à qual são acrescentados: operadores especiais para tratar o não-determinismo dos sistemas, axiomas, regras e teoremas para tratar limitantes de tempo real. A estrutura de descrição dos Sistemas a Eventos Discretos (DEDS), aqui adotada, corresponde aos Modelos Estocásticos Limitados de Tempo Real (RT-BSM), vistos a seguir, juntamente com a sintaxe e

semântica da GRTTL.

4.2-2.1 - RT-BSM

Os sistemas dinâmicos a eventos discretos que têm como características o não-determinismo (com probabilidade conhecida) e limitantes de tempo real, podem ser modelados, para fins de representação na GRTTL, como Modelos Estocásticos Limitados de Tempo Real, ou RT-BSM (Real-time Bounded Stochastic Models). O sistema é denotado por Σ. Associado a ele, existe um conjunto de eventos $\underline{E}$,que ocorrem em certos instantes de tempo e um conjunto de fórmulas lógicas $\underline{F}$, que fornecem as condições para a ocorrência destes eventos ao longo do tempo.

Um estado do sistema pode ser visto como um subconjnto de F e um evento como aquele que leva de um estado para outro. Sendo assim, pode-se dizer que um evento altera o conjunto de fórmulas que dirigem o comportamento do sistema naquele instante.

Cada sequência possível de eventos define um RT-BSM. Deste modo, têm-se vários RT-BSM's relativos a um só Sistema a eventos discretos (Σ).

Um RT-BSM pode ser descrito formalmente como a sétupla (S, E, s_0, A, P, l, u), onde :

S - É um conjunto enumerável e não vazio de estados;

E - É um conjunto de eventos possíveis, sendo cada evento $e_{ij} \in E$ uma transição de s_i a s_j com $s_i, s_j \in S$ (deste modo E é também enumerável);

s_0 - $s_0 \in S$ é o estado inicial.

A - $A: S \longrightarrow F^*$ (F^* denota o conjunto de todos os conjuntos de F) é uma função que associa cada estado de S a um conjunto de fórmulas em F^*.

P - $P: E \longrightarrow [0,1]$ associa a cada evento possível e_{ij} uma probabilidade $p_{ij} \triangleq p(e_{ij})$.

l - $l: SxS \longrightarrow \mathbb{R}^+$ e $u: SxS \longrightarrow \hat{\mathbb{R}}$ são, respectivamente, os limites inferiores e superiores de tempo. Dados dois estados, s_i e s_j, l fornece o tempo mínimo

para que o estado do sistema passe de s_i para s_j. Semelhantemente, u fornece o tempo máximo para a passagem de s_i para s_j.

Na definição acima é interessante notar que se l=0 e u=∞ para todos os conjuntos de estados, recai-se no sistema descrito em [171], em que não há limitantes de tempo real para a ocorrência de eventos. Se J é um conjunto unitário, ou seja, {j}, tem-se p_{ij}=1 e recai-se no caso determinístico.

4.2-2.2 - SINTAXE E SEMÂNTICA DA GRTTL

Os símbolos da linguagem incluem símbolos constantes individuais, chamados símbolos constantes globais; símbolos variáveis individuais, chamados símbolos variáveis locais; letras de funções; letras de predicados; conectivos lógicos: ¬ (símbolo de negação), ∧ (conectivo de conjunção) e ∨ (conectivo de disjunção); operadores temporais: ⊙ (operador <u>próximo</u>), □ (operador <u>daqui-por-diante</u>), ◊ (operador eventualmente) e $\mathcal{U}$(operador <u>até-que</u>); e o operador modal ∇ (operador <u>certeza</u>), que denota probabilidade 1 [90].

É importante notar que a idéia de não-determinismo aqui analisada parte da impossibilidade de se observar, previamente, para qual estado o sistema evoluiu. Dado isto, após a ocorrência de um determinado evento, a única informação disponível é que o sistema saiu de um estado e foi para outro, pertencente a um dado conjunto, sem se saber, contudo, qual é o estado-destino.

A GRTTL é uma linguagem multi-conjunto, semelhante à lógica temporal simples [171].

<u>AXIOMAS</u>

O conjunto de axiomas da GRTTL é dado como a seguir:

Se uma fbf w tem a forma de um dos esquemas seguintes, então w e □w são axiomas:

A1. w, onde w é uma instância de uma tautologia proposicional

A2. $\Box(w1 \Rightarrow w2) \Rightarrow (\Box w1 \Rightarrow \Box w2)$

A3. $\Box w \Rightarrow w$

A4. $\odot\neg w \Leftrightarrow \neg\odot w$

A5. $\odot(w1 \Rightarrow w2) \Rightarrow (\odot w1 \Rightarrow \odot w2)$

A6. $\Box w \Rightarrow \odot w$

A7. $\Box w \Rightarrow \odot \Box w$

A8. $\Box(w \Rightarrow \odot w) \Rightarrow (w \Rightarrow \Box w)$

A9. $w_1 \mathcal{U} w_2 \Leftrightarrow [w_2 \vee (w_2 \wedge \odot(w_1 \mathcal{U} w_2))]$

A10. $w_1 \mathcal{U} w_2 \Rightarrow \Diamond w_2$

Somados a estes axiomas, precisa-se de axiomas que permitam igualar símbolos; um sinal de igualdade.

Tem-se, assim:

A11. $t=t$, para qualquer termo t.

A12. $t_1=t_2 \Rightarrow (w(t_1,t_1) \Rightarrow w(t_1,t_2))$, para quaisquer termos t_1 e t_2 e qualquer fbf $w(t_1,t_1)$e $w(t_1,t_2)$, onde $w(t_1,t_2)$ é obtido de $w(t_1,t_1)$, trocando por t_2 algumas das ocorrências de t_1 que não estejam dentro do escopo de nenhum operador temporal.

A13. $\odot P(t_1,t_2,\ldots,t_n) \Leftrightarrow P(\odot t_1, \odot t_2, \ldots, \odot t_n)$

A14. $\odot f(t_1,t_2,\ldots,t_n) \Leftrightarrow f(\odot t_1, \odot t_2, \ldots, \odot t_n)$

A15. $c = (\odot c)$

Uma abordagem detalhada da descrição da regra de inferência e das regras derivadas são apresentadas em [171]. A única regra de inferência em todo o sistema é o Modus Ponens. A partir dela, têm-se várias regras derivadas, como a Regra da Argumentação Proposicional e a Regra da Cadeia.

Pode-se querer provar só resultados simples. Por isso, definem-se, como axiomas, todas aquelas fórmulas que são verdade sobre uma avaliação desejada. Ou seja:

A16. Se w é uma fórmula obviamente verdade sobre uma avaliação desejada, então w e $\Box w$ são axiomas.

Chamam-se, a estes axiomas, de axiomas de domínio.

4.2-2.3 - DESCRIÇÃO DE EVENTOS

Tratando de sistemas dinâmicos, é importante a descrição de eventos. No presente caso, coloca-se uma ordem no sistema para representar os eventos: os

símbolos constantes globais desta ordem são chamados de "símbolos de eventos" e inclui-se o símbolo δ entre as variáveis locais desta ordem. Intuitivamente os símbolos de eventos representam as várias transições que o sistema sob consideração pode experimentar, sendo que δ representa, para qualquer instante, o próximo evento que venha a ocorrer. Sendo assim, a fórmula $\delta=\alpha$ significa que o evento representado pelo símbolo de evento α ocorre (agora).

A sequência de uma estrutura é infinita, mas pode ocorrer que um sistema pare de transicionar para sempre, talvez porque uma meta tenha sido alcançada e não sejam necessárias mais ações, ou porque ocorreu um "deadlock" e não é possível mais nenhuma transição. Com o propósito de descrever tais situações, inclui-se o símbolo ϕ entre os símbolos de eventos. ϕ serve para identificar o evento nulo. Este evento deixa inalterados os valores de todas as variáveis locais, incluindo δ. Deste modo, se o evento nulo ocorre de um certo ponto em diante, todas as variáveis locais manterão o mesmo valor a partir daí.

O próximo axioma do sistema de prova fixa a interpretação do símbolo ϕ. Se uma fórmula tem a forma abaixo descrita, então ela é um axioma do sistema.

A17. $\Box[\delta=\phi\Rightarrow(\odot x)=x]$

4.2-2.4 - LÓGICA TEMPORAL NUM CONTEXTO DE CONTROLE

Nas aplicações a serem consideradas, as fórmulas da lógica temporal são usadas para descrever formalmente a dinâmica e o comportamento de sistemas. Para mostrar que um sistema possui certa propriedade, é necessário que esta propriedade seja satisfeita pela descrição formal da dinâmica do sistema, em lógica temporal. Esta prova se dá sobre a trajetória do sistema, ao longo do tempo (estrutura).

Numa abordagem voltada para controle, uma fórmula escrita como v, não $\Box$v, representa uma trajetória que nem sempre é satisfeita para todo t. Uma fórmula que não contenha operadores temporais refere-se somente ao estado inicial da estrutura para, descrever as condições iniciais do estado inicial do sistema.

Os próximos axiomas estarão relacionados com a noção de certeza.

A18) $\nabla(w1 \Rightarrow w2) \Rightarrow (\nabla w1 \Rightarrow \nabla w2)$;

A19) $\Delta\nabla w \Leftrightarrow \nabla w$;

A20) $\nabla w \Rightarrow w$;

Apresentam-se, a seguir, axiomas que descrevem a interrelação entre tempo e imprevisibilidade (chance), como encontrada em [90].

A21) $\nabla \odot w \Rightarrow \odot \nabla w$

A21) diz que a passagem do tempo só pode reduzir a faixa do que é possível.

A22) $\Box \Diamond \Delta[\bigwedge_{i=0}^{k} \odot^{(i)} \nabla w_i] \Rightarrow \Diamond[\bigwedge_{i=0}^{k} \odot^{(i)} \nabla w_i]$

que é o mesmo que

$$\Box \Diamond (\nabla w_0 \wedge \Delta\odot(\nabla w_2 \wedge \Delta\odot(\ldots \wedge \Delta\odot\nabla w_k)\ldots))) \Rightarrow$$

$$\Diamond(w_0 \wedge \odot w_1 \wedge \odot\odot w_2 \wedge \ldots \wedge \odot^{(k)} w_k)$$

expressa o fato de que acontecimentos sucessivos aleatórios são independentes.

4.2-2.5 - REPRESENTAÇÃO DOS LIMITES DE TEMPO REAL

Outro conjunto da linguagem é usada para representação do tempo associado às transições (equivalente conceitualmente a tempo de permanência num estado). Os símbolos constantes globais são os números reais. O símbolo t é incluído entre as variáveis locais deste conjunto. Sendo assim, a fórmula t=T, em que T é uma meta-variável, significa que o tempo, no contexto desta fórmula, vale T unidades. Já nesta estrutura, faz sentido falar no operador "next", ou seja, os eventos acontecem em determinados momentos, e também definir algumas convenções para o limitantes inferior e superior de tempo. Sendo assim tem-se:

l_A=a representa $\Box[(\delta=\alpha \wedge t=T) \Rightarrow \Diamond[\delta=\beta \wedge t \geq T+a]$

u_A=b representa $\Box[(\delta=\alpha \wedge t=T) \Rightarrow \Diamond[\delta=\beta \wedge t \leq T+b]$

onde $\geq$ e $\leq$ são predicados binários que representam as noções de "maior ou igual que" e "menor ou igual que", respectivamente. $\alpha \in E$ é um evento que leva de um estado O (O $\in$ S) para A (A $\in$ S) e $\beta \in E$ é um evento que leva de um estado A para B (B $\in$ S), como na figura 4.18.

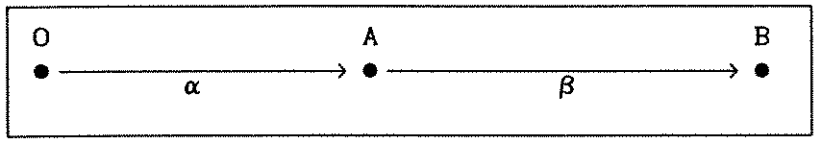


Fig. 4.18 - Exemplo de transição de estados

Intuitivamente falando, os limites l_A e u_A representam, respectivamente, o tempo mínimo e máximo de permanência num estado.

Para a prova dos teoremas que possuem considerações de tempo real, necessita-se de regras que possibilitem o manuseio e a argumentação formal destas expressões. Além das regras e dos teoremas descritos anteriormente, apresenta-se uma regra que expressa a soma de limites de tempo superiores.

Sejam w1, w2 e w3 quaisquer fórmulas temporais (no presente caso são principalmente fórmulas do tipo $\delta=\alpha$, em que α é um evento) e sejam d1 e d2 constantes quaisquer do conjunto de tempo e T qualquer valor do mesmo conjunto. Então:

$$\frac{\Box[(\forall T:\ (w1 \wedge t=T) \longrightarrow \Diamond(w2 \wedge t = T + d1))] \quad \Box[(\forall T:\ (w2 \wedge t=T) \longrightarrow \Diamond(w3 \wedge t = T + d2))]}{\Box[(\forall T:\ (w1 \wedge t=T) \longrightarrow \Diamond(w3 \wedge t = T + d1 + d2))]}$$

Esta regra garante que se o intervalo máximo de tempo, compreendido entre um estado que satisfaça w1 para um satisfazendo w2, é d1 e se o intervalo máximo de tempo, compreendido entre um estado que satisfaça w2 para um satisfazendo w3, é d2, então o intervalo máximo de tempo, compreendido entre um estado que satisfaça w1 para um satisfazendo w3, é d = d1 + d2.

Uma outra regra incluída no sistema de prova é a "regra da cadeia temporalizada" que é uma versão muito parecida com a apresentada em [129].

A sua apresentação é feita por:

$$\Box[v_i \wedge w_i \Rightarrow \odot_j \underline{\vee}_0^{i-1} v_j]$$

$$\Box[v_i \Rightarrow \Diamond w_i]$$

$$\Box[v_i \Rightarrow (\odot(_j \underline{\vee}_0^{i-1} v_j) \vee \odot v_i)]$$

$$\forall\, i=1,\ldots,k;\ \forall\, T\ [v_i \wedge w_i \wedge t=T \Rightarrow \Diamond(_j \underline{\vee}_0^{i-1} (v_j \wedge w_j \wedge (t \leq T+d_j)))]$$

$$\Box[(_i \underline{\vee}_0^{n} v_i) \Rightarrow ((_i \underline{\vee}_1^{n} v_i)\ \mathcal{U}\ (v_0)) \wedge (t \leq T+d)]$$

Onde $d = d_1 + d_2 + \ldots + d_n$

A regra acima diz que se o sistema está em v_n, ele chega no estado v_0 dentro de, no máximo, d instantes de tempo. Esta regra é similar à regra da cadeia, adicionando-se as considerações de tempo real.

4.2-3 - Transformação do Modelo Formal em Modelo Declarativo

É importante observar que existem requisitos para que uma especificação em lógica temporal genérica seja compatibilizada com uma especificação declarativa genérica. A transformação dos dois tipos de especificação, aqui considerados, cumprem tais requisitos. Um mapeamento da transformação das características de uma especificação em outra e uma metodologia para a transformação destas especificações são apresentadas a seguir.

4.2-3.1 - REQUISITOS PARA TRANSFORMAÇÃO DE ESPECIFICAÇÕES

A transformação de especificações, em lógica temporal (L.T.), para especificações informais declarativas deve obedecer a alguns requisitos para que se garanta uma relação clara entre estas. Estes requisitos dependem tanto da L.T. usada quanto do modelo de informação declarativa adotado, podendo apresentar um grau variável de complexidade. A princípio, toda especificação,

realizada em uma lógica temporal linear, pode ser manipulada de forma a adequar-se ao modelo abstrato de informação adotado. Aqui, restringe-se a discussão ao relacionamento entre especificações em uma GRTTL e o modelo abstrato de informação adotado, acima discutido.

As especificações em GRTTL apresentam certas características que devem ser transferidas e adequadas às características do modelo abstrato de informação. Destacam-se, abaixo, as principais características de cada um.

a) CARACTERÍSTICAS DA GRTTL

As características mais importantes nas especificações em GRTTL são:

1) Características que permitem a descrição da evolução do sistema ao longo do tempo - São representados por operadores temporais ($\odot, \Diamond, \Box, P, U$, etc) e sua axiomatização;

2) Características que permitem a descrição de condicionantes de tempo real- Neste item, têm-se prazos. Isto é normalmente descrito via limites inferiores e superiores de tempo e a axiomatização destes conceitos;

3) Características que permitem a descrição do não-determinismo de sistemas- A noção de não-determinismo está associada à idéia de multiplicidade de transições a partir de um estado, em contraste com a noção de determinismo. É interessante notar que não se tem um tratamento estocástico (média, variância, etc), muito embora seja não-determinístico;

4) Características que permitem a descrição do estado inicial - Na lógica temporal, uma das assertivas fornece os estados iniciais das variáveis, filas, contadores, etc.

b) CARACTERÍSTICAS DO MODELO DECLARATIVO DE INFORMAÇÃO

A descrição de uma entidade, respeitando o modelo abstrato de informação tem como principais características:

1 - Indicação e descrição de seus eventos;

2 - sequenciamento dos eventos;

3 - Limitações de tempo real;
4 - Condicionantes internos para a ocorrência de eventos;
5 - Descrição das entradas e saídas;
6 - Representação do estado inicial do sistema.

Na descrição da interação entre entidades, as principais características são:
1 - Interligação entre entradas e saídas de diferentes entidades;
2 - condicionantes mútuas para a ocorrência de eventos entre duas ou mais entidades.

4.2-3.2 - MAPEAMENTO DAS ESPECIFICAÇÕES FORMAIS EM DECLARATIVAS

Uma vez realizadas as especificações em GRTTL, informando-se as características acima citadas, é possível realizar o mapeamento para as características do modelo abstrato. Apresenta-se, a seguir, um esboço das principais relações entre os elementos usados para as especificações nos dois formalismos:

i) ENTIDADES - Especificações em GRTTL geralmente não explicitam a caracterização de entidades, tornando o mapeamento não direto. No entanto, é possível reconhecer, na descrição dos estados em GRTTL, quais sejam os elementos físicos que compõem o sistema em análise. Desta forma, distinguem-se as entidades a partir do conjunto de estados.

ii) EVENTOS - O conjunto de eventos da especificação em GRTTL é mapeado diretamente para o conjunto de eventos que compõe a especificação de cada entidade. Relacionado com a ocorrência de cada evento, existe um conjunto de procedimentos. Na especificação em GRTTL estes procedimentos são expressos pelos consequentes atribuidos à ocorrência de cada evento, excetuando-se a transição de estados.

iii) SEQUENCIAMENTO DE EVENTOS - O sequenciamento de eventos pode ser obtido das especificações em GRTTL, a partir do sequenciamento das transições de estado motivadas pelos eventos. Nesta etapa, são considerados os significados

dos operadores temporais sobre a dinâmica da planta. O não-determinismo, no sequenciamento de eventos (multiplicidade de caminhos possíveis a partir de um determinado estado), pode ser descrito por especificações em GRTTL, utilizando-se os operadores ∇ e Δ. Estas especificações podem ser diretamente mapeadas para ramificações de sequenciamentos, denominados rotinas.

iv)TEMPORALIDADE DAS TRANSIÇÕES - O conceito de temporalidade expressa os limites inferior e superior do intervalo de tempo entre a ocorrência de dois eventos determinados. Estes limites são as cotas inferior e superior de tempo, expressas em GRTTL, em função dos estados. Assim, o mapeamento é feito diretamente para intervalos de tempo entre eventos.

v) CONDICIONANTES - Os condicionantes para habilitação de ocorrência de um evento podem ser obtidos do conjunto de fórmulas das especificações em GRTTL. Isto pode ser feito eliminando-se as informações de sequenciamento das fórmulas relacionadas com o evento a ser condicionado.

vi) DESCRIÇÃO DAS ENTRADAS E SAÍDAS - Os elementos de comunicação física são, a princípio, as portas. A critério do usuário especificador e dependendo da aplicação, estes podem ser usados para apresentar informações adicionais a respeito do estado atual de uma entidade. Estes mesmos elementos, adicionados aos sinais de controle (flags), são usados como elementos de controle. As informações que caracterizam estes elementos são obtidas das fórmulas nas especificações em GRTTL.

vii) INTERAÇÃO ENTRE ENTIDADES - Nas especificações em GRTTL, a interação entre entidades não é identificada de forma direta. Isto se deve ao fato das entidades, conforme entendidas no modelo declarativo, não serem explicitadas diretamente nas especificações em GRTTL. No modelo declarativo, esta interação pode ser caracterizada através da 1) interligação entre os elementos de comunicação física e de controle e 2) das condicionantes mútuas entre entidades.

A interligação 1) não acrescenta novas informações às já obtidas acima (item *vi*), apenas relacionando canais de comunicação inter-entidades e não intra-entidades.

As condicionantes mútuas entre entidade, 2), são obtidas das especificações do controlador, na parte em que os eventos de planta são condicionados.

Normalmente, trata-se de variáveis, portas e bandeiras, comuns às entidades.

viii) ESTADO INICIAL - O mapeamento da equação que descreve o estado inicial do conjunto PLANTA-CONTROLADOR corresponde à criação de um Frame especial no modelo de simulação, denominado Frame de Condições Iniciais, que indica o estado inicial de cada uma das variaveis, filas, contadores e portas do modelo.

4.2-3.3 - METODOLOGIA PARA A TRANSFORMAÇÃO DE ESPECIFICAÇÕES

Para a transformação de especificações em GRTTL para modelos declarativos de informação, considera-se que estas especificações são apresentadas como a seguir:

a) Especificações da Planta

Como especificações de planta, entendem-se equações que apresentem:

• ESTADOS - Estes estados são apresentados como variáveis globais. Normalmente uma só equação é suficiente.

• EVENTOS - Os eventos possíveis são apresentados como variáveis locais. Este conjunto é normalmente descrito por uma só equação.

• TRANSIÇÕES - As equações de transição correspondem à maior parte das fórmulas do conjunto de especificações da planta. Neste conjunto, encontram-se informações sobre as transições de estado, condicionantes de tempo, sequenciamento de eventos e as consequências de cada transição.

b) Especificações de Controle

Como especificações de controle, entendem-se equações que apresentem:

• condições impostas para a ocorrência dos eventos de planta;
• descrição das ações de controle tomadas para cada evento ocorrido.

Nestas especificações, normalmente aparecem variáveis locais adicionais,

usadas para implementar a lógica de controle (portas e bandeiras).

Os passos a serem seguidos para a transformação citada são:

1) O usuário deve examinar as equações de planta (e controlador) e determinar quais as entidades a serem consideradas para a simulação;

2) Determinadas as entidades ,conforme o item acima, o usuário deve associar a cada entidade os respectivos estados e eventos especificados. Vinculado à ocorrência de cada evento, existe um conjunto de procedimentos que fornece as ações a eles correspondentes. Existem procedimentos relacionados exclusivamente aos eventos ou às condicionantes (regras) de controle;

3) O sequenciamento de eventos para cada entidade pode ser feito acompanhando-se as transições de estado. Dado um evento e um estado inicial, pode-se saber qual o próximo estado e, consequentemente, o(s) próximo(s) evento(s). Deste modo, sucessivamente, obtém-se todo o sequenciamento;

4) a temporalidade das transições é obtida a partir dos limites inferiores e superiores de tempo para a transição de estado correspondente à ocorrência de cada evento. Estas informações são reconhecidas a partir da comparação dos valores da variável tempo, "t", do antecedente e consequente das fórmulas em GRTTL. Nestas fórmulas, os limites inferior e superior de tempo são obtidos das expressões ($t \geq T + lim.inf$) e ($t \leq + lim.sup$) respectivamente, onde "T" é o valor da variável "t" no instante que a transição ocorre;

5) Os condicionantes são obtidos dos consequentes das fórmulas do controlador, excetuando-se as expressões (termos) de ação. Estas expressões (termos) de ação são reconhecidas através do símbolo "o" - next, do símbolo "*" - inserção de elemento em cauda de lista e do símbolo "|" - retirada de elemento da cabeça de lista. Todos estes elementos operam sobre as variáveis de controle (contadores, listas, flags, etc). Também observa-se aqui o aparecimento de regras que controlam o funcionamento da planta, mas que são internas a ela;

6) Elementos de comunicação física e de controle : A transformação destes elementos, das especificações em GRTTL para o modelo declarativo, é imediata, pois são as filas, contadores e flags das especificações em GRTTL. Para casos

em que sejam desejadas informações adicionais (estado atual, etc), pode-se utilizar as portas como registros;

7) A interação entre entidades é representada, no modelo declarativo, através da indicação da conexão de conteúdo de portas e bandeiras, que interligam entidades distintas;

8) As condições iniciais são obtidas da fórmula em GRTTL que traz esta informação e armazenada no frame de status. Neste caso, a passagem de uma especificação à outra é direta.

Um exemplo da transformação de especificações escritas em GRTTL para o modelo declarativo de informação é apresentado no capítulo seguinte.

4.3 - IMPLEMENTAÇÃO DO AMBIENTE INTEGRADO DE APOIO À TOMADA DE DECISÃO (AIATD)

A implementação do Ambiente Integrado de Apoio à Tomada de Decisão (AIATD) desenvolveu-se a partir de dois pontos fundamentais: requisitos funcionais dos módulos que compõem o ambiente (AIATD); e implementação do modelo abstrato de informação.

Considerando o modelo funcional do AIATD (seção 3.4) pode-se perceber que seus elementos funcionais dividem-se em duas classes operacionais. Uma classe composta pelos elementos funcionais que manipulam o conhecimento, e a outra composta pelos elementos de gerenciamento de operações e de comunicação.

Os elementos funcionais que manipulam o conhecimento são representados, a nível de objetos funcionais primários, por: Aquisitor de Conhecimento, Simulador, Processador de Conhecimento, e Base de Conhecimento.

Os elementos funcionais de gerenciamento de operações e de comunicação são representados, a nível de objetos funcionais primários, pelo Gerenciador de Sistema e pela Interface de Comunicação.

A implementação dos elementos funcionais difere bastante de uma classe para outra, devido, principalmente, aos fatores que direcionam a implementação de cada uma.

A implementação dos elementos funcionais manipuladores de conhecimento é direcionada por dois principais fatores: pela implementação do modelo abstrato de informação, e pelos requisitos funcionais de cada elemento manipulador de conhecimento. Estes fatores apresentam características muito bem definidas, e invariantes, quanto a seu aspecto operacional. Assim, a implementação destes elementos funcionais caracteriza-se por apresentar uma certa independência do contexto operacional.

A implementação dos elementos funcionais de gerenciamento de operações e de comunicação também é direcionada por dois principais fatores: pelos requisitos funcionais de cada elemento, e pelos requisitos de comunicação destes elementos com outros pertencentes ao ambiente, ou de fora dele. Entretanto, os requisitos de comunicação podem ser variantes conforme o contexto operacional a que se destine o ambiente. Isto leva a que tais elementos funcionais tenham sua implementação dependente destes requisitos de comunicação.

Como exemplo pode-se tomar a Interface de Comunicação. Sua implementação depende do tipo de comunicação a que esteja habilitada, isto é, se está capacitada a operar de forma aberta em rede, ou se apenas opera de forma dedicada

e mono-usuário. Também a implementação do Gerenciador de Sistema depende, em parte, das características de comunicação atribuídas ao ambiente. Quando capacitado a operar em rede tem atribuições muito mais complexas que quando operando de forma dedicada.

Devido às possíveis variações na operacionalidade de alguns dos elementos funcionais do ambiente, os aspectos de implementação são apresentados de forma distinta. São apresentadas, de forma detalhada, as características de implementação dos elementos manipuladores de conhecimento. Para os outros elementos descreve-se a informação a ser transmitida.

Dada a necessidade de processamento simbólico do modelo abstrato de informação adotado, a maior parte do AIATD foi implementada utilizando-se a linguagem Prolog. Foram implementados em linguagem C apenas as partes de análise estatística e apresentação gráfica, embora permaneça sempre aberta a possibilidade da inclusão de trechos de código em outras linguagens que não o Prolog, dada a conectividade desta linguagem com outras, como C e Pascal.

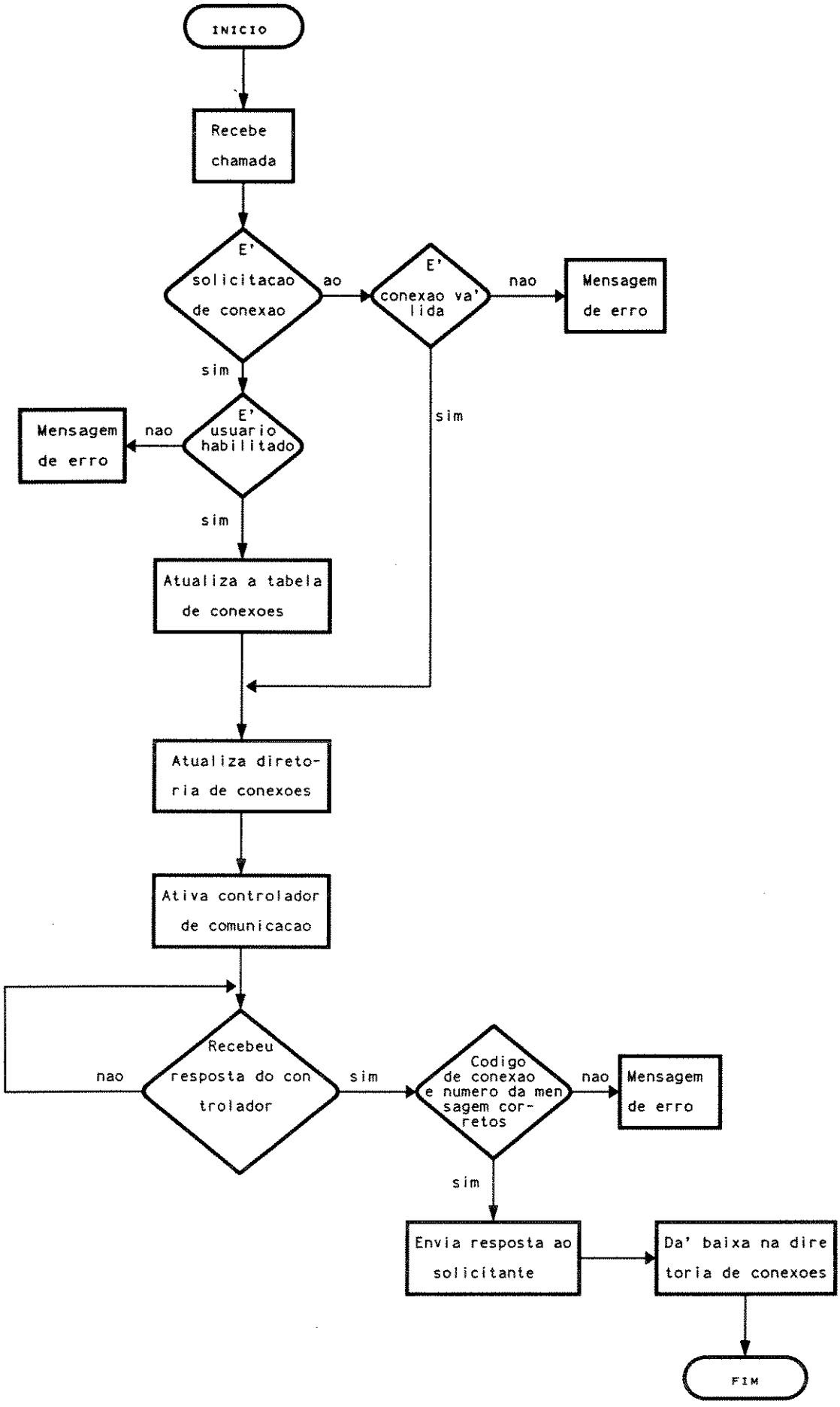
A apresentação dos aspectos de implementação dos objetos funcionais do ambiente dá ênfase à lógica operacional de cada um. Esta apresentação segue, práticamente, a mesma ordem adotada no item 3.4, quando da descrição do modelo funcional do ambiente.

4.3-1 - Interface de Comunicação

A Interface de Comunicação distribui suas funções de comunicação através de três distintos objetos: Gerenciador de Comunicações, Controlador de Comunicação Interativa, Controlador de Comunicação por Pacotes.

Na atual versão da implementação do ambiente, os passos de codificação e decodificação das mensagens para transmissão em rede de comunicação não foram desenvolvidos. Isto se deve à atual indisponibilidade de uma unidade de manufatura integrada por computador para uma completa caracterização do necessário ambiente de comunicação.

O sequenciamento de operações da Interface de Comunicação é apresentado a seguir, destacando-se as atividades de cada elemento funcional. Um diagrama representando este sequenciamento é apresentado no quadro 1.



Quadro 1 - Diagrama de operações da Interface de Comunicação

GERENCIADOR DE COMUNICAÇÕES

O Gerenciador de Comunicações é o primeiro elemento do AIATD a ser acessado por qualquer comunicação proveniente de um usuário. A operação do ambiente é iniciada através deste gerenciador, ativado por uma comunicação de chamada, recebida por ele.

Quando do recebimento de uma nova chamada, o Gerenciador de Comunicações executa os seguintes passos:

- verifica se o elemento chamador é habilitado a se comunicar com o ambiente. Em caso negativo envia mensagem de erro.
- atualiza a tabela de conexões estabelecidas, atribuindo ao usuário um código de conexão, ao qual é anexado seu endereço e o roteamento da estação usuária solicitante.
- atualiza a diretoria de conexões, onde ficam registrados: o número da mensagem, e seu destino.
- envia ao Controlador de Comunicação Interativa uma mensagem de inicialização de comunicação com o usuário, juntamente com o código da conexão. Esta mensagem é enumerada, internamente, como sendo a primeira.

Uma vez estabelecida a conexão, todas as outras comunicações desenvolvem-se de maneira padrão, descrita a seguir.

Quando do recebimento de uma mensagem, o Gerenciador de Comunicações desenvolve os seguintes passos:

- avalia a validade do código de conexão em sua tabela de conexões. Caso negativo envia mensagem de erro.
- enumera a mensagem e atualiza a diretoria de conexões.
- encaminha a mensagem ao Controlador de Comunicação Interativa se o solicitante for uma estação de trabalho. Se o solicitante for uma unidade da manufatura que atualiza dados, encaminha ao Controlador de Comunicação por Pacotes.

Quando do recebimento de uma resposta, a ser enviada ao usuário solicitante, o Gerenciador segue os seguintes passos:

- recebe a mensagem do controlador, juntamente com o código de conexão e o número da mensagem.
- verifica o código de conexão. Caso negativo retorna erro.
- envia a mensagem para o endereço indicado na tabela de conexões, juntamente

com a indicação do número da mensagem respondida.
- dá baixa na diretoria de conexões quanto ao número da referida mensagem.

CONTROLADOR DE COMUNICAÇÃO INTERATIVA

Ao receber uma mensagem, a partir do Gerenciador de Comunicações, este controlador desenvolve os seguintes passos:
- decodifica a mensagem, de acordo com os protocolos estabelecidos.
- encaminha a mensagem ao Gerenciador de Sistema, junto com o número desta e o código de conexão.

Ao receber a resposta de uma mensagem, proveniente do Gerenciador de Sistema, este controlador desenvolve os seguintes passos:
- codifica a mensagem, de acordo com os protocolos estabelecidos.
- encaminha a mensagem ao Gerenciador de Comunicação, juntamente com o código de conexão e o número da mensagem respondida.

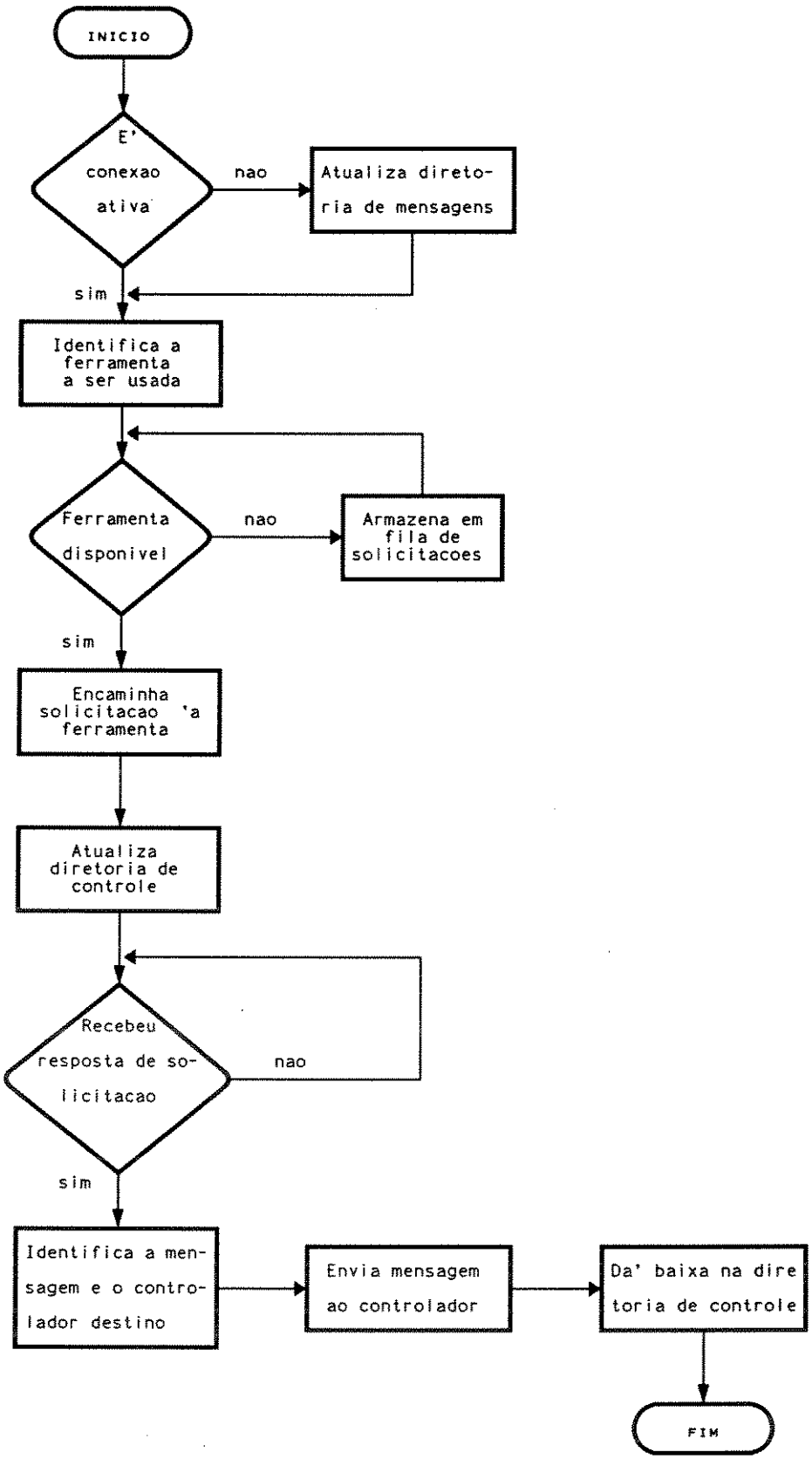
CONTROLADOR DE COMUNICAÇÃO POR PACOTES

Ao receber do Gerenciador de Comunicações uma mensagem, este controlador desenvolve os mesmos passos descritos para o Controlador de Comunicação Interativa.

A única mensagem a ser enviada por este controlador é de requisição de dados a alguma entidade da manufatura. Sua tarefa principal é codificar as mensagens recebidas, de acordo com os protocolos estabelecidos. Após isto a mensagem é enviada ao Gerenciador de Comunicações, com o endereço da entidade destino.

4.3-2 - Gerenciador de Sistema

O Gerenciador de Sistema encarrega-se da ativação ordenada dos objetos funcionais componentes do ambiente. O sequenciamento de suas operações é descrito a seguir, apresentando-se também, no quadro 2, um diagrama que representa este sequenciamento.



Quadro 2 - Diagrama de operações do Gerenciador de Sistema

Ao receber da Interface de Comunicação a solicitação de alguma operação do ambiente, este gerenciador desenvolve os seguintes passos:

- recebe a mensagem, com seu número e código da conexão
- verifica se o código de conexão já consta de sua diretoria de controle. Caso negativo, atualiza diretoria de controle, anexando este código.
- identifica o tipo de operação solicitada, e a qual ferramenta de análise se relaciona.
- avalia a disponibilidade da ferramenta de análise solicitada. Caso negativo armazena a mensagem em uma fila.
- encaminha a mensagem de solicitação.
- atualiza a diretoria de controle, indicando o número da mensagem de solicitação de operação, a que ferramenta foi endereçada, e de que controlador partiu a solicitação.

Ao receber, das ferramentas de análise do ambiente, a resposta de alguma solicitação, o Gerenciador de Sistema desenvolve os seguintes passos:

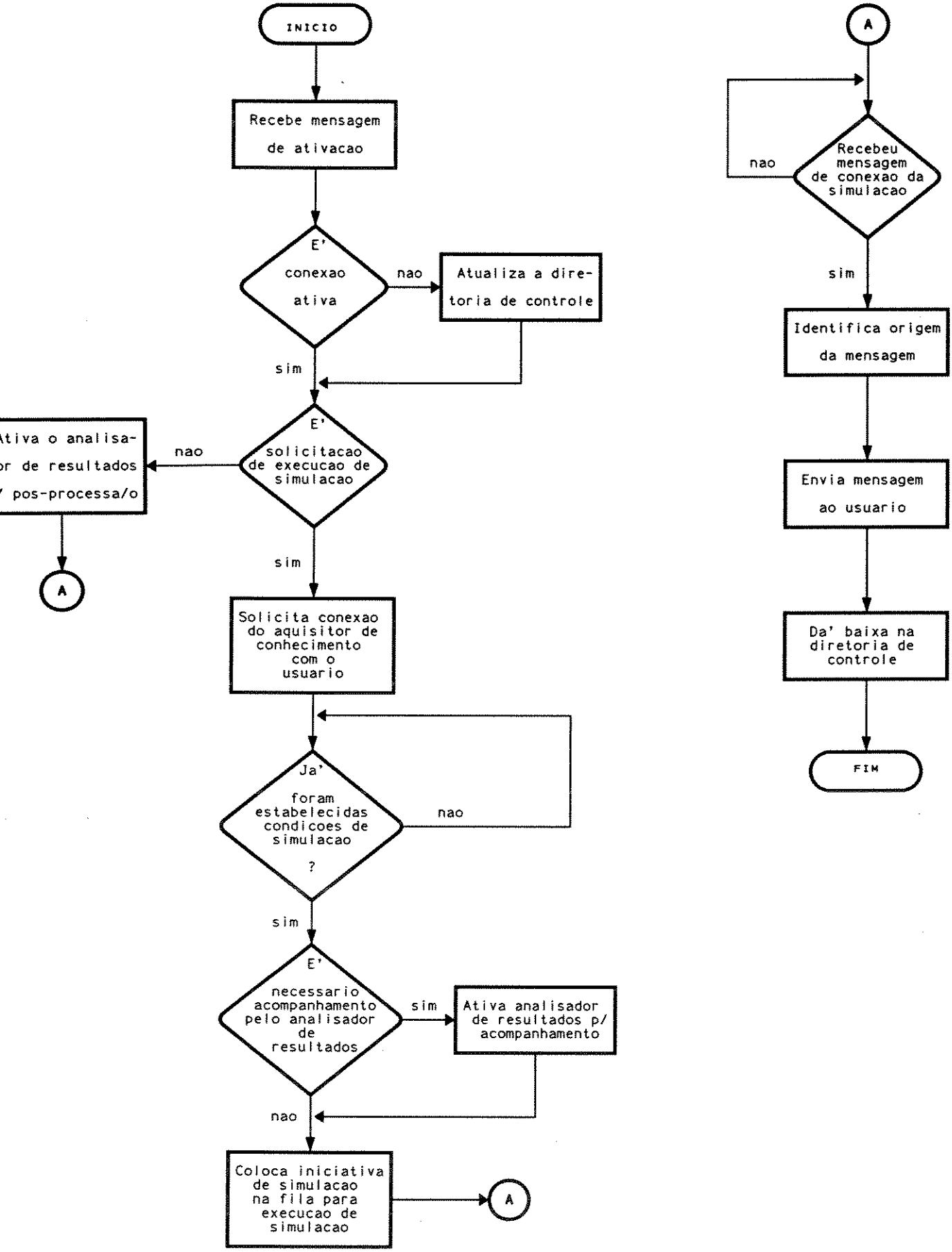
- verifica, em sua diretoria de controle: o código de conexão, o número da mensagem, e a qual controlador deve encaminhar a resposta.
- envia a resposta da mensagem ao controlador adequado.
- dá baixa em sua diretoria de controle da mensagem enviada.

4.3-3 - Simulador

O Simulador realiza suas funções através de três distintos objetos funcionais: Interface do Simulador, Executor da Simulação, e Analisador de Resultados. O sequenciamento de suas operações é descrito a seguir, destacando-se as atividades de cada elemento funcional. Apresenta-se também, no quadro 3 um diagrama que representa o sequenciamento do simulador como um todo.

INTERFACE DO SIMULADOR

A Interface do Simulador encarrega-se da recepção de mensagens e da ativação ordenada dos objetos componentes do simulador.



Quadro 3 - Diagrama de operações do Simulador

Ao receber alguma mensagem para a execução de simulação, ou de análise pós-processada de alguma simulação, esta interface desenvolve os seguintes passos:

- verifica se o código de conexão e o número da mensagem já constam de sua diretoria de controle. Caso negativo, atualiza a diretoria de controle, anexando estes valores.
- verifica se a solicitação corresponde à execução de simulação ou de análise de resultados de simulação.

Para o caso de ser uma solicitação de simulação, segue os seguintes passos:

- encaminha ao Gerenciador de Sistema uma solicitação de abertura de conexão entre o Aquisitor de Conhecimento e o usuário, para o estabelecimento das condições de simulação do sistema. Nesta encaminha-se um identificador (nome) do sistema a ser simulado.
- aguarda a mensagem de resposta desta solicitação, indicando que já foram estabelecidas as condições para a execução da simulação.
- verifica se existe a solicitação de acompanhamento do Analisador de Resultados. Caso afirmativo ativa este Analisador.
- insere a identificação do sistema a ser simulado, o número da mensagem de solicitação de simulação, e o código de conexão do usuário, na fila de ativações do executor de simulação.

Para o caso de ser uma solicitação de execução de análise pós processada de alguma simulação já realizada, esta interface ativa o Analisador de Resultados, indicando o identificador da simulação da qual se deseja a realização de análises.

Ao receber do Executor de Simulação, ou do Analisador de Resultados, a mensagem de conclusão de alguma solicitação, esta interface desenvolve os seguintes passos:

- verifica em sua diretoria de controle o código de conexão, e a qual mensagem se refere.
- envia esta resposta ao Gerenciador de Sistema.
- dá baixa em sua diretoria de controle da mensagem enviada.

EXECUTOR DA SIMULAÇÃO

O Executor de Simulação encarrega-se da lógica de simulação desenvolvida no ambiente. Desenvolve suas funções através dos elementos funcionais denominados: Controlador de Simulação, Monitor de Entidades, Ordenador Lógico, e Interface com o Processador de Conhecimento.

É descrito, a seguir, o sequenciamento de uma simulação, com o propósito de enfatizar a lógica de simulação. Embora indicados quando da menção de ativação de cada elemento funcional, nesta descrição são desprestigiados todos os aspectos de uma execução concorrente de diferentes simulações.

Nos quadros 4 e 5 apresenta-se um diagrama representando esta lógica de simulação.

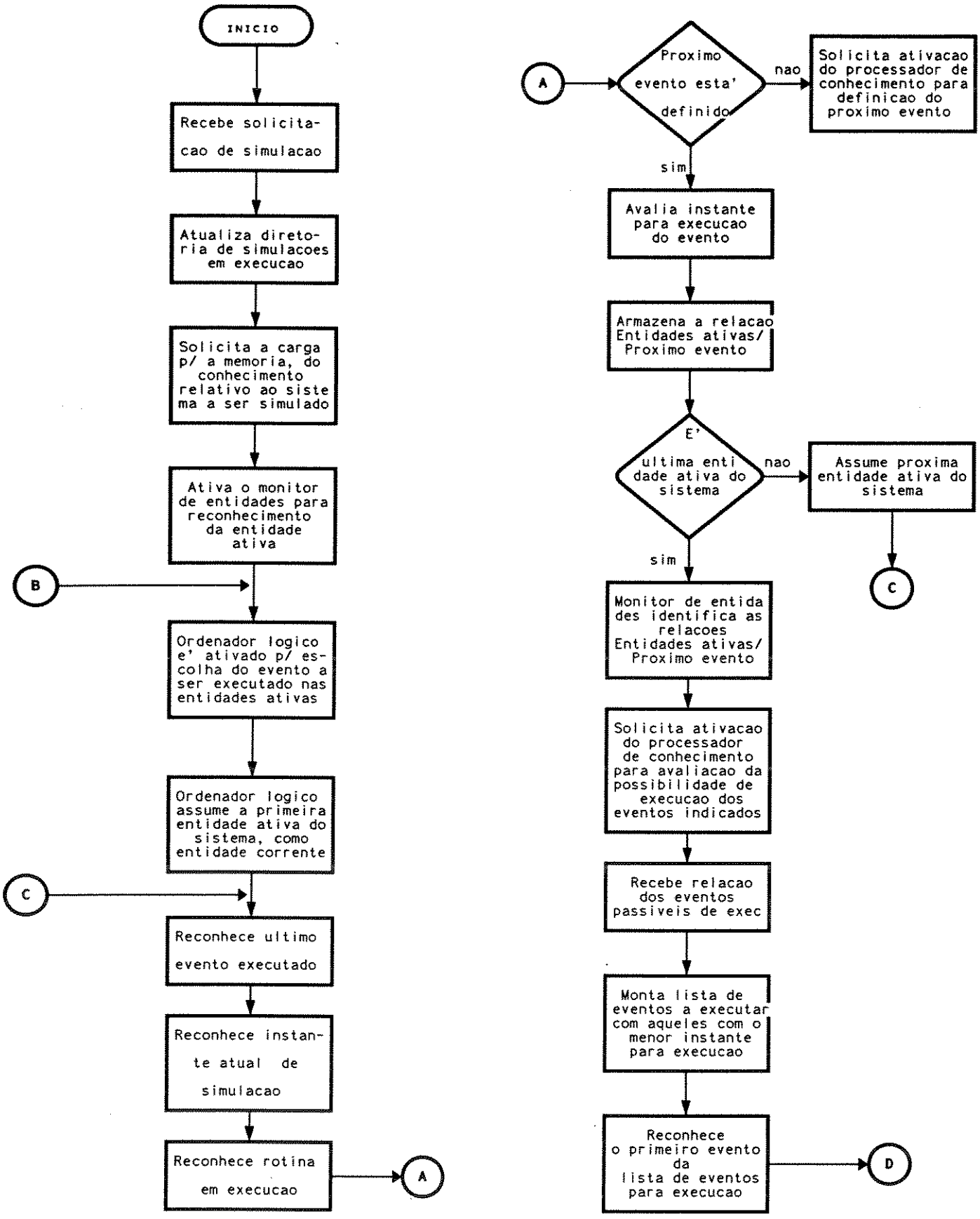
A atividade do Executor de Simulação inicia-se quando o Controlador de Simulação recebe, da Interface do Simulador, uma mensagem para a realização de uma simulação. A partir do recebimento desta mensagem desenvolve os seguintes passos:

- Atualiza sua diretoria de simulações em execução. Nesta constam o identificador do sistema sob simulação, o código de conexão de usuário, e o número da mensagem originadora da simulação.
- solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, a realização da carga, para a memória de trabalho, das informações (frames, procedimentos, e regras) relativas ao sistema a ser simulado.
- ativa o Monitor de Entidades, indicando quais as entidades ativas para esta simulação, o identificador do sistema sob simulação, e atribui um número indicativo da ativação do Monitor (em caso inicial 1), para tal identificador.

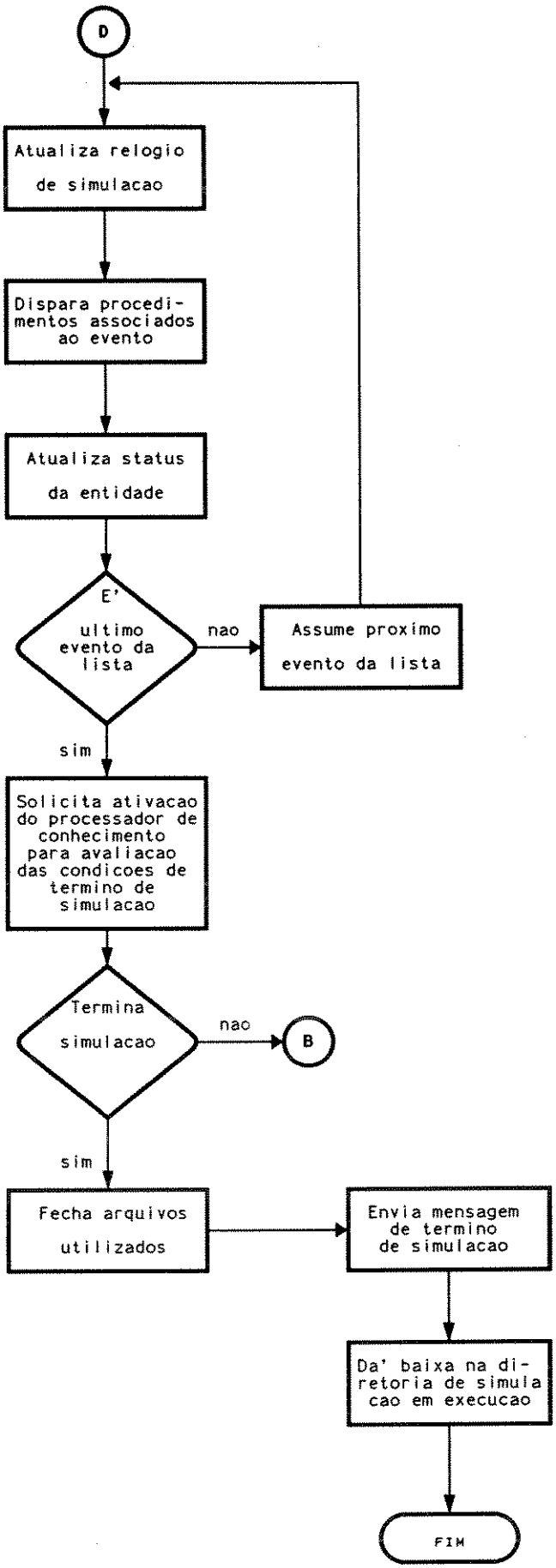
O Monitor de Entidades, uma vez ativado, reconhece quais as entidades ativas do sistema sob simulação, e ativa o módulo Ordenador Lógico, para todas estas entidades.

A sequência de operações do Ordenador Lógico, sobre cada entidade ativa do sistema sob simulação é a seguinte:

- reconhece qual o último evento executado pela entidade, e em qual de suas



Quadro 4 - Lógica de Simulação



Quadro 5 - Lógica de Simulação (cont.)

rotinas.

- reconhece qual o instante atual de simulação.
- encontra na descrição da entidade qual a próxima ocorrência após tal evento. No caso em que vários eventos podem acontecer após este, ativa o Processador de Conhecimento para inferir qual deva ser o próximo evento, dadas as atuais condições. Para tal ativação indica qual a entidade sob análise, qual o último evento e a qual rotina pertence, qual a base de regras a ser consultada, e qual o identificador de simulação.
- Recebe do Processador de Conhecimento qual o próximo evento a ser executado pela entidade.
- Avalia o instante mínimo para execução do evento escolhido.

Uma vez realizada a escolha do próximo evento a ser executado, para cada entidade ativa, o controle de operações é devolvido ao Monitor de Entidades. Este, então, por sua vez, desenvolve os seguintes passos:

- reconhece quais os eventos indicados como candidatos para próxima ocorrência, em cada entidade.
- avalia a possibilidade de ocorrência de cada um destes eventos, frente ao estado atual do sistema sob simulação. Esta avaliação é realizada através de ativação sequencial do Processador de Conhecimento, para cada evento indicado. Para esta ativação o Monitor de Entidades indica o evento a ser avaliado, sua rotina, qual a base de regras a ser consultada, e qual o identificador de simulação. Recebe como resposta "verdadeiro", ou "falso", para cada consulta realizada. Todos aqueles que estiverem aptos a ocorrer continuam a ser avaliados para ocorrência.
- avalia quais destes eventos apresentam o menor instante previsto de ocorrência. Todos os outros são descartados.
- monta uma relação de eventos a ocorrer, composta pelos eventos aprovados pelo critério temporal acima. Esta relação compõe uma mensagem enviada ao Controlador de Simulação.
- envia ao Controlador de Simulação uma mensagem composta por esta relação, pelo identificador do sistema sob simulação, e pelo número indicativo da presente ativação.

Uma vez recebida a relação de eventos a ocorrer, o Controlador de Simulação desenvolve os seguintes passos:

- reconhece e retira o primeiro evento da relação.
- atualiza o relógio de simulação com o instante calculado para ocorrência deste evento.
- dispara a execução dos procedimentos associados ao evento.
- atualiza o status da entidade.
- verifica a existência de algum outro evento que componha a relação recebida. Caso positivo volta ao primeiro passo acima.
- ativa o Processador de Conhecimento para avaliação das condições de término de simulação. Para esta ativação apresenta o identificador do sistema sob simulação, a base de regras a ser consultada, e o número da mensagem de solicitação da simulação. Caso a resposta seja negativa, retorna ao terceiro passo da sequência do executor de simulação, acrescendo de 1 o número indicativo da ativação do monitor, ativando novamente o Monitor de Entidades.
- fecha todos os arquivos utilizados durante a simulação corrente.
- envia à Interface do Simulador a indicação de término de simulação, indicando o número da mensagem originadora da simulação, e o código de conexão do usuário.
- dá baixa, na diretoria de simulações em execução, do número identificador do sistema sob simulação.

ANALISADOR DE RESULTADOS

O Analisador de Resultados encarrega-se da avaliação de simulações já realizadas, e do acompanhamento de simulações em execução, quando assim solicitado pelo usuário. Apesar de aqui especificado, este Analisador não se encontra completamente especificado. O analisador desenvolve suas atividades com base em quatro objetos funcionais: Formatador de Resultados, Processador Estatístico, Animador Gráfico, e Interface com Processador de Conhecimento.

O Formatador de Resultados ao receber uma ativação, emitida pela Interface do Simulador, executa os seguintes passos:
- verifica o tipo da ativação.
- encaminha mensagem ao usuário, solicitando informações a respeito do tipo e formato do acompanhamento desejado, e sobre quais arquivos ou variáveis se referem.
- ao receber a resposta desta mensagem, estabelece o formato apropriado das

futuras comunicações a serem enviadas ao usuário.
- atualiza a diretoria de análises, com os indicadores da ativação.
- ativa o Processador Estatístico, ou o Animador Gráfico, dependendo do tipo da solicitação do usuário. Nesta ativação encaminha o identificador da simulação a ser analisada, o tipo de análise a ser efetuada, e sobre quais arquivos ou variáveis se referem.
- recebe cada mensagem de retorno de informações e verifica se indica final de análise. Em caso afirmativo, dá baixa, em sua diretoria de análises, dos indicadores da ativação.
- coloca a mensagem no formato apropriado ao usuário, e a envia à Interface do Simulador.
- retorna a três passos atrás nesta sequência, aguardando novas informações a serem formatadas.

Tanto o Processador Estatístico como o Animador Gráfico apresentam o mesmo sequenciamento de atividades. Quando de sua ativação, recebem do Formatador de Resultados a indicação do tipo de análise a ser realizada, e sobre quais informações devem atuar.

Podem ser ativados para dois diferentes tipos de operação: acompanhamento de simulação, e análise pós-processada de simulações.

Uma vez ativados para acompanhamento de simulações desenvolvem os seguintes passos:
- solicita a abertura de um canal de comunicação com o Processador de Conhecimento, através da Interface com o Processador de Conhecimento. Indica quais as variáveis que devem ser acompanhadas e a quais entidades elas se referem.
- aguarda recebimento da informação.
- realiza a análise solicitada (processamento estatístico ou animação gráfica) sobre cada informação recebida.
- encaminha o resultado da análise ao Formatador de Resultados, indicando a qual identificador de simulação se refere.

Quando ativados para análise pós-processada de uma simulação, tanto o Processador Estatístico como o Animador Gráfico desenvolvem os seguintes passos:
- solicita ao Processador de Conhecimento, através da Interface com Processador de Conhecimento, o conjunto de informações necessário ao processamento

requerido.

- procede ao processamento requerido, enviando uma ou várias mensagens para o usuário, através do Formatador de Resultados.
- uma vez terminado o processamento, encaminha mensagem de conclusão de análise para o Formatador de Resultados.

A Interface com o Processador de Conhecimento responsabiliza-se por mapear as chamadas, do Processador Estatístico e do Animador Gráfico, para o Processador de Conhecimento. Sua implementação depende tanto do tipo de mensagem recebida pelo Processador de Conhecimento como do tipo daquelas geradas pelos elementos componentes do Analisador de Resultados.

Na presente implementação, este mapeamento é desnecessário, uma vez que ocorre perfeita compatibilidade entre as mensagens geradas pelos elementos do Analisador de Resultados e as recebidas pelo Processador de Conhecimento.

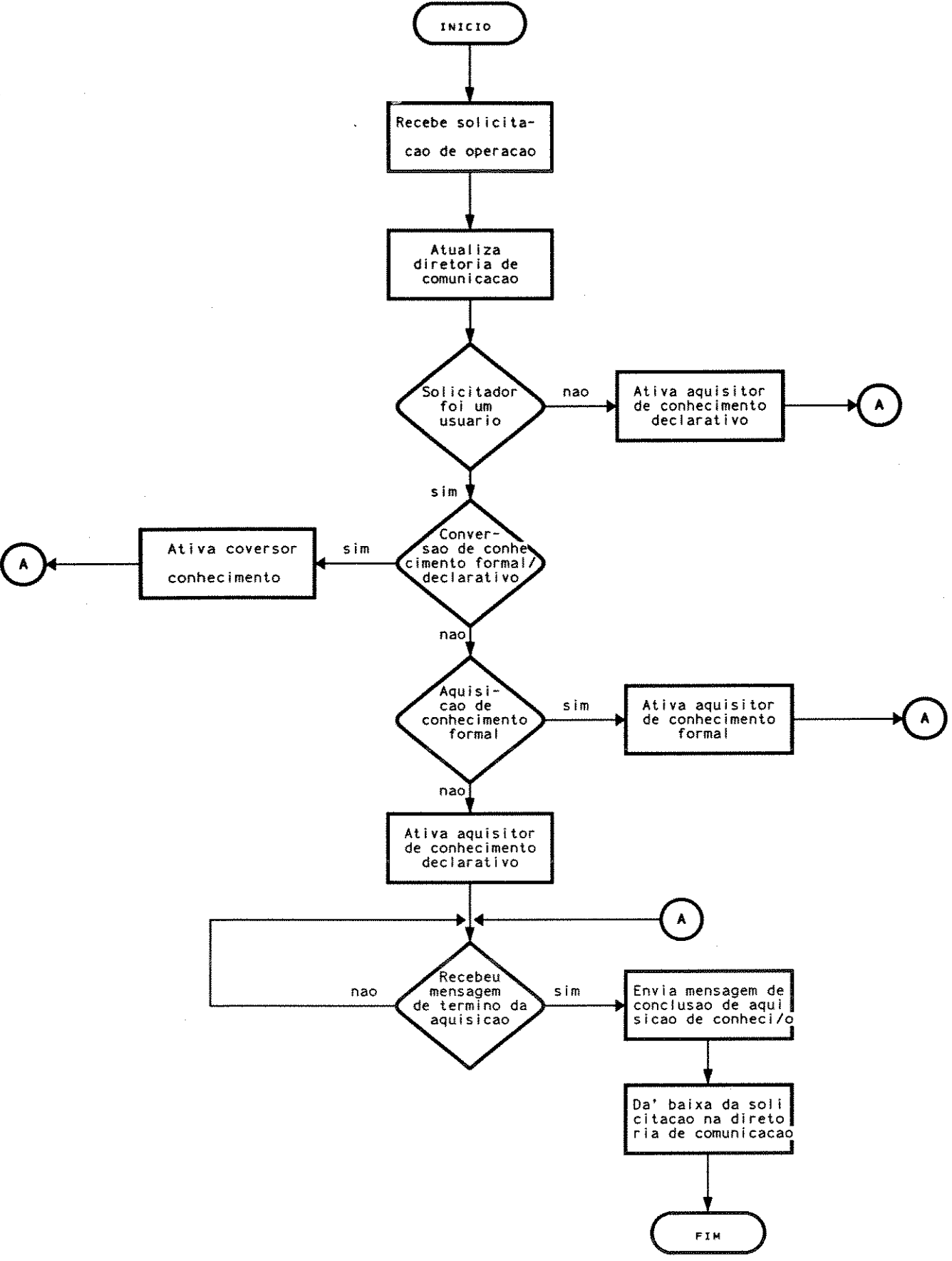
4.3-4 - Aquisitor de Conhecimento

O Aquisitor de Conhecimento realiza a aquisição de conhecimento para o ambiente. Executa esta aquisição através de sessões de diálogo com o usuário. Tem por sua incumbência a aquisição de dois tipos de conhecimento: conhecimento declarativo e conhecimento formal. Desenvolve, ainda, a conversão de modelos de simulação gerados segundo uma especificação formal, para a forma declarativa utilizada pelo Executor de Simulação do ambiente.

Sua ativação é feita pelo Gerenciador de Sistema, em duas situações: quando solicitado por um usuário, e quando solicitado pelo Simulador. No primeiro caso o usuário pode optar entre as diferentes formas de entrada de conhecimento, enquanto no segundo a ativação é específica para a edição de condições de realização de uma dada simulação.

Em ambos os casos, suas atividades são desenvolvidas através do: Gerenciador de Aquisição de Conhecimento, Aquisitor de Conhecimento Formal, Aquisitor de Conhecimento Declarativo, Conversor de Conhecimento Formal/Declarativo, e Interface com Processador de Conhecimento.

O sequenciamento de operações do Aquisitor de Conhecimento é apresentado a seguir, destacando-se as atividades de cada elemento funcional. Um diagrama representando este sequenciamento é apresentado no quadro 6.



Quadro 6 - Diagrama de Operações do Aquisitor de Conhecimento

GERENCIADOR DE AQUISIÇÃO DE CONHECIMENTO

O Gerenciador de Aquisição de Conhecimento encarrega-se do diálogo com o usuário, para estabelecimento do tipo de aquisição de conhecimento a ser desenvolvido. Também ativa os elementos aquisitores, conforme o estabelecido pelo usuário, ou o solicitado pelo Simulador.

Ao ser ativado, a partir de solicitação do usuário, este gerenciador segue os seguintes passos:

- atualiza sua diretoria de comunicação com o código de conexão, e o número da mensagem originadora da ativação.
- envia mensagem consultando o usuário sobre o tipo de entrada de conhecimento desejado.
- verifica a qual elemento do aquisitor de conhecimento corresponde a resposta recebida.
- ativa o elemento solicitado para comunicação com o usuário.

Para o caso em que a ativação procede do Simulador, estabelece uma ativação do Aquisitor de Conhecimento Declarativo, para aquisição das condições de execução de uma simulação já requerida. Esta aquisição se processa com uma ativação, primeiro do Editor de Frames, e depois, do Editor de Regras.

O Editor de Frames é ativado para a entrada de valores iniciais de simulação, e da matriz de ligação. Para isto já são carregados para a memória de trabalho os frames das entidades ativas e de status.

O Editor de Regras é ativado para edição de regras de sistema. Já recebendo, nesta ativação, a carga para memória de trabalho do conjunto de regras de sistema, composto para o sistema a ser simulado.

No caso do Gerenciador de Aquisição de Conhecimento receber uma comunicação de término de aquisição de conhecimento, este gerenciador segue os seguintes passos:

- verifica em sua diretoria de comunicação, a qual mensagem originadora corresponde a conclusão recebida.
- envia mensagem de conclusão de aquisição de conhecimento ao Gerenciador de Sistema.
- dá baixa em sua diretoria de comunicação, da mensagem originadora da ativação, com o respectivo código de conexão.

AQUISITOR DE CONHECIMENTO FORMAL

O Aquisitor de Conhecimento Formal corresponde a um editor de textos. Além dos caracteres em código ASCII, dispõe de alguns símbolos especiais que fazem parte da sintaxe da lógica temporal utilizada no ambiente.

Uma vez ativado, solicita ao usuário a entrada dos principais identificadores do modelo a editar. Após isto atua de forma interativa com o usuário, procedendo à edição de expressões em lógica temporal. Após a edição de cada expressão pode-se proceder a uma verificação na consistência de sua sintaxe. Esta verificação é feita a partir de consultas encaminhadas a uma específica base de conhecimento, através da Interface com o Processador de Conhecimento.

Uma vez concluída a edição do modelo, este aquisitor solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, que proceda à armazenagem do modelo na base de conhecimento.

AQUISITOR DE CONHECIMENTO DECLARATIVO

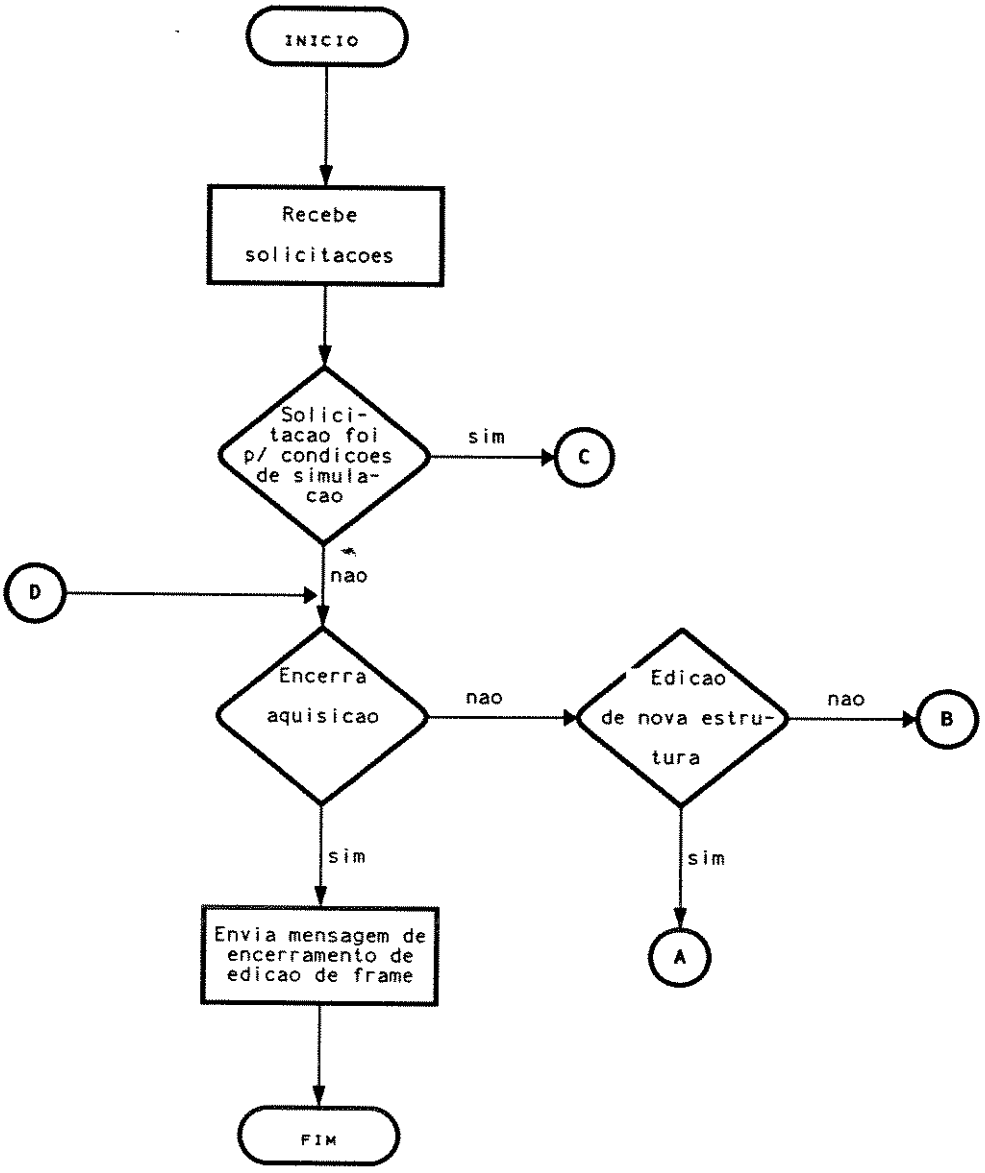
O Aquisitor de Conhecimento Declarativo encarrega-se do diálogo com o usuário para a aquisição de conhecimento segundo o modelo abstrato de informação desenvolvido.

Suas atividades são desenvolvidas através de três diferentes editores: Editor de Frames, Editor de Regras, e Editor de Procedimentos.

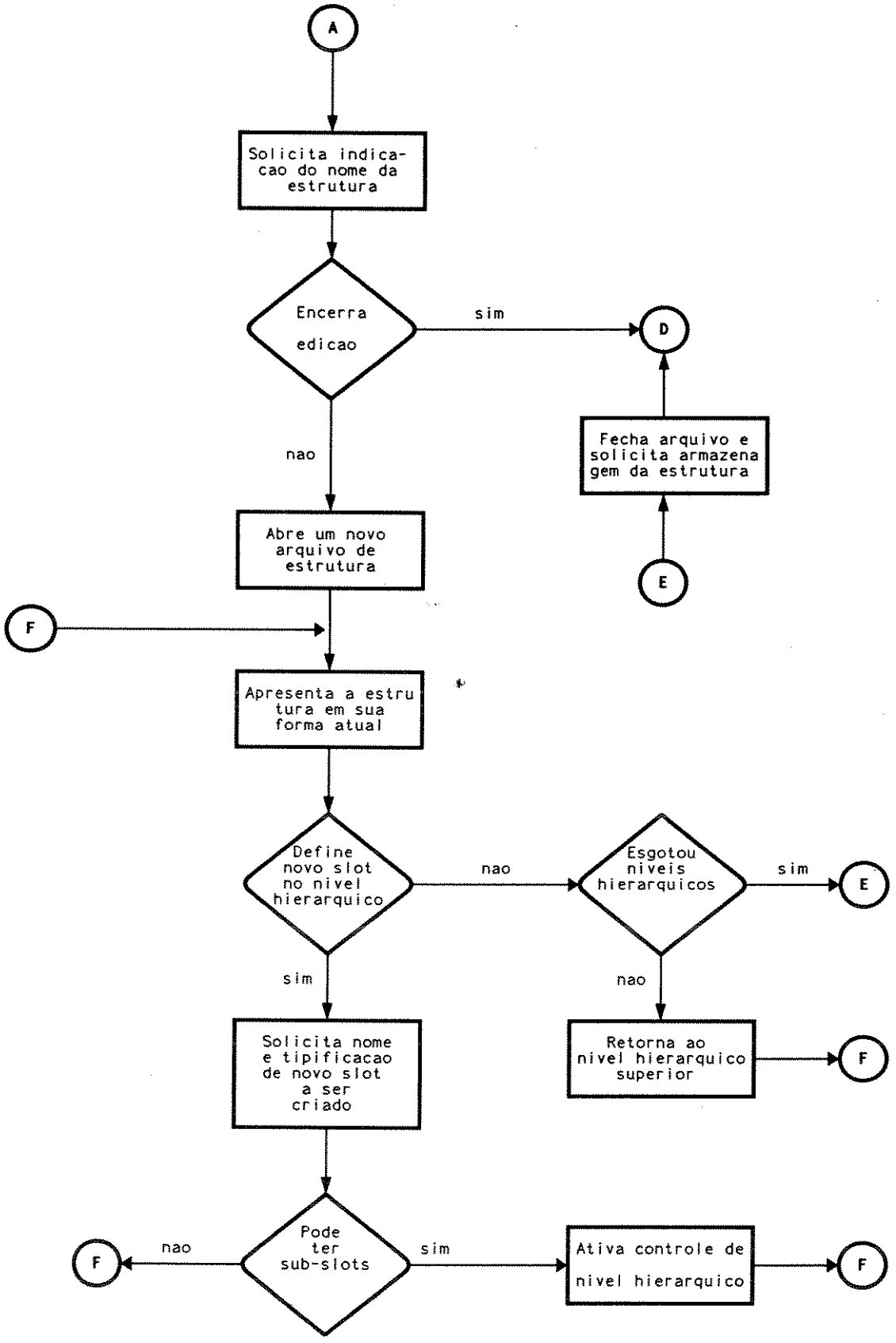
a) Editor de Frames

O Editor de Frames pode ser ativado em duas diferentes situações: por solicitação do usuário, para aquisição de conhecimento referente a uma entidade ou sistema; por solicitação do Simulador, para edição de condições de simulação.

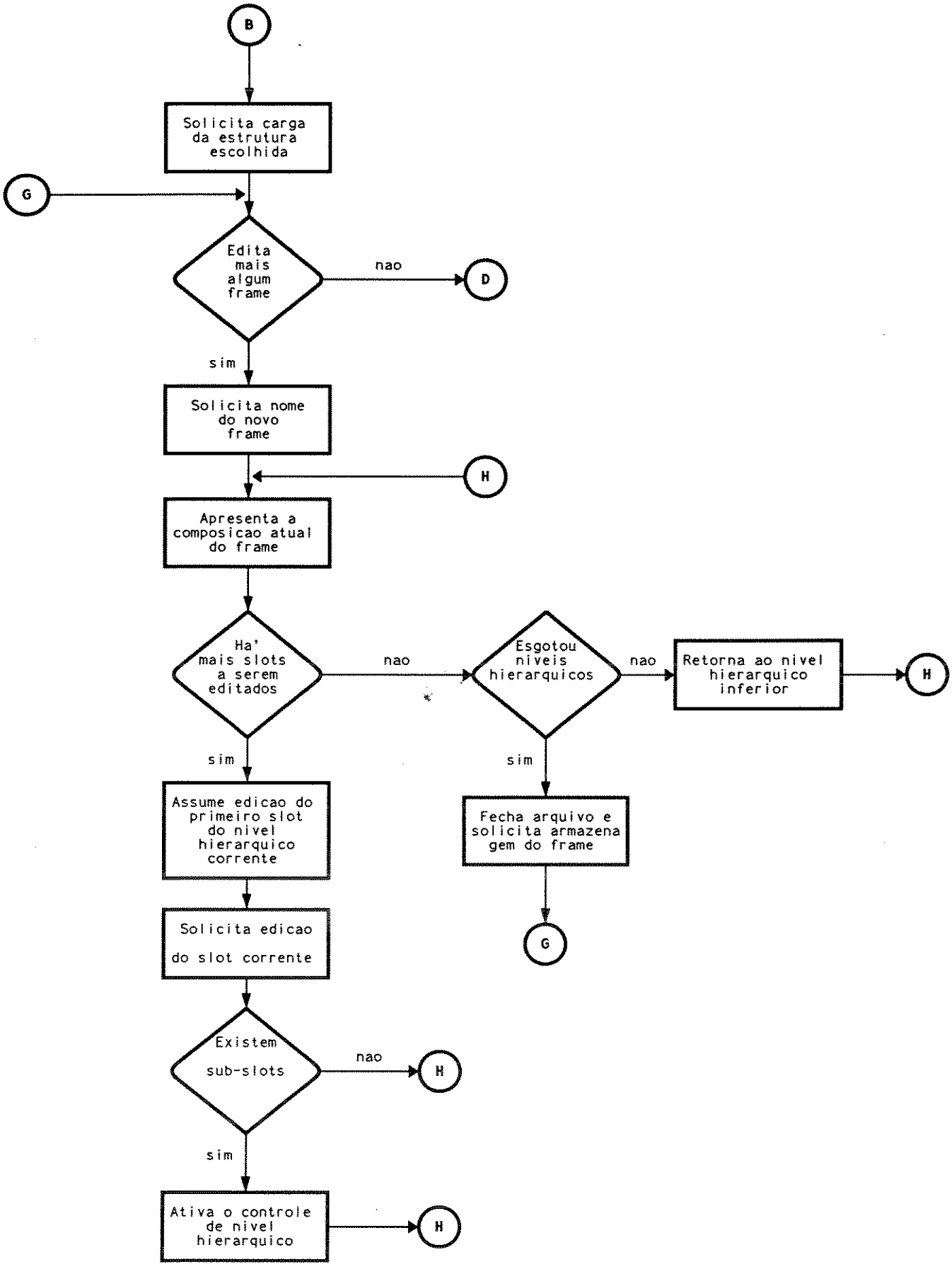
O sequenciamento de operações do Editor de Frames é descrito a seguir, de acordo com as seguintes possibilidades de atuação. Este sequenciamento também é representado nos quadros 7 a 10. No quadro 7 representa-se as possíveis chamadas a este editor, no quadro 8 a edição de uma nova estrutura de frame, no quadro 9 a edição de um frame genérico com estrutura já definida, e no quadro 10 a edição de condições de simulação.



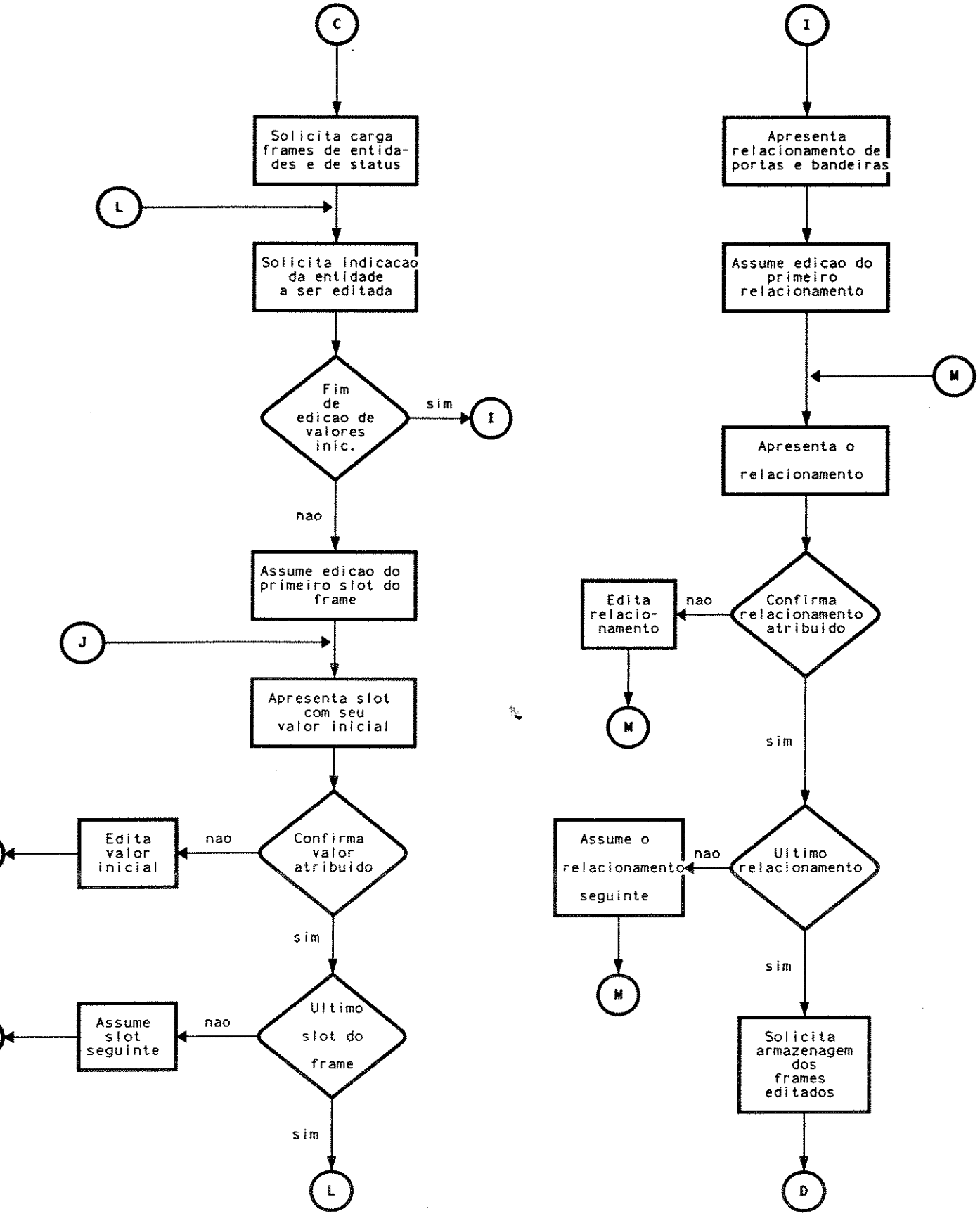
Quadro 7 - Chamadas ao Editor de Frames



Quadro 8 - Edição de nova estrutura para frame



Quadro 9 - Edição de frame genérico



Quadro 10 - Edição de Condições de Simulação

Uma vez ativado pelo Gerenciador de Aquisição de Conhecimento, por solicitação do usuário, este editor desenvolve os seguintes passos:

- apresenta ao usuário os nomes das estrutura de frames já definidas.
- solicita do usuário a escolha entre edição: de uma nova estrutura de frames; ou de frame pertencente a alguma das estruturas já definidas. Uma última opção corresponde ao término de edição de frames.
- verifica a resposta do usuário. Para cada uma das opções escolhidas, desenvolve um dos sequenciamentos abaixo. No caso de término de edição de frames, envia mensagem neste sentido ao Gerenciador de Aquisição de Conhecimento.

No caso de criação de uma nova estrutura, desenvolve a seguinte sequência:

= solicita a indicação do nome da nova estrutura de frame. Uma opção é encerrar a edição de novas estruturas, caso em que retorna ao primeiro passo da sequência do Editor de Frames.

= abre um arquivo para a nova estrutura.

= apresenta a estrutura em sua forma atual (vazia no caso de início de definição).

= questiona o usuário da necessidade de abertura de um novo slot. Em caso negativo reassume o nível hierárquico anterior e volta para o terceiro passo desta sequência. Quando não houver outro nível hierárquico anterior vai ao encerramento de definição da estrutura, quatro passos abaixo nesta sequência.

= solicita do usuário o nome do slot a ser inserido na estrutura.

= solicita a escolha da tipificação da informação a ser incluída no slot.

= questiona se podem ser abertos sub-slots do slot em questão. Caso positivo, ativa o controle de níveis hierárquicos e retorna ao terceiro passo desta sequência, para a definição recursiva de cada sub-slot.

= encerra a edição da nova estrutura, encaminha o arquivo de nova estrutura de frame para armazenagem na base de conhecimento, através da Interface com o Processador de Conhecimento.

= retorna ao primeiro passo da sequência do editor de frames.

No caso de escolha de edição de um frame com estrutura já definida, desenvolve a seguinte sequência:

= solicita ao Processador de Conhecimento, através da Interface com Proces-

sador de Conhecimento, a carga para memória de trabalho da estrutura escolhida.

= solicita ao usuário a indicação do nome do novo frame a ser editado. Caso não haja frame a ser editado retorna ao primeiro passo da sequência do editor de frames.

= apresenta a composição atual do frame.

= questiona se existem slots a serem editados. Em caso negativo reassume o nível hierárquico anterior e volta ao terceiro passo desta sequência. Quando não houver nível hierárquico anterior vai ao encerramento de edição do frame, quatro passos abaixo nesta sequência.

= procede à edição de cada slot do frame. Um tutor de edição de frames pode acompanhar o desenrolar da edição, orientando quanto à composição do slot. O tutor também verifica a sintaxe desenvolvida na composição de cada slot.

= questiona se pode ser aberto sub-slot do slot em edição. Em caso negativo retorna ao terceiro passo desta sequência.

= ativa o controle de níveis hierárquicos e retorna ao terceiro passo desta sequência.

= encerra a edição do frame. Ativa o tutor de edição de frames, no sentido de verificar a consistência e integridade da composição do frame.

= encaminha o arquivo de composição do frame para armazenagem na base de conhecimento, através da Interface com o Processador de Conhecimento.

= retorna ao segundo passo desta sequência.

A edição dos frames para simulação corresponde a casos particulares do frame acima descrito, mas por sua importância são aqui descritos. No caso de edição de frames para simulação, pode-se editar, tanto frames de entidades, como frames que descrevam as condições de simulação de sistemas. Os frames de entidade descrevem cada entidade de uma forma independente do ambiente em que possa estar inserida. Os frames descritores de condições de simulação de sistemas apresentam: uma descrição dos interrelacionamentos que acontecem entre as entidades que venham a compor um sistema a ser simulado; e as condições iniciais de simulação de tal sistema. Nesta implementação, as descrições de sistemas encontram-se em um frame denominado de "status", cujas instâncias correspondem a cada sistema modelado.

No caso de edição de frame de entidade, este editor desenvolve a seguinte

sequência:

= solicita ao Processador de Conhecimento, através da Interface com Processador de Conhecimento, a carga para memória de trabalho da estrutura do frame de entidade para simulação.

= solicita, ao usuário, a indicação do nome da entidade, o qual é atribuído ao frame em edição. Caso não haja novo frame a ser editado, retorna ao primeiro passo do editor de frames.

= atribui novos valores de versão, data, e hora, ao frame sob edição.

= solicita do usuário a indicação da classe (entidade mãe) desta entidade.

= solicita ao Processador de Conhecimento, através da Interface com Processador de Conhecimento, a verificação da existência de algum frame com o nome indicado como classe. Caso positivo, solicita sua carga para a memória de trabalho, e altera a composição de seu slot "membros", acrescentando o nome do frame em edição. Em caso negativo, cria um frame vazio, a cujos slots "nome" e "classe" são atribuídos o nome indicado como classe.

= apresenta o conjunto de portas já definidas (nenhuma no início da edição, ou no caso de não haver).

= solicita ao usuário a entrada do nome da porta a ser editada. Caso não haja portas, vai para a edição de bandeiras, dois passos abaixo nesta sequência.

= edita a composição da porta, solicitando ao usuário seus valores característicos, e retorna à apresentação do conjunto de portas, dois passos atrás nesta sequência.

= apresenta o conjunto de bandeiras já definidas.

= solicita ao usuário a entrada dos nomes de bandeiras a serem definidas.

= apresenta ao usuário o conjunto de slots definidos como informações.

= solicita do usuário o nome de um sub-slot a ser editado como informação. Caso não hajam sub-slots a serem definidos vai para a edição de eventos, quatro passos abaixo nesta sequência.

= solicita a escolha da tipificação da informação a ser incluída nesse sub-slot, ou apenas confirmação, em caso de já estar corretamente definido.

= questiona se podem ser abertos novos sub-slots deste sub-slot em questão. Caso positivo, questiona até quantos e retorna a dois passos atrás neste sequência para uma definição recursiva destes novos sub-slots.

= retorna ao nível anterior de profundidade no sub-slot em definição, e remete o fluxo a três passos atrás nesta sequência, para a consideração de

novos sub-slots.

= apresenta ao usuário o conjunto de eventos definidos no frame (nenhum no caso de início de edição).

= solicita ao usuário a entrada do nome do evento a ser editado. Caso não haja novos eventos a serem editados vai para a edição de rotinas dois passos abaixo nesta sequência.

= realiza a composição de todos os atributos do evento, solicitando ao usuário seus valores característicos, e retorna à apresentação dos eventos, dois passos atrás nesta sequência.

= apresenta o conjunto de rotinas já definidas (nenhuma no início da edição).

= solicita ao usuário a entrada do nome de nova rotina a ser editada. Caso não hajam novas rotinas a serem editadas, vai para o encerramento da edição do frame, dois passos abaixo nesta sequência.

= realiza a composição da rotina, solicitando ao usuário a indicação do próximo evento a ocorrer, ou quais são eles, no caso de poderem ser vários. Retorna à apresentação das rotinas, dois passos atrás nesta sequência.

= encerra a edição do frame. Ativa o verificador de composição de frames, para a avaliação de consistência, integridade do frame, e para a anexação de elementos de controle referentes à edição de procedimentos e regras.

= solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, a armazenagem deste frame na base de frames.

= retorna ao segundo passo desta sequência para edição de novos frames.

No caso de edição de frames de condições de simulação de sistemas, este editor desenvolve a seguinte sequência:

= solicita ao Processador de Conhecimento, através da Interface com Processador de Conhecimento, a carga para memória de trabalho de: estrutura do frame de status, e nome de todas as entidades já especificadas.

= solicita, ao usuário, a indicação do nome do sistema a ser composto para simulação. Este nome é atribuído ao slot "tipo" do frame status que especifica o sistema. Caso a escolha do usuário seja por encerrar, retorna ao primeiro passo do editor de frames.

= solicita ao Processador de Conhecimento, através da Interface com Processador de Conhecimento, a verificação da existência de algum frame de sta-

tus com o slot "tipo" contendo o mesmo nome indicado para o sistema. Caso positivo, solicita sua carga para a memória de trabalho.

= atribui novos valores de versão, data, e hora, ao frame sob edição.

= apresenta ao usuário o nome das entidades já especificadas, e as entidades já escolhidas para comporem o sistema.

= questiona a necessidade de incluir novas entidades ao sistema. Caso negativo vai para a carga dos frames na memória de trabalho, dois passos abaixo nesta sequência.

= retorna à apresentação das entidades escolhidas para o sistema, dois passos atrás nesta sequência.

= solicita ao Processador de Conhecimento, através da Interface com Processador de Conhecimento, a carga para memória de trabalho de: estrutura do frame de entidades, frames de todas as entidades escolhidas para comporem o sistema a ser simulado.

A partir deste ponto, a sequência desenvolvida pelo editor de frames é a mesma do caso em que este editor foi ativado por solicitação do Simulador, descrita a seguir.

O Editor de Frames também pode ser ativado para edição de frames de condições de simulação, por solicitação do Simulador. Neste caso, já são carregados automáticamente os frames de entidades do sistema e o frame de status. Este editor desenvolve, então, a seguinte sequência:

= apresenta o conjunto de entidades do sistema a ser simulado, e solicita do usuário a escolha de uma entidade para a qual devam ser editados seus valores iniciais. Caso seja escolhido o encerramento da edição dos valores iniciais, vai para a indicação de relacionamento entre diferentes entidades do sistema, sete passos abaixo nesta sequência.

= apresenta o conjunto de portas definidas para a entidade escolhida, e solicita do usuário a confirmação, ou alteração, do valor inicial de cada uma delas.

= apresenta o conjunto de bandeiras definidas para a entidade, e solicita do usuário a confirmação, ou alteração, do estado inicial de cada uma delas.

= apresenta o conjunto de slots definidos como "informações", e questiona se há necessidade de entrada de valores iniciais relativos a tais slots. Em caso negativo, vai para o encerramento da edição do slot "informações", três passos abaixo nesta sequência.

= solicita do usuário a indicação do sub-slot no qual deseja inserir, ou alterar, o valor inicial, e qual este valor.

= retorna à apresentação dos slots "informações", dois passos acima nesta sequência.

= encerra a edição do slot "informações". Retorna ao primeiro passo desta sequência, apresentando o conjunto de entidades do sistema a ser simulado.

= apresenta o relacionamento já estabelecido entre as portas de diferentes entidades, e entre as bandeiras de diferentes entidades, conforme definidos no frame de status.

= solicita a confirmação, ou alteração, de cada uma delas.

= encerra a edição de condições de simulação deste sistema. Solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, a armazenagem destas informações nos frames de entidades e de status do sistema.

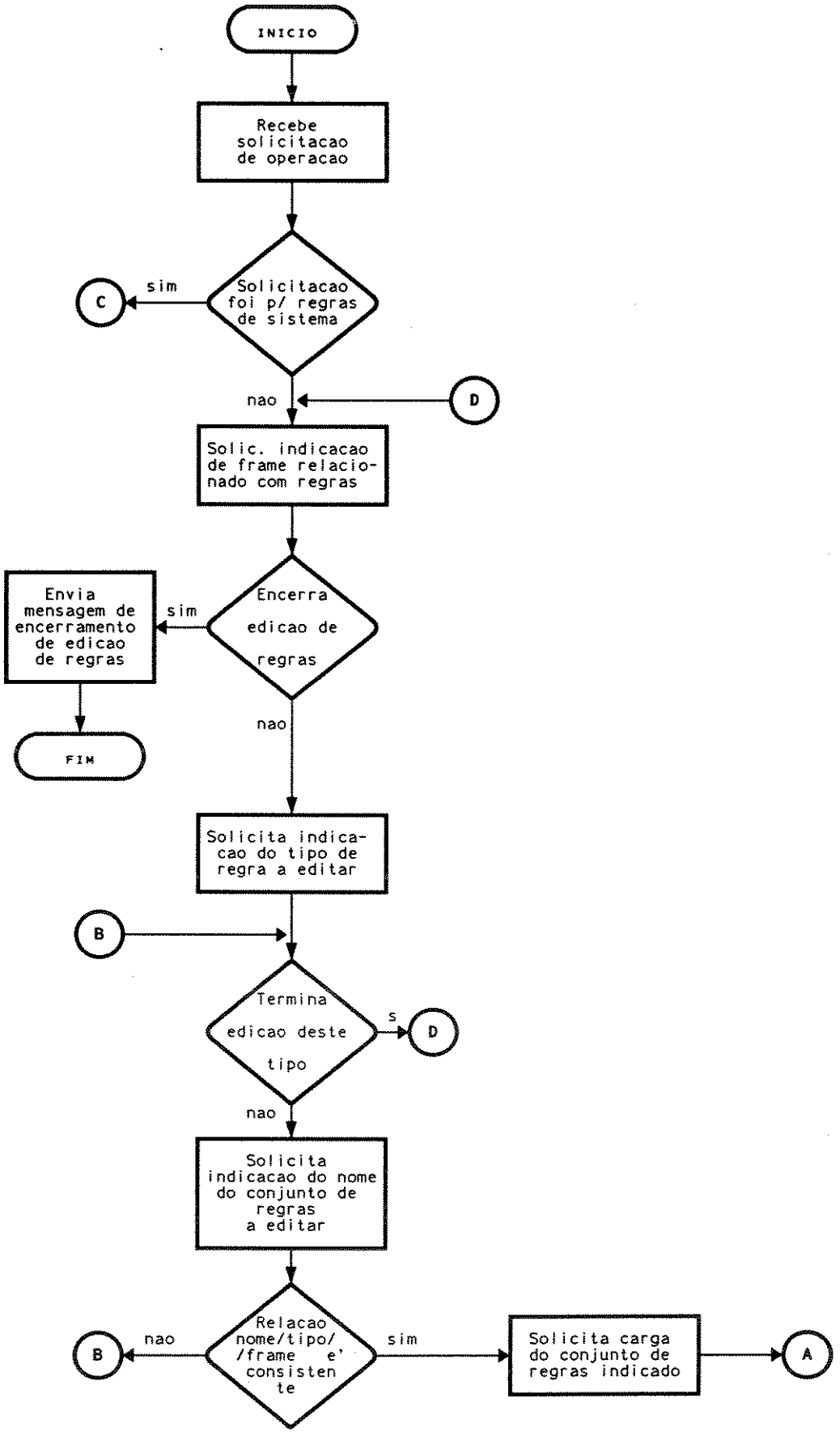
= encaminha mensagem ao Gerenciador de Aquisição de Conhecimento, indicando o encerramento de edição de valores iniciais.

b) Editor de Regras

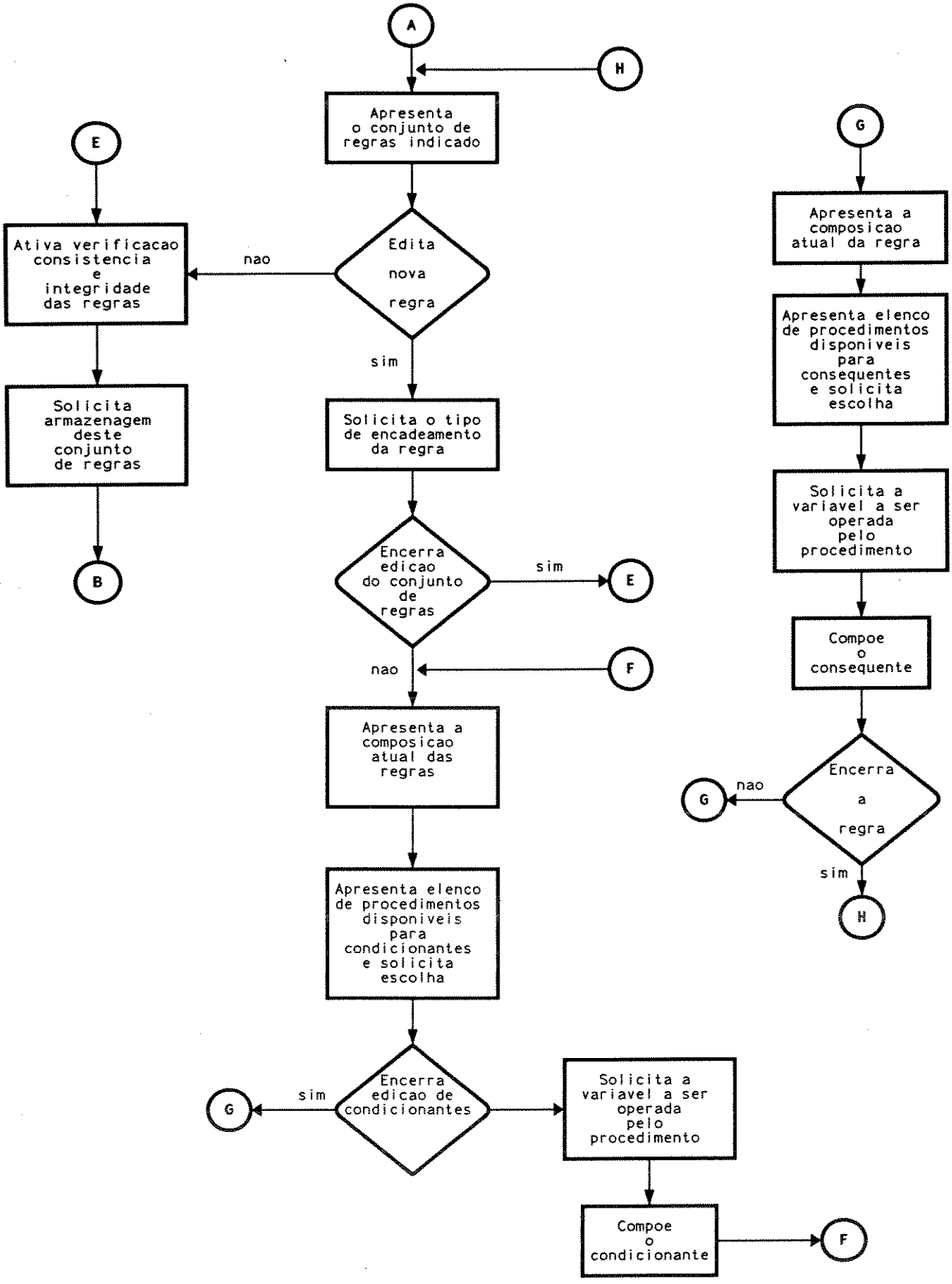
O Editor de Regras pode ser ativado devido a uma solicitação, tanto do usuário, como do Simulador. A sequência de operações do Editor de Regras é apresentada a seguir, para os diferentes tipos de regras. Nos quadros 11 a 13 representa-se estas sequências.

Uma vez ativado pelo Gerenciador de Aquisição de Conhecimento, por solicitação do usuário, este editor desenvolve os seguintes passos:

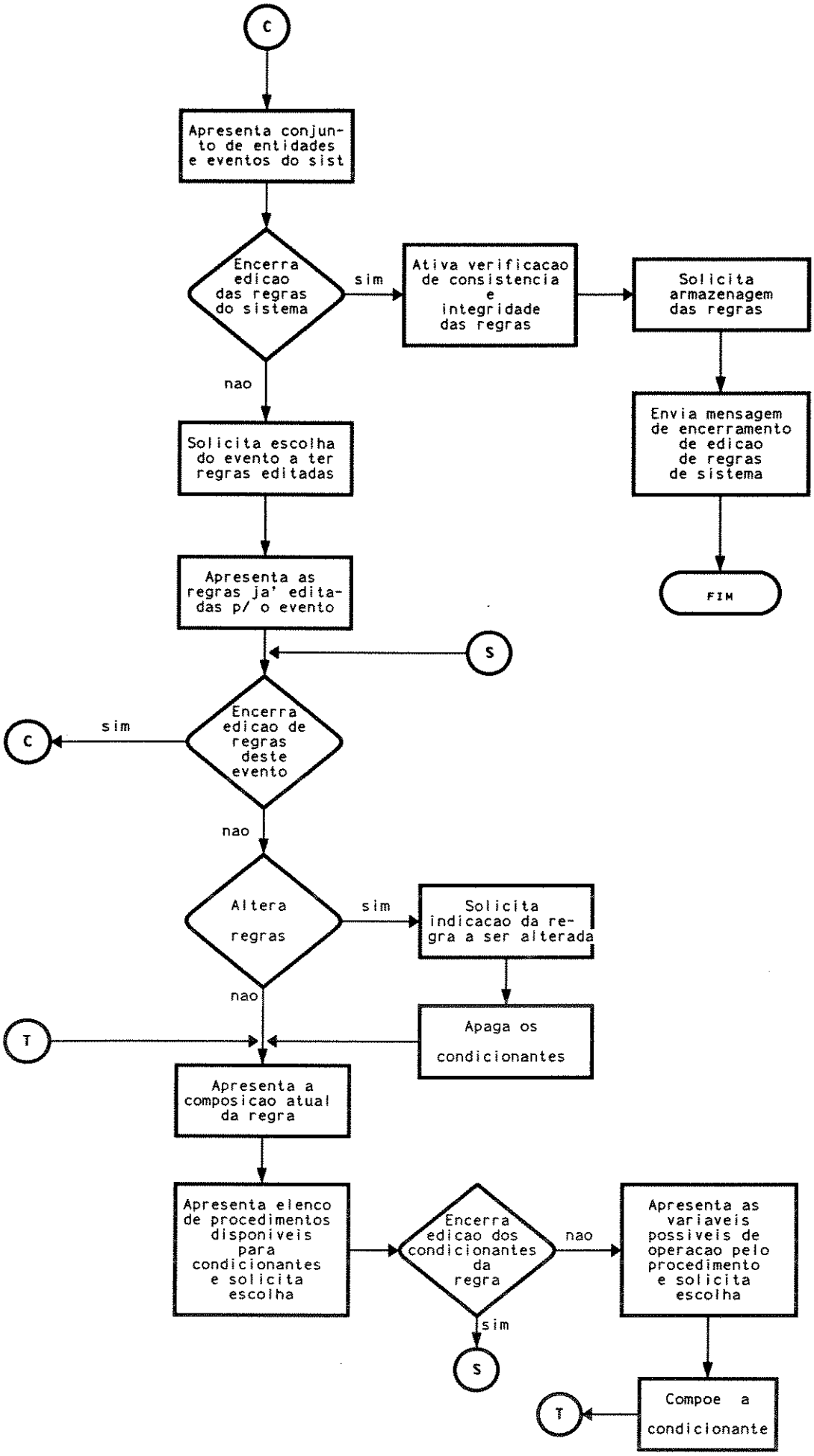
- solicita do usuário a indicação do frame ao qual se refere o conjunto de regras a ser editado, ou se deseja encerrar a edição de regras. Em caso de término de edição de regras encaminha mensagem ao Gerenciador de Aquisição de Conhecimento indicando o término de edição de regras.
- solicita do usuário a indicação do tipo de regra a ser editada (genérica de produção, de decisão de fluxo, de entidade, de sistema), ou o término da edição de regras do frame indicado.
- testa o tipo de regra escolhida pelo usuário. Para cada tipo de regra escolhida, desenvolve uma entre as sequências de operações abaixo, orientada por um tutor de composição de regras. Para o caso de término da edição de regras do frame indicado, retorna ao primeiro passo desta sequência.



Quadro 11 - Edição de Regras



Quadro 12 - Edição de Regras: regras genéricas de produção



Quadro 13 - Edição de Regras: regras de sistema

Para o caso de regras genéricas de produção tem-se:

= solicita ao usuário o nome do conjunto de regras a ser editado.

= solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, a verificação da integridade da base de regras, para o nome indicado. Caso positivo carrega, ou inicia, o conjunto de regras indicado.

= apresenta o conjunto de regras editado sob o nome indicado. Este se apresenta vazio no caso de início de edição.

= questiona a necessidade de edição de nova regra. Em caso negativo vai para o encerramento da edição de regras, 13 passos abaixo nesta sequência.

= solicita do usuário se a regra a ser editada tem encadeamento direto ou reverso. Uma opção é o encerramento da edição de regras, caso em que vai ao encerramento, 12 passos abaixo nesta sequência.

= apresenta a composição atual da regra. Em caso de início de edição, esta se apresenta vazia e preparada para a edição dos condicionantes.

= apresenta um elenco de procedimentos disponíveis para a composição dos condicionantes da regra.

= solicita que o usuário indique o procedimento escolhido.

= testa se o procedimento escolhido é o de encerramento dos condicionantes da regra, ou de término da edição de regras. Caso seja de final de edição dos condicionantes, passa para o encerramento dos condicionantes da regra, descrito três passos abaixo nesta sequência.

= solicita do usuário a entrada da variável, sobre a qual o procedimento escolhido deve atuar.

= compõe o condicionante conforme escolhido, e retorna ao sexto passo desta sequência.

= encerra a edição dos condicionantes da regra. Para o caso de não haver condicionantes, o consequente a ser editado apresenta-se como fato, e apresenta a composição atual da regra.

= apresenta um elenco de procedimentos disponíveis para a composição dos consequentes da regra.

= solicita do usuário a entrada da variável, sobre a qual o procedimento escolhido deve atuar.

= compõe o consequente conforme escolhido.

= solicita do usuário se este deseja editar nova regra. Caso afirmativo

retorna ao terceiro passo desta sequência.

= encerra a edição de regras. Ativa o verificador de composição de regras, para a avaliação de consistência e integridade das regras.

= solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, a armazenagem na base de regras, das regras editadas sob o nome indicado.

= retorna ao terceiro passo da sequência do editor de regras, questionando o término da edição do tipo de regras escolhido.

A edição de outros tipos de regras corresponde a casos particulares das regras de produção, mas pela sua importância são também aqui descritas.

Para o caso de regras de decisão de fluxo tem-se a seguinte sequência:

= solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, a carga para a memória de trabalho do slot "rotinas" do frame indicado, e das regras de decisão de fluxo da entidade.

= apresenta ao usuário o conjunto de rotinas do frame, e solicita a indicação da rotina para a qual deseja editar regras.

= testa se existe alguma indicação de necessidade de regra de decisão de fluxo (aparecimento do símbolo "/") ainda não considerada nas regras da rotina escolhida. Caso negativo vai para o encerramento das regras desta rotina, 7 passos abaixo.

= apresenta ao usuário a regra de decisão de fluxo em sua versão atual (apenas indicando o primeiro evento para o qual haja a necessidade desta decisão de fluxo no caso de ser início de edição da regra).

= apresenta um elenco de procedimentos disponíveis para a composição da regra.

= aguarda que o usuário indique o procedimento escolhido.

= testa se o procedimento escolhido é o de encerramento da regra. Caso afirmativo retorna ao terceiro passo desta sequência.

= apresenta as possíveis variáveis sobre as quais o procedimento escolhido pode atuar, e aguarda uma escolha por parte do usuário.

= compõe a regra conforme escolhido, e retorna ao terceiro passo desta sequência.

= encerra a edição de regras de decisão de fluxo para a rotina indicada.

= ativa o verificador de composição de regras, para a avaliação de consis-

tência e integridade das regras editadas.

= verifica se ainda existem rotinas para as quais devam ser editadas regras de decisão de fluxo. Caso afirmativo retorna ao segundo passo desta sequência.

= solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, a armazenagem das regras desta rotina na base de regras.

= retorna ao segundo passo da sequência do Editor de Regras, solicitando indicação de novo tipo de regra a ser editada.

Para o caso de regras de entidade tem-se:

= solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, a leitura do slot "eventos", e das regras de entidade já editadas para o frame indicado.

= apresenta ao usuário o conjunto de eventos do frame e solicita a indicação do evento para o qual devam ser editadas as regras. Uma última opção corresponde ao encerramento da edição das regras de entidade.

= testa a indicação do usuário. Caso seja encerramento da edição de regras de entidade, vai para o encerramento das regras de entidade, dez passos abaixo nesta sequência.

= apresenta ao usuário o conjunto de regras já editadas para tal evento, vazio no início da edição de regras de entidade.

= apresenta ao usuário a regra de entidade referente ao evento, no estado atual. No caso de início da edição da regra, esta se apresenta vazia quanto aos condicionantes, apenas apresentando como <ação> a indicação de execução do evento escolhido.

= apresenta um elenco de procedimentos disponíveis para a composição da regra.

= aguarda que o usuário indique o procedimento escolhido.

= testa se o procedimento escolhido é o de encerramento da regra. Caso afirmativo encerra a edição da regra, indicado três passos abaixo nesta sequência.

= apresenta as possíveis variáveis sobre as quais o procedimento escolhido pode atuar, e aguarda uma escolha por parte do usuário.

= compõe a meta da regra conforme escolhido, e retorna ao quinto passo desta sequência.

= encerra a edição da regra de entidade para o evento indicado. Para o caso de não haver condicionantes, a execução do evento é estabelecida como fato.

= solicita do usuário se este deseja editar nova regra para o mesmo evento. Caso afirmativo retorna ao quarto passo desta sequência. Caso negativo, retorna ao segundo passo desta sequência.

= encerra a edição de regras da entidade. Ativa o verificador de composição de regras, para a avaliação de consistência e integridade das regras já editadas.

= solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, a armazenagem das regras desta entidade na base de regras.

= retorna ao segundo passo da sequência do editor de regras, solicitando indicação de novo tipo de regra a ser editada.

Para o caso de regras de sistema tem-se:

= solicita ao usuário a indicação das entidades ativas no sistema, e o nome a ser dado ao conjunto de regras do sistema em edição.

= solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, a leitura do slot "eventos" de todos os frames indicados como de entidades ativas que compõem o sistema.

= apresenta ao usuário o conjunto de entidades e seus eventos, e solicita a indicação do evento para o qual devam ser editadas as regras. Uma última opção corresponde ao encerramento da edição de regras de sistema.

= testa a indicação do usuário. Caso seja encerramento da edição de regras de sistema, vai para o encerramento das regras de sistema, dez passos abaixo nesta sequência.

= apresenta ao usuário o conjunto de regras de sistema já editadas para o evento. Este conjunto apresenta-se vazio no caso de início da edição destas regras.

= apresenta ao usuário a regra de sistema referente ao evento, no estado atual. No caso de início da edição da regra, esta se apresenta vazia quanto aos condicionantes, apenas apresentando como <ação> a indicação de execução do evento e entidade escolhidos.

= apresenta um elenco de procedimentos disponíveis para a composição das condicionantes da regra.

= aguarda que o usuário indique o procedimento escolhido.

= testa se o procedimento escolhido é o de encerramento da regra. Caso afirmativo encerra a edição da regra, indicado três passos abaixo nesta sequência.

= apresenta as possíveis variáveis sobre as quais o procedimento escolhido pode atuar, e aguarda uma escolha por parte do usuário.

= compõe a condicionante da regra conforme escolhido, e retorna ao sexto passo desta sequência.

= encerra a edição da regra de entidade para o evento indicado. Para o caso de não haver condicionantes, a execução do evento é estabelecida como fato.

= solicita do usuário se este deseja editar nova regra para o mesmo evento. Caso afirmativo retorna ao quinto passo desta sequência. Caso negativo, retorna ao terceiro passo desta sequência.

= encerra a edição de regras de sistema. Ativa o verificador de composição de regras, para a avaliação de consistência e integridade das regras editadas.

= solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, a armazenagem deste conjunto de regras, na base de regras, sob o nome de sistema escolhido pelo usuário.

= retorna ao segundo passo da sequência do editor de regras, solicitando indicação de novo tipo de regra a ser editada.

Para o caso em que a edição de regras de sistema é solicitada pelo Simulador, a sequência de passos desenvolvida é a seguinte:

= apresenta ao usuário o conjunto de entidades e seus eventos, e solicita a indicação do evento para o qual devam ser editadas as regras. Uma última opção corresponde ao encerramento da edição de regras de sistema.

= testa a indicação do usuário. Caso seja encerramento da edição de regras de sistema, vai para o encerramento das regras de sistema, dez passos abaixo nesta sequência.

= apresenta ao usuário o conjunto de regras de sistema já editadas para o evento, e solicita escolha entre confirmação e edição. Em caso de confirmação retorna ao primeiro passo desta sequência.

= apresenta ao usuário a regra de sistema referente ao evento, no estado

atual. No caso de início da edição da regra, esta se apresenta vazia quanto aos condicionantes, apenas apresentando como <ação> a indicação de execução do evento e entidade escolhidos.

= apresenta um elenco de procedimentos disponíveis para a composição das condicionantes da regra.

= aguarda que o usuário indique o procedimento escolhido.

= testa se o procedimento escolhido é o de encerramento da regra. Caso afirmativo encerra a edição da regra, indicado três passos abaixo nesta sequência.

= apresenta as possíveis variáveis sobre as quais o procedimento escolhido pode atuar, e aguarda uma escolha por parte do usuário.

= compõe a condicionante da regra conforme escolhido, e retorna ao quarto passo desta sequência.

= encerra a edição da regra de entidade para o evento indicado. Para o caso de não haver condicionantes, a execução do evento é estabelecida como fato.

= solicita do usuário se este deseja editar nova regra para o mesmo evento. Caso afirmativo retorna ao terceiro passo desta sequência. Caso negativo, retorna ao primeiro passo desta sequência.

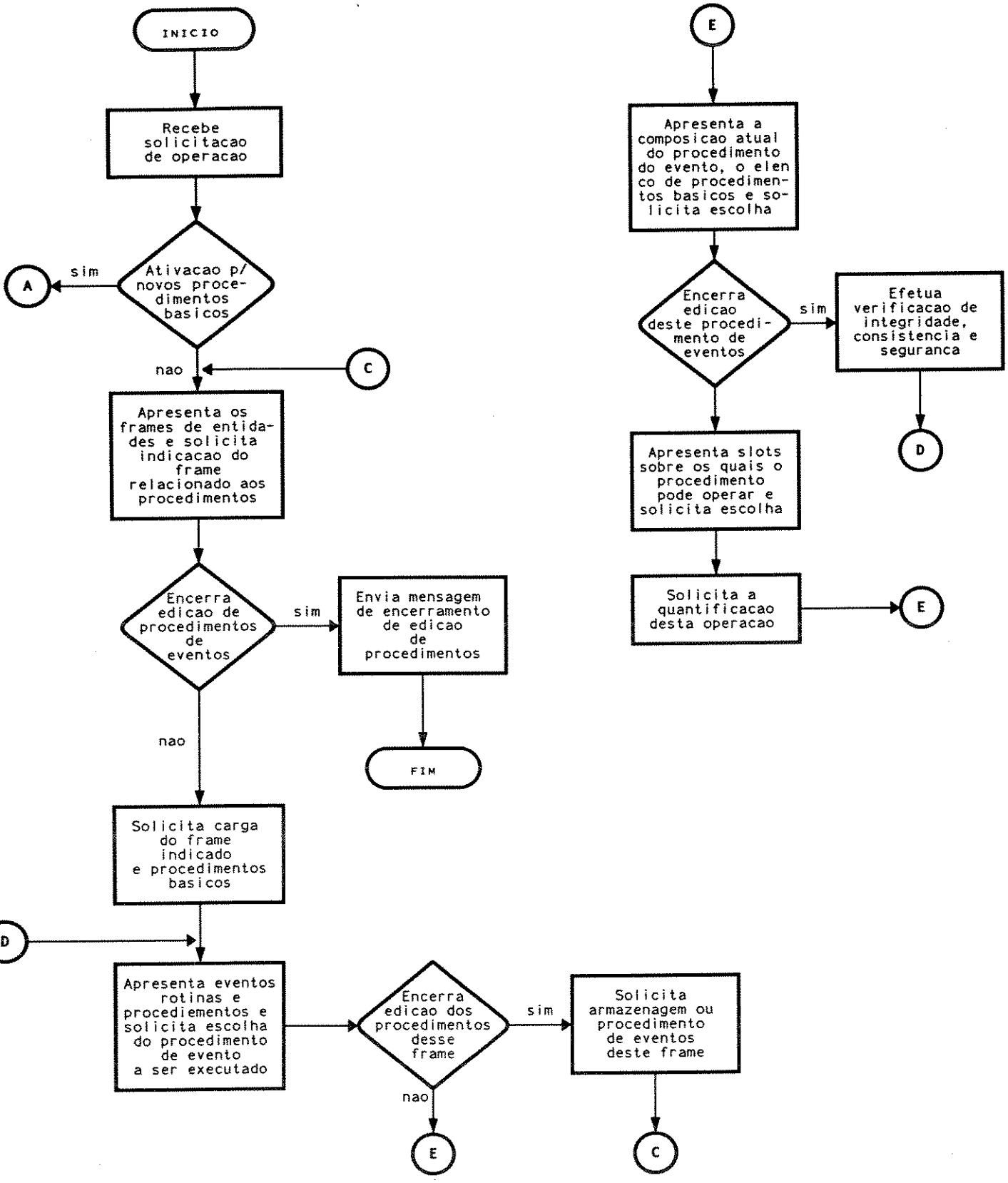
= encerra a edição de regras de sistema. Ativa o verificador de composição de regras, para a avaliação de consistência e integridade das regras editadas.

= solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, a armazenagem deste conjunto de regras, na base de regras, sob o nome de sistema escolhido pelo usuário.

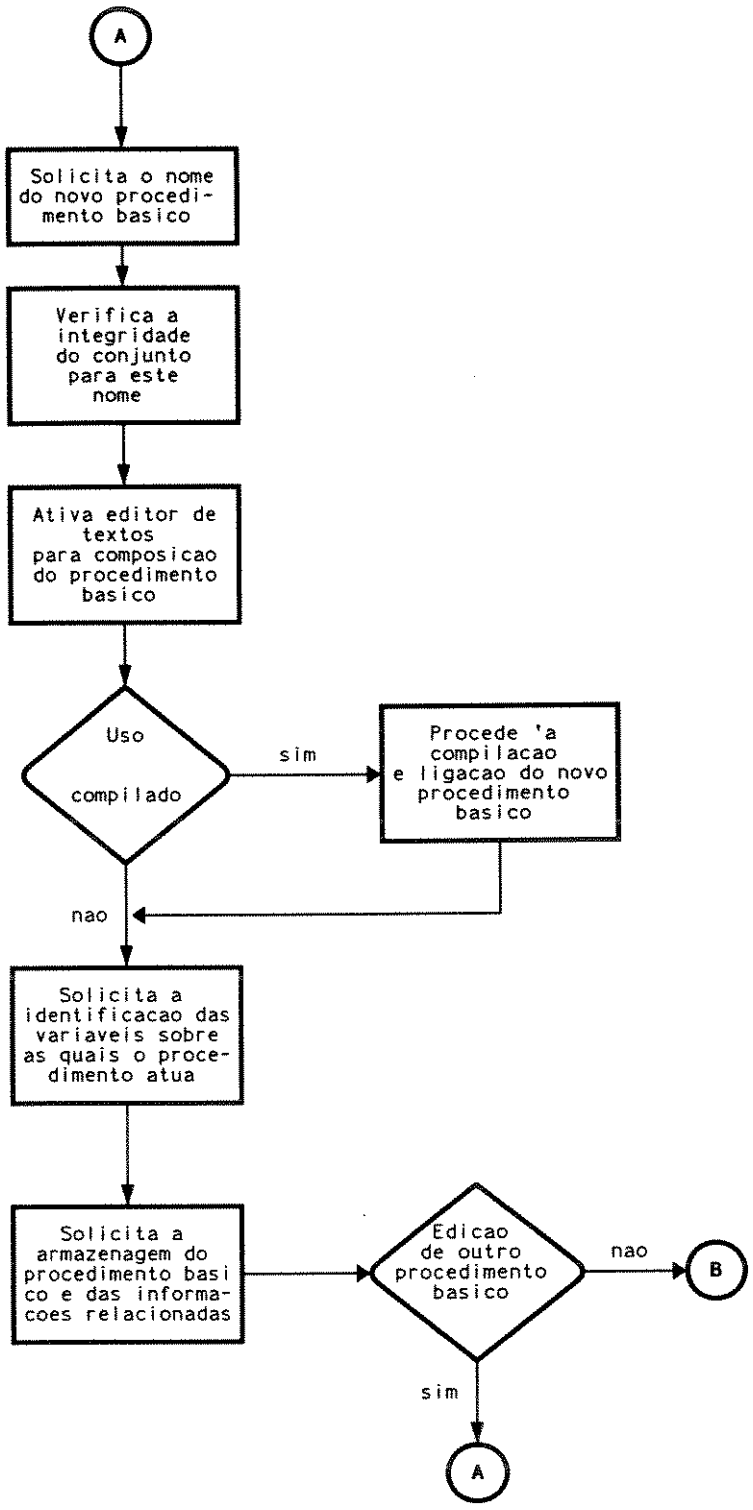
= encaminha mensagem ao Gerenciador de Aquisição de Conhecimento, indicando o término da edição de regras de sistema, conforme solicitado pelo Simulador.

c) Editor de Procedimentos

O sequenciamento de operações do Editor de Procedimentos é descrito a seguir, detalhando-se a edição de procedimentos de eventos e de novos procedimentos básicos. Este sequenciamento também é representado nos quadros 14 e 15.



Quadro 14 - Editor de Procedimentos: edição de procedimentos de eventos



Quadro 15 - Editor de Procedimentos: edição de procedimentos básicos

Uma vez ativado pelo Gerenciador de Aquisição de Conhecimento, este editor desenvolve os seguintes passos:

- solicita ao usuário a indicação do tipo de procedimento a ser gerado: se composição de novos procedimentos, ou composição de procedimentos referentes aos eventos das entidades.
- testa o tipo de procedimento escolhido pelo usuário. Para cada tipo escolhido, desenvolve uma sequência de operações. Para o caso de término de edição de procedimentos encaminha mensagem ao Gerenciador de Aquisição de Conhecimento, indicando o término de edição de procedimentos.

Os procedimentos básicos já constantes da biblioteca são usados para a edição de procedimentos vinculados aos eventos de um frame de entidade. Os novos procedimentos podem ser utilizados de forma genérica para quaisquer aplicações de inferência do usuário.

Para o caso de composição de aplicações dos procedimentos já existentes, ou seja, composição de procedimentos de eventos, tem-se a seguinte sequência:

= apresenta os frames de entidade e solicita ao usuário a indicação do frame de entidade para o qual os procedimentos serão gerados. Uma opção é o encerramento de edição de procedimentos de eventos.

= verifica a escolha realizada. Para o caso de encerramento da edição de procedimentos, encaminha mensagem ao Gerenciador de Aquisição de Conhecimento indicando o término de edição de procedimentos.

= solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento à leitura: dos eventos pertencentes ao frame da entidade indicada, das rotinas desta entidade, e dos procedimentos básicos existentes na biblioteca.

= apresenta ao usuário o conjunto dos eventos da entidade (juntamente com o nome de seus procedimentos asociados), as rotinas desta entidade, e o conjunto dos procedimentos básicos disponíveis na biblioteca de procedimentos.

= solicita do usuário o nome do procedimento de evento a ser editado. Uma opção é o encerramento de edição dos procedimentos de eventos desta entidade.

= verifica a opção do usuário. Caso a opção seja o encerramento de edição dos procedimentos de eventos desta entidade, vai para o encerramento da edição dos procedimentos de eventos da entidade, 8 passos abaixo nesta

sequência.

= apresenta a composição atual do procedimento do evento escolhido (vazio no caso de início da edição), e o elenco de procedimentos básicos disponíveis.

= solicita a escolha do procedimento básico desejado. Uma opção é encerrar a edição do procedimento do evento.

= avalia o procedimento escolhido. Caso seja o encerramento de edição do procedimento do evento vai para o encerramento deste, 4 passos abaixo nesta sequência.

= apresenta os slots sobre os quais o procedimento pode operar e solicita a escolha de um deles.

= identifica o slot a ser operado e apresenta as posibilidades de operação sobre este slot.

= uma vez escolhido o tipo de operação a ser efetuada, solicita a quantificação da operação, quando for o caso, e retorna à apresentação da composição do procedimento do evento, cinco passos acima nesta sequência.

= encerra a edição do procedimento do evento. Procede a uma verificação de integridade, consistência, e segurança referentes ao procedimento editado.

= encerra a edição dos procedimentos de eventos da entidade. Solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, que seja efetuada a armazenagem dos procedimentos relativos à entidade corrente.

= Retorna ao primeiro passo desta sequência.

Para o caso de composição de novos procedimentos básicos tem-se a seguinte sequência:

= solicita do usuário o nome a ser dado ao novo procedimento.

= verifica a integridade do conjunto de procedimentos básicos da biblioteca, considerando o novo nome a ser inserido.

= avalia a resposta do usuário, para posterior direcionamento de seu sequenciamento.

= compõe o novo procedimento básico. Isto corresponde, inicialmente, a executar uma chamada a um editor de textos, que opera sobre o arquivo de procedimentos básicos. Aqui é editado o método computacional correspondente ao procedimento. Posteriormente, quando for o caso, processa-se à compilação e ligação deste método, para composição da nova biblioteca de

procedimentos. Para procedimentos editados em Prolog, apenas inclui-se o novo procedimento no conjunto de procedimentos a serem carregados na memória de trabalho, quando da chamada da base de procedimentos.

= questiona o usuário se o novo procedimento deve ser compilado e ligado à atual biblioteca, ou se será editado em Prolog e utilizado de forma interpretada, juntamente com a base de procedimentos dos eventos.

= uma vez editado o procedimento, solicita do usuário a tipificação e o nome das variáveis sobre as quais o procedimento pode atuar. Estas variáveis são consideradas como as entradas do procedimento.

= solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, a armazenagem do conjunto atualizado de procedimentos.

CONVERSOR DE CONHECIMENTO FORMAL/DECLARATIVO

O Conversor de Conhecimento Formal/Declarativo corresponde, atualmente, em sua essência, a um tutor para a realização da transformação de conhecimento expresso de maneira formal em conhecimento expresso de modo declarativo.

Isto se deve ao fato de que, nas atuais circunstâncias, ainda não se desenvolveu uma perfeita adequação entre a GRTTL [100, 163] e a implementação dada ao modelo abstrato de informação, de forma que seja possível a realização de conversão automática de especificações geradas em GRTTL, para o modelo abstrato de informação discutido neste trabalho.

Embora já discutidos o mapeamento e a metodologia de transformação de especificações de uma forma de representação de conhecimento em outra, estes ainda se revestem de um caráter extremamente subjetivo.

Assim sendo optou-se, no escopo deste trabalho, na implementação de um tutor que apenas direcione o usuário na transformação de um modelo em outro.

O modelo formal, atualmente, só é utilizado para a descrição de entidades do tipo "planta-controle", com o propósito de realizar-se uma simulação deste sistema. Assim, a conversão de conhecimento corresponde à geração de frames, regras e procedimentos de eventos, que representem as entidades componentes da planta, descritas formalmente.

Uma vez ativado pelo Gerenciador de Aquisição de Conhecimento, este conversor desenvolve os seguintes passos:

- solicita ao usuário a informação do nome do arquivo onde se encontra a modelagem formal do sistema a ser simulado.
- solicita ao Processador de Conhecimento, através da Interface com Processador de Conhecimento, a carga, para a memória de trabalho, do arquivo escolhido.
- apresenta ao usuário o conjunto de especificações formais, e solicita que sejam digitados os nomes das entidades definidas nas especificações.
- verifica a resposta do usuário. Para cada nome de entidade cria a indicação de geração de um arquivo de frame de entidade.
- solicita do usuário a indicação de uma entidade a ter sua especificação transformada para a forma declarativa. Uma opção apresentada é concluir a transformação de especificações.
- verifica a opção do usuário. Caso a opção adotada seja concluir a conversão de especificações, envia mensagem de conclusão de conversão ao Gerenciador de Aquisição de Conhecimento.
- apresenta o frame da entidade em seu estado atual (vazio no caso de início de transformação), e o conjunto de especificações formais.
- solicita do usuário a entrada dos estados desta entidade.
- solicita a definição dos eventos que levam a entidade de um estado a outro.
- apresenta ao usuário a relação de eventos definidos, e solicita a especificação do sequenciamento imposto a estes. Para cada evento escolhido apresenta as expressões em lógica temporal que contém os símbolos dos estados vinculados a este evento.
- apresenta ao usuário a sequência de eventos, e solicita a indicação das temporalidades envolvidas com cada evento.
- solicita do usuário a especificação das portas e bandeiras a serem utilizadas no modelo declarativo.
- gera um frame com as especificações já definidas para a entidade, apresenta-o ao usuário e solicita sua confirmação.
- uma vez confirmadas as informações básicas do frame solicita ao Processador de Conhecimento, através da Interface com o Processador de Conhecimento, a armazenagem do frame da entidade na base de conhecimento.
- solicita do usuário a opção por edição de regras, ou de procedimentos, para a entidade em destaque. Uma opção corresponde ao encerramento de transformação das especificações da entidade.
- verifica opção do usuário e emite solicitação de ativação do Aquisitor de

Conhecimento Declarativo, para a edição de regras, ou de procedimentos, conforme indicação do usuário. Caso a opção seja de encerramento da transformação de especificações da entidade, retorna ao quinto passo desta sequência, para escolha de outra entidade a ter suas especificações transformadas.

- aguarda retorno da mensagem de edição de regras ou procedimentos, e retorna à opção por edição de regras ou procedimentos, dois passos acima nesta sequência.

INTERFACE COM PROCESSADOR DE CONHECIMENTO

A Interface com o Processador de Conhecimento responsabiliza-se por mapear as chamadas ao Processador de Conhecimento realizadas pelo Aquisitor de Conhecimento Formal, pelo Aquisitor de Conhecimento Declarativo, e pelo Conversor de Conhecimento Formal/Declarativo. Sua implementação depende tanto do tipo de mensagem recebida pelo Processador de Conhecimento como do tipo daquelas geradas pelos elementos componentes do Aquisitor de Conhecimentos.

Na presente implementação, este mapeamento é desnecessário, uma vez que ocorre perfeita compatibilidade entre as mensagens geradas pelos elementos do Aquisitor de Conhecimento e aquelas recebidas pelo Processador de Conhecimento.

4.3-5 - Processador de Conhecimento

O Processador de Conhecimento realiza toda a manipulação de conhecimento no ambiente. Baseia-se no Sistema de Manipulação e Armazenagem de Conhecimento - SMAC [161], desenvolvido a partir de um ambiente de controle baseado em conhecimento [6, 162]. Executa esta manipulação através de comandos de diferentes níveis hierárquicos, gerenciados pelo Supervisor de Manipulação.

Sua ativação é feita pelos diferentes elementos do Ambiente Integrado de Apoio à Tomada de Decisão, conforme as necessidade de manipulação de conhecimento.

Suas atividades são desenvolvidas através do: Supervisor de Manipulação,

Manipulador de Frames, Manipulador de Regras, e Manipulador de Procedimentos.

SUPERVISOR DE MANIPULAÇÃO

O Supervisor de Manipulação recebe solicitações de manipulação de conhecimentos vindas de diferentes interfaces. Tais chamadas são funções de alto nível, que solicitam funções especiais de manipulação de conhecimento.

Uma vez ativado por tais chamadas, o Supervisor desenvolve os seguintes passos:

- reconhece a função solicitada como uma das constantes do universo de manipulação de conhecimento.
- verifica se o elemento chamador está habilitado à função solicitada.
- verifica a qual dos manipuladores se refere a função solicitada de manipulação
- monta o comando de baixo nível para a manipulação de conhecimento de forma a ser compreendido pelo manipulador específico. Em todos os comandos, o último argumento corresponde a uma resposta gerada pelo manipulador chamado. Esta resposta pode corresponder a valores lidos de um dado em um frame, ou a variáveis lógicas do tipo "falso" ou "verdadeiro", consequência de operações de inferência ou de execução de procedimentos.
- encaminha ao manipulador específico uma mensagem composta pelo comando, e por um código indexador atribuído a este comando.
- atualiza sua diretoria de comandos de manipulação de conhecimento, incluindo uma entrada composta pelo código indexador da solicitação de manipulação, indicação do manipulador solicitado, o nome do solicitante, e o nome do comando enviado.
- aguarda o retorno da mensagem enviada ao manipulador. Uma vez recebida a resposta da mensagem de comando, monta a resposta a ser enviada à interface que solicitou a manipulação.
- dá baixa em sua diretoria de comandos de manipulação, do código indexador da solicitação de manipulação de conhecimento, já executada.

MANIPULADOR DE FRAMES

Recebe ativação através de mensagens enviadas pelo Supervisor de Manipulação. A cada mensagem corresponde uma sequência de operações referentes à

execução do comando recebido através da mensagem de ativação.

Os comandos a serem considerados dividem-se em comandos de manipulação de estrutura de frames, e de manipulação de dados em frames.

Os principais comandos de manipulação de estrutura de frames são os de definição de slots e de ramificações de slots; leitura de ramificações de slots; eliminação de slots.

A composição genérica dos comandos de manipulação de estruturas de frames toma a forma apresentada na figura 4.19.

```
comando(Frame, Instância, Slotpai, Slot)
```

Fig. 4.19 - Composição dos comandos de manipulação de estruturas de Frames

Os principais comandos de manipulação de dados em frames são os de escrita na base de frames; leitura da base de frames; eliminação de dado, atributo, ou slot na base de frames; inserção de atributo na base de frames; criação de sub-slot (ramificação de slot) na base de frames.

A composição genérica dos comandos de manipulação de dados em frames toma a forma apresentada na figura 4.20.

```
comando(Frame, Instância, Slotpai, Slot, Atrib, Struct, Resp)
```

Fig. 4.20 - Composição dos comandos de manipulação de dados em Frames

Os argumentos utilizados nos comandos de manipulação de estruturas de frames, e dados em frames correspondem a:

- "Frame" é um indicador do frame a que se refere o comando.
- "Instância" corresponde à sub-classe ou instância do frame indicado. No caso de não haver instâncias definidas, assume-se o nome do próprio frame.
- "Slotpai" corresponde ao pai do slot definido no argumento seguinte, se uma cadeia de slots já foi definida.
- "Slot" define o nome do slot sobre o qual o comando de manipulação deva operar.
- "Atrib" corresponde ao atributo sob o qual deva ocorrer a operação do coman-

do. Os atributos mais comuns são: <valor>, <regra>, <procedimento>. Atributos especiais são os de segurança, tais como:<se_ler>, <se_escrever>, <se_adicionar>, <se_remover>. A ocorrência destes atributos ativa a verificação de condições de segurança, respectivamente, quanto à leitura de um slot; escrita em um slot; geração de ramificação de um slot; remoção de um slot, atributo, ou dado.

- "Struct" corresponde ao dado propriamente dito, a ser manipulado.
- "Resp" corresponde à resposta de execução do comando, enviada ao elemento chamador. A resposta pode ser de confirmação ou negação da manipulação de dados solicitada.

Tanto os comandos de manipulação de estrutura dos frames, como os de manipulação de dados nos frames, consideram a composição estrutural de armazenagem de conhecimento adotada nas bases de frames, descrita posteriormente.

MANIPULADOR DE REGRAS

A manipulação de regras é desenvolvida através dos elementos: Máquina de Inferência Generalizada, Núcleo Dedicado de Inferência, e Montador e Analisador Sintático de Regras.

A Máquina de Inferência Generalizada é ativada por chamadas específicas para a execução de inferências sobre regras de produção com encadeamento reverso.

O Núcleo Dedicado de Inferência é ativado por chamadas específicas referentes ao tipo de implementação assumido. Nesta versão foi implementada, como exemplo, uma máquina de inferência com capacidade de encadeamento alternativo, reverso ou direto.

As mensagens para a ativação de ambas as máquinas de inferência contém uma identificação do tipo <base - identificadores - meta>.

"Base" corresponde ao nome da base de regras a ser considerada para a inferência.

"Meta" corresponde à meta passada para a máquina de inferência, a qual deverá ser provada ou rejeitada.

"Identificadores" corresponde a indicações como: qual máquina de inferência deve ser usada (generalizada ou núcleo dedicado); em qual tipo de operação deve realizar-se a inferência; qual o tipo de interfaceamento com o usuário.

Os tipos de operação da Máquina de Inferência Generalizada são: inclusão,

ajuda, e consulta. Para o Núcleo Dedicado de Inferência os tipos de operação são os mesmos acima, acrescidos da possibilidade de encadeamento reverso ou direto, para cada conjunto de regras.

O interfaceamento com o usuário pode ser de permissão, ou proibição, de sua interferência na execução de uma inferência. Esta interferência, atualmente, resume-se em acrescentar conhecimento quando não houver informações suficientes que permitam a realização de uma certa inferência. No caso da operação da Máquina de Inferência Generalizada durante simulações não se habilita a interferência do usuário.

Após cada ativação, a Máquina de Inferência Generalizada desenvolve os seguintes passos:

- registra a origem da solicitação de inferência.
- avalia a habilitação do usuário para interferência na execução da inferência. Caso este esteja habilitado, é solicitada sua atuação apenas quando certa inferência não encontrar informações, afirmativa ou negativa, a respeito de uma meta procurada.
- avalia a condição de operação da inferência. No caso de ajuda, todos os passos intermediários de uma inferência são apresentados ao usuário.
- avalia o nome da base de regras solicitada. Caso seja o mesmo da já existente na memória de trabalho apenas apaga os fatos temporários anexados em inferência anterior. Caso contrário carrega para a memória de trabalho a base de regras solicitada, através do Montador e Analisador Sintático de Regras.
- efetua, na base de regras, um conjunto de diferentes verificações recursivas, através do Montador e Analisador Sintático de Regras, tais como:
  = verifica se a meta é um "fato". Caso afirmativo retorna <verdadeiro> como resposta da inferência.
  = verifica se a meta é um "fato-temporário". Caso afirmativo retorna <verdadeiro> como resposta da inferência.
  = realiza a inversão da meta, acrescentando "nao_" ao operador da meta.
  = verifica se a nova meta é um "fato". Caso afirmativo retorna <falso> como resposta da inferência.
  = verifica se a nova meta é um "fato-temporário". Caso afirmativo retorna <falso> como resposta da inferência.
  = verifica se existe alguma regra cuja meta seja a mesma que a originalmente

solicitada na chamada da inferência. Caso afirmativo, assume como nova meta a condicionante da meta e retorna ao início das verificações, cinco passos acima nesta sequência.

= verifica se a meta é uma meta composta. Caso afirmativo separa a primeira condicionante, assume-a como nova meta e retorna ao início das verificações, seis passos acima nesta sequência.

= verifica se a meta utilizada na chamada da inferência é um procedimento definido na base de regras. Caso afirmativo, solicita ao Gerenciador de Manipulação o disparo de tal procedimento.

= questiona o usuário, se habilitado, caso a meta não tenha sido reconhecida. Retorna mensagem indicando falha no reconhecimento de tal meta, se o usuário não estiver habilitado.

- avalia o resultado das verificações acima. Retorna como resultado o valor <verdadeiro>, ou <falso>, conforme o conjunto de inferências acima tenha resultado em comprovação, ou refutação, da meta sob inferência.
- retorna ao nível superior na inferência recursiva, fornecendo informação para compor a inferência anterior.
- envia ao Gerenciador de Manipulação uma resposta à mensagem inicialmente recebida. Esta reposta corresponde a <verdadeiro>, ou <falso>, conforme a conclusão da inferência sobre a meta originalmente recebida.

O Núcleo Dedicado de Inferência, conforme implementado, desenvolve uma metodologia de controle de inferência semelhante àquela implementada para a Máquina de Inferência Generalizada.

Como diferença mais significativa considera-se a existência de inferência com encadeamento direto. Neste caso, o reconhecimento da meta é feito a partir dos condicionantes das regras, e não pelo consequente da regra como na Máquina de Inferência já descrita.

O resultado destas inferências não é mais do tipo <verdadeiro> ou <falso>, mas um conjunto de consequentes, inferido a partir da prova das condicionantes da(s) regra(s) consideradas.

O Montador e Analisador Sintático de Regras encarrega-se: da carga do arquivo referente à base de regras solicitada; de sua montagem de acordo com os parâmetros definidos para sua utilização pelas máquinas de inferência consideradas; e do reconhecimento sintático de cada termo da regra.

Quando da montagem das regras considera a estrutura de representação de conhecimento por regras de produção, préviamente definidas no ambiente.

Ao proceder ao carregamento da base de regras para a memória de trabalho, executa, inicialmente, a leitura do cabeçalho desta base. Dele reconhece: a definição do conjunto de operadores a serem agregados aos operadores básicos; e a definição coloquial descritiva utilizada para a representação de metas de regras através de procedimentos.

A definição de operadores segue o modelo de representação da linguagem Prolog, onde são especificados: a representação do operador; seu tipo, se pré-fixo, infixo, ou pós-fixo; seu nível de precedência, apresentado através de um índice relativo.

Como exemplo desta definição apresenta-se um conjunto de operadores como na figura 4.21.

operador	**tipo**	**precedência**
:	xfx	900
entao	xfx	880
se	fx	870
e	xfy	540
eh	xfx	100
'nao eh'	xfx	100
ind	xfx	50

Fig. 4.21 - Definição de operadores

Nesta definição são apresentados operadores do tipo infixo e não associativo (xfx), um operador pré-fixo não associativo (fx), e um operador infixo associativo da direita para a esquerda (xfy). Operadores não associativos são aqueles em que só pode existir um operador de mesma precedência na mesma expressão.

Os operadores de maior precedência têm seus índices relativos maiores, e são os que têm seus termos agrupados primeiro. Assim, os operadores que definem a estrutura da regra (":", "*então*", e "*se*") são os de maior precedência, já que definem o corpo da regra (":"); dentro desta quem é a <ação>, pela utilização do operador "então"; e, por fim, onde começam as condições da regra, através do operador "se".

O operador "ind" procede à separação entre a identificação numérica de uma regra e seus identificadores.

Um exemplo de como uma regra é tratada sintáticamente é apresentado na figura 4.22.

Na composição sintática apresentada informa-se que: trata-se da regra de número 05 dentro da referida base; é utilizada para inferência na decisão de fluxo da entidade torno; é considrada sob encadeamento reverso.

```
regra_05 ind fluxo_torno_rev :
            se pintura eh nova
             e peça 'nao eh' velha
             entao 'recondicionar'.
```

Fig. 4.22 - Exemplo de composição sintática de regra

O corpo da regra é separado pelo operador ":", as condições vão do operador "*se*" até o operador "*então*". O operador "*e*" indica uma associação de condições, e os operadores "*eh*" e "'*nao eh*'" reúnem termos em uma mesma condição. A <ação> correspondente a esta regra encontra-se após o operador "*então*", neste caso indicando a execução de um procedimento, ou conclusão, denominada 'recondicionar'.

A apresentação de fatos e fatos temporários, adota o mesmo conjunto de operadores e composição sintática utilizados na composição das regras. Fatos e fatos temporários são apresentados com o conjunto de identificadores e o corpo do fato, como mostra a figura 4.23. A diferenciação se encontra em que o identificador de fato se apresenta como "fato_X", enquanto para o fato temporário como "foi_dito".

```
fato_09 ind fluxo_torno :'terminar'
foi_dito ind fluxo_torno : peça eh nova
```

Fig. 4.23 - Exemplo de composição sintática de fatos e fatos temporários

Na apresentação do fato da figura, 'terminar' corresponde à execução de algum procedimento, ou conclusão, associados às decisões de fluxo da entidade "torno". A informação "peça *eh* nova" pode ter sido agregada durante uma infe-

rência realizada sobre a base de regras referente às decisões de fluxo da entidade "torno".

Também durante a carga de uma base de regras, o Montador e Analisador Sintático de Regras faz o reconhecimento da definição coloquial descritiva, utilizada para a representação de metas de regras através de procedimentos. Esta definição faz-se necessária já que o usuário pode compor cada regra utilizando-se de uma meta-linguagem coloquial.

Esta meta-linguagem permite que as metas das regras sejam compostas por procedimentos expressos na forma em que o usuário ache mais conveniente. Esta capacidade confere um considerável poder de expressão e de entendimento da composição das regras, ao poder-se utilizar de uma linguagem coloquial na composição das regras.

No entanto provê-se que tais metas sejam reconhecidos pelo ambiente, relacionando-as com os procedimentos já definidos internamente, e com as variáveis sobre as quais o usuário deseja operar, através da definição descritiva dos procedimentos.

A definição descritiva dos procedimentos é expressa através de um comando que estabelece uma relação direta entre as metas das regras, conforme expressas pelo usuário, e os procedimentos a elas associados. Este comando, denominado "executar", é apresentado uma única vez no cabeçalho da base de regras, contendo todas as definições de metas necessárias àquela base.

O comando de definição de metas para a base de regras apresenta uma estrutura composta por: um identificador da base; apresentação da meta descrita coloquialmente; e descrição do procedimento e seus argumentos, conforme passível de identificação pelo ambiente.

Um exemplo desta definição é apresentada na figura 4.24. Nela representa-se uma base de regras genérica, onde no cabeçalho é apresentado o comando "executar", com a definição das diferentes metas compostas por procedimentos, para uma dada base de regras.

O argumento "ind" apresenta a identificação da base de regras a que se referencia. "Meta" corresponde a uma variável a ser instanciada, dentre as diferentes metas definidas no corpo do comando. No caso de apenas uma meta haver sido apresentada de forma coloquial, esta já é apresentada diretamente neste argumento, no lugar da variável. "Saida" corresponde à descrição completa do procedimento relacionado com a meta, conforme contido no corpo do comando.

O corpo do comando é composto pela definição de cada meta. "MetaX" corresponde a uma meta qualquer, em sua forma coloquial, conforme constante da composição da regra. "Proc" corresponde ao procedimento interno do ambiente, a ser atribuído à meta coloquial.

```
executar(ind, Meta, Saida) -->
                (meta1 --> [proc, arg1, ..., argn],
                 meta2 --> [proc, arg1, ..., argm],
                   :                  :
                 metax --> [proc, arg1, ..., argy]
                 ; Meta=nao_atrib.
```

Fig. 4.24 - Definição de metas de regras apresentadas em forma coloquial.

Os argumentos, em número variável para cada procedimento, correspondem aos argumentos do procedimento escolhido para representar a meta. O último elemento do corpo do comando, considera que: se alguma meta foi apresentada em uma regra, e não foi reconhecida dentro do comando, esta deve ser considerada como ainda não definida. Esta consideração gera uma mensagem de erro a ser enviada ao usuário, para a tomada das providências de correção.

Toda meta de regra, que seja reconhecida como um procedimento, é automáticamente avaliada em sua sintaxe pelo Montador e Analisador Sintático de Regras, considerando a definição apresentada pelo comando "executar", constante do cabeçalho da base de regras.

Um exemplo conjunto de cabeçalho de base de regras, e regra utilizando metas definidas coloquialmente, é apresentado na figura 4.25. Nela apresenta-se uma suposta base de regras denominada XYZ. Após a definição dos operadores da base, apresenta-se as definições necessárias para as metas representadas em forma coloquial. As duas metas consideradas são 'verifica o valor porta_1', e 'verifica a bandeira XPTO'.

Para a primeira meta, procede-se à descrição de que o procedimento a ser relacionado com a meta coloquial é o "comp_valor", préviamente definido no ambiente, operando sobre a entidade "entx". Este procedimento deve avaliar se o valor atual da porta "porta_1" é igual a 5. A consequência da atuação deste procedimento é a inclusão na devida base de regras de um fato temporário. Posteriormente uma nova meta da regra avalia o conteúdo da informação anexada.

Na segunda definição de meta, procede-se à descrição de que o procedimento a ser relacionado com a meta coloquial é o "verif_flag", préviamente definido no ambiente. Este procedimento deve operar sobre a bandeira "XPTO" da entidade "entx". Este procedimento avalia se esta bandeira está em estado "set", ou "reset". A consequência de sua operação é a anexação de um fato temporário à base de regras, posteriormente avaliado por outra meta da regra.

```
Base_XYZ
      ⋮                    ⋮
   executar(XYZ,Meta,Saida) ⟶
       case([Meta='verifica o valor porta_1'
             ⟶ Saida=[comp_valor,entx,porta_1,igual,5],
             Meta='verifica a bandeira XPTO'
             ⟶ Saida=[verif_flag,entx,XPTO]
             ; Meta=nao_atrib]).
      ⋮                    ⋮

   regra_05 ind fluxo_entx_rev :
              se pintura eh nova
               e peça 'nao eh' velha
               e 'verifica o valor porta_1'
               e valor porta_1 eh 5
               e 'verifica a bandeira XPTO'
               e bandeira XPTO esta set
               entao recondicionar.
```

Fig. 4.25 - Exemplo de definição de metas de regra por procedimentos apresentados coloquialmente

MANIPULADOR DE PROCEDIMENTOS

O Manipulador de Procedimentos encarrega-se da ativação dos métodos computacionais definidos internamente ao ambiente, ou pelo usuário.

As características de todos os procedimentos definidos no ambiente, ou

neste inseridos por determinação dos usuários, são arquivadas em uma matriz de procedimentos. Esta matriz relaciona o nome do procedimento, o nome de seus argumentos, a tipificação destes argumentos, e a localização do mesmo.

Uma chamada a um procedimento é iniciada por uma solicitação enviada pelo Supervisor de Manipulação. Esta chamada provoca a procura de um método computacional definido com o nome chamado. Uma vez encontrado, procede-se à verificação dos argumentos passados na chamada. Se coincidirem, é disparado o método computacional correspondente.

Ao final da execução do método computacional referido, uma mensagem de término de execução do procedimento é enviada ao Supervisor de Manipulação.

4.3-6 - Base de Conhecimento

A Base de Conhecimento segue o modelo adotado pelo Sistema de Manipulação e Armazenagem de Conhecimento - SMAC [161], desenvolvido utilizando-se a linguagem Prolog.

A Base de Frames compõe-se dos conjuntos de estruturas de frames já definidas e de diferentes conjuntos de dados referentes a estas estruturas.

Os frames são organizados em árvores para estruturar classes, de forma a agrupar informações sobre determinado ponto de interesse. Dentro de cada frame as informações são armazenadas em slots, e classificadas quanto ao tipo através de um atributo. Os atributos podem ser: <valor>, <regra>, ou <procedimento>.

Os slots também podem ser organizados para formar uma árvore, onde a informação vai se especializando, no sentido da raiz para a folha.

A metodologia de armazenagem de informações adotada é de uma árvore balanceada, de forma a tornar mais eficiente a busca da informação, dentro da estrutura em árvore dos frames.

Dentro dos frames não são armazenados os dados, própriamente ditos, mas apenas os atributos que os especificam. Os dados são colocados de acordo com uma tabela Hush, associadas às classes, de acordo com o frame raiz da estrutura.

A separação entre estrutura do frame, e seus dados, permite uma maior eficiência na utilização da estrutura, já que a mesma pode ser aproveitada em diferentes aplicações, com uma simples mudança do conjunto de dados, e sem necessidade de repetições.

As regras são estruturadas em conjuntos distintos para diferentes aplicações. Cada conjunto recebe um identificador, que o distingue e permite seu acesso. A reunião destes conjuntos de regras compõe a chamada Base de Regras.

O cabeçalho de um conjunto de regras é composto pelo nome identificador do conjunto, pela definição dos operadores utilizados, e pela definição descritiva das metas representadas por procedimentos, conforme já discutido em item acima. O corpo do conjunto de regras é composto pelas regras encontradas sob um mesmo identificador, também já descrito em item acima.

O tipo considerado de regras são as de produção, compostas segundo os operadores especificados no cabeçalho do conjunto de regras.

A Base de Procedimentos é composta na forma de uma biblioteca de métodos computacionais. O acesso aos métodos desta biblioteca é obtido através de chamadas, onde são apresentados a identificação do método e os argumentos por ele requeridos.

CAPÍTULO 5 - APLICAÇÕES

5.1 - INTRODUÇÃO

As aplicações apresentadas a seguir foram escolhidas dentro de um elenco de testes realizados com o ambiente descrito anteriormente, valendo-se do modelo declarativo de conhecimento proposto.

O principal enfoque dos exemplos, aqui discutidos, corresponde à apresentação e descrição do uso do modelo abstrato de informação na composição de modelos de sistemas a serem simulados. A utilização destes modelos, na execução de simulações, procurou avaliar a efetividade da operação integrada e cooperativa do Simulador e do Processador de Conhecimento. Pôde-se, também, avaliar a confiabilidade da Lógica de Simulação e da Máquina de Inferência operando sobre os modelos.

Os resultados das simulações, realizadas com estes modelos, mostraram-se em confiável compatibilidade com os resultados obtidos por outros métodos. Em face da não conclusão da implementação do Analisador de Resultados, as comparações foram realizadas entre o "Trace" da simulação, obtido a partir da indicação dos eventos ocorridos e seu instante de ocorrência em cada entidade, com as planilhas de resultados de simulações emitidas pelas linguagens de simulação.

Apresenta-se, a seguir, a descrição de três diferentes exemplos de modelagens para simulação, utilizando-se das premissas apresentadas nos capítulos 3 e 4.

No primeiro exemplo, apresenta-se a modelagem de uma célula flexível de montagem de circuito impresso. Neste exemplo, o principal enfoque da modelagem é quanto aos aspectos de sincronismo, segurança das operações e análise de desempenho das operações programadas para os dois robôs que compõem a célula.

No segundo exemplo, apresenta-se a modelagem para simulação de uma rede de computadores. Esta rede é composta de estações de trabalho independentes, ligadas em uma rede de comunicação, através de um barramento. Procura-se, através da simulação, avaliar-se as características de operação da rede, em função de alguns fatores variantes como: política de gerenciamento de concessão de licença para tranmissão de mensagens na rede, volume de mensagens a serem transmitidas entre as diferentes estações de trabalho.

O terceiro exemplo corresponde à modelagem de uma célula flexível de processamento, composta por duas máquinas distintas. A modelagem desta célula é realizada, primeiramente, através de um conjunto de especificações em Lógica Temporal de Tempo Real Generalizada (GRTTL), tanto para a planta, como para seu controlador. A seguir desenvolve-se uma transformação destas especificações para o modelo declarativo de informação adotado no ambiente. Os resultados da simulação realizada leva a uma constatação da confiabilidade da metodologia de transformação entre especificações, proposta no capítulo 4.

5.2 - CÉLULA FLEXÍVEL DE MONTAGEM DE CIRCUITO IMPRESSO

Este exemplo corresponde à modelagem de uma célula flexível de montagem de circuito impresso, operada por dois tipos de robôs. A configuração da célula aqui modelada segue um exemplo tradicional [91, 96, 97, 98], para o estudo de sincronismo, segurança e análise de desempenho de células flexíveis de montagem.

a) Descrição da célula

A tarefa da célula considerada é de preencher um cartão de circuito impresso com componentes eletrônicos, que, por fim, é soldado em uma máquina de solda por onda. O planejamento proposto é de que dois robôs operem de forma concorrente, para operações de manipulação das diferentes partes envolvidas.

Um dos robôs procede ao preenchimento do cartão com os chips, enquanto o outro encarrega-se de manipulação do cartão, além de proceder à soldagem deste na máquina de solda por onda.

Os chips são providos através de alimentadores gravitacionais. Dois buffers, um fixador e a máquina de solda por onda, completam a célula. O primeiro dos buffers é utilizado para armazenagem de cartões virgens, sem componentes, enquanto o segundo é utilizado para armazenagem dos cartões prontos. O fixador é utilizado para manter o cartão de circuito impresso em posição correta para a inserção dos componentes. A máquina de solda por onda é utilizada uma única vez por cartão, apenas após a completa montagem do cartão.

A concorrência de operações entre os dois robôs evidencia-se após a montagem do primeiro cartão, quando são desenvolvidas operações em paralelo.

Um dos principais pontos de análise, desejados neste exemplo, corresponde à análise de segurança, já que ambos os robôs partilham uma área comum de trabalho. Outro ponto a ser analisado corresponde ao acompanhamento temporal das tarefas desenvolvidas por ambos, de forma a avaliar a eficiência da metodologia de trabalho proposta.

Para efeito de distinção entre os componentes da célula denomina-se de "robô 1" ao robô montador de componentes no cartão virgem. O robô manipulador de cartões, que também procede à soldagem, é denominado de "robô 2".

O buffer de cartões virgens é denominado de "buffer 1" e o de cartões prontos de "buffer 2". Em ambos os buffers procede-se apenas ao controle da quantidade de cartões virgens, e prontos, respectivamente, em função do tempo.

Em função da análise pretendida para esta simulação, não são modelados os alimentadores gravitacionais. Apenas considera-se um certo tempo de trabalho do robô 1, para a completa montagem dos cartões

O plano de trabalho proposto para a célula de montagem é o seguinte:

ROBÔ 1 (montador)

1 - aguarda um sinal de liberação do espaço comum de trabalho, proveniente do robô 2;
2 - inicia a colocação dos componentes no cartão virgem, já posicionado no fixador, após o recebimento do sinal de liberação, emitido pelo outro robô;
3 - move-se para posição de segurança, após a conclusão do trabalho de montagem do cartão;
4 - sinaliza para o robô 2, indicando a liberação do espaço comum de trabalho;
5 - retorna ao primeiro passo desta sequência.

ROBÔ 2 (manipulador)

1 - apanha um cartão virgem, no buffer de cartões virgens (buffer 1);
2 - coloca o cartão no fixador;
3 - move-se para posição de segurança;
4 - sinaliza para o robô 1 a liberação do espaço comum de trabalho;
5 - aguarda sinal de liberação do espaço comum de trabalho, proveniente do robô 1;

6 - retira do fixador o cartão montado e o coloca, temporáriamente, no buffer de cartões prontos (buffer 2);
7 - apanha novo cartão no buffer de cartões virgens (buffer 1);
8 - coloca este cartão no fixador;
9 - move-se para posição de segurança;
10- sinaliza para o robô 1 a liberação do espaço comum de trabalho;
11- apanha, do buffer de cartões prontos (buffer 2), um cartão já montado;
12- procede à soldagem do cartão montado;
13- coloca o cartão pronto no buffer de cartões prontos (buffer 2);
14- vai para posição de segurança, sinaliza para o outro robô a liberação do espaço comum de trabalho e retorna ao quinto passo desta sequência.

Para efeito de composição do plano de trabalho das entidades, foram definidos os eventos apresentados a seguir, representando o comportamento de cada uma destas entidades. Também expressa-se as condicionantes à execução dos eventos e a descrição das operações realizadas em cada situação.

Robô 1:
1- Início do movimento 1 - robô vai da posição de descanso para o alimentador. Sinaliza sua entrada no espaço comum de trabalho, "resetando" as bandeiras sinalizadoras deste espaço.
2- Fim do movimento 1 - término do movimento acima.
3- Início da colocação dos componentes no cartão.
4- Fim dessa colocação de componentes.
5- Início do movimento 2 - robô vai do fixador, onde estava colocando os componentes, para a posição de segurança.
6- Fim do movimento acima - Emite sinal de liberação de espaço para o outro robô, "setando" sua bandeira sinalizadora de espaço comum de trabalho livre. O próximo evento a ocorrer é o de número 1.

Robô 2:
1- Início do movimento 1 - robô vai da posição de descanso para o buffer de cartões virgens. Sinaliza sua entrada na área de trabalho, "resetando" as bandeiras sinalizadoras de espaço comum livre.

2- Fim do movimento acima.

3- Início do movimento 2 - robô apanha o cartão no buffer de cartões virgens e o leva para o fixador.

4- Fim do movimento acima.

5- Início do movimento 3 - robô vai do fixador para a posição de segurança.

6- Fim do movimento acima - emite sinal de liberação do espaço de trabalho para o outro robô, "setando" sua bandeira sinalizadora de espaço comum livre. O próximo evento é o de número 7 ou o 13, dependendo do estado da bandeira sinalizadora de existência de cartão semipronto.

7- Início do movimento 4, se a bandeira sinalizadora de existência de cartão semipronto estiver no estado "reset" - vai da posição de segurança para o fixador, e apanha o cartão preenchido. Também "reseta" as bandeiras sinalizadoras de espaço comum livre

8- Fim do movimento acima.

9- Início do movimento 5 - vai do fixador para o buffer de cartões prontos para aí deixá-lo temporáriamente.

10-Fim do movimento acima. Sinaliza que deixou o cartão temporáriamente, "setando" a bandeira sinalizadora de existência de cartão semipronto.

11-Início do movimento 6 - vai do buffer de cartões prontos para o de cartões virgens.

12-Fim do movimento acima. O próximo evento é o de número 3.

13-Início do movimento 7, se a bandeira sinalizadora de existência de cartão semipronto estiver no estado "set" - vai da posição de descanso para o buffer de cartões prontos e apanha o cartão lá deixado temporáriamente. "Reseta" as bandeiras sinalizadoras de espaço comum livre.

14-Fim do movimento acima - sinaliza que retirou o cartão semipronto, "resetando" a bandeira sinalizadora de existência de cartão semi-pronto.

15-Início do movimento 8 - vai do buffer de cartões prontos para a máquina de solda.

16-Fim do movimento acima.

17-Início do processo de soldagem.

18-Fim do processo de soldagem.

19-Início do movimento 9 - vai da máquina de solda para o buffer de cartões prontos e aí coloca o cartão, definitivamente.

20-Fim do movimento acima - sinaliza que completou mais um cartão.

21-Início do movimento 10 - vai do buffer de cartões prontos para a posição de

segurança.

22-Fim do movimento acima. Após este movimento o próximo evento a ocorrer é o de número 7. "Seta" sua bandeira sinalizadora de espaço comum livre.

A partir do reconhecimento dos eventos, feito acima, pode-se efetuar a modelagem da célula, conforme descrito a seguir.

B - MODELAGEM DA CÉLULA

A modelagem (frames, regras e procedimentos) das entidades que compõem a célula de montagem, denominada celula1, é apresentada a seguir.

B1 - ENTIDADES

Robô 1

```
nome       : <valor> ⟶ robo1
versão     : <valor> ⟶ 1
data_hora  : <valor> ⟶ 01-12-89
tipo       : <valor> ⟶ celula1
membros    : <valor> ⟶ []
portas     : <valor> ⟶ []
bandeiras :
  elivre2 : <valor> ⟶ [reset]
  slivre1 : <valor> ⟶ [reset]
informações : <valor> ⟶ []
eventos    :
   inic_mov1    : <valor> ⟶ [1,inic_mov1,desc,alim,des,
                             2,0,_,_,T]
   fim_mov1     : <valor> ⟶ [2,fim_mov1,desc,alim,des,
                             0,0,_,_,T]
   inic_proces1: <valor> ⟶ [3,inic_proces1,_,_,des,
                             15,0,_,_,T]
   fim_proces1 : <valor> ⟶ [4,fim_proces1,_,_,des,
                             0,0,_,_,T]
   inic_mov2    : <valor> ⟶ [5,inic_mov2,fix,desc,des,
                             2,0,_,_,T]
   fim_mov2     : <valor> ⟶ [6,fim_mov2,fix,desc,des,
                             0,0,_,_,T]
rotinas :
   rot1 :
     sequência : <valor> ⟶ [1>2>3>4>5>6>/]
     decisão_fluxo : <regra> ⟶ [rrob1]
     regras_entidade : <regra> ⟶ [brobo1]
   evento_rotina : <valor> ⟶ [_]
```

As regras referentes ao robô 1 são apresentadas a seguir:

- regras de entidade (referente aos eventos)

brobo1

```
operadores ([]).
   executar(brobo1,Meta,X) :-
         case([Meta = 'verifica a flag elivre2'
             --> X = [verif_flag,robo1,elivre2]
             ; X = nao_atrib]).

regra_1 ind brobo1_robo1_rev :
             se 'verifica a flag elivre2',
             e  elivre2 esta set,
             entao
                  [executar,inic_mov1,1,robo1,rot1].
```

Esta regra indica que, se a bandeira "elivre2" estiver no estado set, então pode ser executado o evento "inic_mov1", de número 1, na rotina 1.

- regras de decisão de fluxo

rrobo1

```
operadores ([]).
   executar(rrob1,Meta,X) :-
         case([Meta = 'escreve evento 1 - rot1'
             --> X = [escreve,robo1,rot1,1]
             ; X = nao_atrib]).

regra_1 ind rrob1_robo1_rev :
             se 'escreve evento 1 - rot1',
             entao
                  [decisao,fim_mov2,6,robo1,rot1].
```

Esta regra indica que, uma vez ocorrido o evento "fim_mov2", o próximo evento a ocorrer é o de número 1, da rotina 1. A existência desta regra é mais por haver sido encerrada a representação da sequência na rotina, do que por necesidade de decisão de fluxo, propriamente dita.

Os procedimentos de eventos referentes à entidade robô 1 são apresentados a seguir.

Proced_robo1

```
inic_mov1([robo1,rot1]) :- bandeira(robo1,elivre2,reset).
                           bandeira(robo1,slivre1,reset).
```

Este procedimento indica que, ao ocorrer o evento "inic_mov1", as bandeiras "elivre2" e "slivre1" têm seus estados levados a "reset".

```
fim_mov2([robo1,rot1]) :- bandeira(robo1,slivre1,set).
```

Este procedimento indica que, ao ocorrer o evento "fim_mov2", a bandeira "slivre1" tem seu estado levado a "set".

Robô 2

```
nome       : <valor> --> robo2
versão     : <valor> --> 1
data_hora  : <valor> --> 01-12-89
tipo       : <valor> --> celula1
membros    : <valor> --> []
portas     :
    ebuf1  : <valor> --> [num,1,1,1000,_,_]
    sbuf2  : <valor> --> [num,1,1,0,_,_]
bandeiras  :
  slivre2  : <valor> --> [reset]
  elivre1  : <valor> --> [reset]
  stemp    : <valor> --> [reset]
informações : <valor> --> []
eventos    :
   inic_mov1   : <valor> --> [1,inic_mov1,desc,buf1,des,
                              10,0,_,_,T]
   fim_mov1    : <valor> --> [2,fim_mov1,desc,buf1,des,
                              0,0,_,_,T]
   inic_mov2   : <valor> --> [3,inic_mov2,buf1,fix,des,
                              7,0,_,_,T]
   fim_mov2    : <valor> --> [4,fim_mov2,buf1,fix,des,
                              0,0,_,_,T]
   inic_mov3   : <valor> --> [5,inic_mov3,fix,desc,des,
                              5,0,_,_,T]
   fim_mov3    : <valor> --> [6,fim_mov3,fix,desc,des,
                              0,0,_,_,T]
   inic_mov4   : <valor> --> [7,inic_mov1,desc,fix,des,
                              7,0,_,_,T]
   fim_mov4    : <valor> --> [8,fim_mov1,desc,fix,des,
                              0,0,_,_,T]
   inic_mov5   : <valor> --> [9,inic_mov1,fix,buf2,des,
                              8,0,_,_,T]
   fim_mov5    : <valor> --> [10,fim_mov1,fix,buf2,des,
                              0,0,_,_,T]
   inic_mov6   : <valor> --> [11,inic_mov1,buf2,buf1,des,
                              6,0,_,_,T]
   fim_mov6    : <valor> --> [12,fim_mov1,buf2,buf1,des,
                              0,0,_,_,T]
   inic_mov7   : <valor> --> [13,inic_mov1,desc,buf2,des,
```

```
                                    10,0,_,_,T]
      fim_mov7     : <valor> ⟶ [14,fim_mov1,desc,buf2,des,
                                    0,0,_,_,T]
      inic_mov8    : <valor> ⟶ [15,inic_mov1,buf2,sold,des,
                                    10,0,_,_,T]
      fim_mov8     : <valor> ⟶ [16,fim_mov1,buf2,sold,des,
                                    0,0,_,_,T]
      inic_solda   : <valor> ⟶ [17,inic_solda,_,_,des,
                                    25,0,_,_,T]
      fim_solda    : <valor> ⟶ [18,fim_solda,_,_,des,
                                    0,0,_,_,T]
      inic_mov9    : <valor> ⟶ [19,inic_mov9,sold,buf2,des,
                                    8,0,_,_,T]
      fim_mov9     : <valor> ⟶ [20,fim_mov9,sold,buf2,des,
                                    0,0,_,_,T]
      inic_mov10   : <valor> ⟶ [21,inic_mov10,buf2,desc,des,
                                    8,0,_,_,T]
      fim_mov10    : <valor> ⟶ [22,fim_mov10,buf2,desc,des,
                                    0,0,_,_,T]

   rotinas :
      rot1 :
        sequência : <valor> ⟶ [1>2>3>4>5>6>/>7>8>9>10>11>
                                12>/>13>14>15>16>17>18>
                                19>20>21>22>/]
        decisão_fluxo : <regra> ⟶ [rrob2]
        regras_entidade : <regra> ⟶ [brobo2]
      evento_rotina : <valor> ⟶ [_]
```

As regras referentes ao robô 2 são apresentadas a seguir:

- regras de entidade (referente aos eventos)

```
   brobo2

operadores ([]).

executar(brobo2,Meta,X) :-
         case([Meta = 'verifica a flag elivre1'
             ⟶ X = [verif_flag,robo2,elivre1]
             ; X = nao_atrib]).


regra_1 ind brobo2_robo2_rev :
             se 'verifica a flag elivre1',
             e  elivre1 esta set,
             entao
                  [executar,inic_mov4,7,robo2,rot1].
```

Esta regra indica que o evento "inicio_mov4" só pode ocorrer se a bandeira "elivre1" estiver no estado "set".

```
regra_2 ind brobo2_robo2_rev :
            se 'verifica a flag elivre1',
            e  elivre1 esta set,
            entao
                 [executar,inic_mov7,13,robo2,rot1].
```

Esta regra indica que o evento "inicio_mov7" só pode ocorrer se a bandeira "elivre1" estiver no estado "set".

- regras de decisão de fluxo

rrobo2

```
operadores ([]).
executar(rrob2,Meta,X) :-
          case([Meta = 'escreve evento 3 - rot1'
              --> X = [escreve,robo2,rot1,3],
                Meta = 'escreve evento 7 - rot1'
              --> X = [escreve,robo2,rot1,7],
                Meta = 'escreve evento 13 - rot1'
              --> X = [escreve,robo2,rot1,13],
                Meta = 'verifica a flag stemp'
              --> X = [verif_flag,robo2,stemp]
              ; X = nao_atrib]).
```

```
regra_1 ind rrob2_robo2_rev :
            se 'verifica a flag stemp'
            e  stemp esta reset
            e  'escreve evento 7 - rot1',
            entao
                 [decisao,fim_mov3,6,robo2,rot1].

regra_2 ind rrob2_robo2_rev :
            se 'verifica a flag stemp'
            e  stemp esta set
            e  'escreve evento 13 - rot1',
            entao
                 [decisao,fim_mov3,6,robo2,rot1].
```

Estas duas regras são utilizadas para a decisão de fluxo, após a ocorrência do evento "fim_mov3". Dependendo do estado da bandeira "stemp", o próximo evento é o de número 7 ou 13, ambos na rotina 1.

```
regra_3 ind rrob2_robo2_rev :
            se 'escreve evento 3 - rot1'
            entao
                 [decisao,fim_mov6,12,robo2,rot1].
```

Esta regra indica que, após a ocorrência do evento "fim_mov6", o evento a ocorrer é o de número 3, da rotina 1. Esta decisão, embora incondicional, se torna necessária, devido ao fato de o evento de número 7 corresponder ao início de uma nova subsequência. Com isto, evita-se uma repetição da apresentação de um evento na descrição da sequência.

```
regra_4 ind rrob2_robo2_rev :
            se 'escreve evento 7 - rot1'
            entao
                 [decisao,fim_mov10,22,robo2,rot1].
```

Esta regra indica que, após a ocorrência do evento "fim_mov10", o evento a ocorrer é o de número 7, da rotina 1. Esta decisão se tornou necessária devido ao encerramento da representação da sequência.

Os procedimentos de eventos, referentes à entidade robô 2, são apresentados a seguir.

Proced_robo2

```
fim_mov1([robo2,rot1]) :-
          bandeira(robo2,slivre2,reset),
          bandeira(robo2,elivre1,reset),
          altera_porta(robo2,ebuf1,num,[-]).
```

Este procedimento indica que, ao ocorrer o evento "fim_mov1", as bandeiras sinalizadoras de espaço comum livre são levadas ao estado "reset" e que a porta "ebuf1" tem seu valor decrescido de um lote, de acordo com a definição da porta.

```
fim_mov3([robo2,rot1]) :-
          bandeira(robo2,slivre2,set).
```

Este procedimento indica que, ao ocorrer o evento "fim_mov3", a bandeira "slivre2" tem seu estado levado a "set".

```
inic_mov4([robo2,rot1]) :-
          bandeira(robo2,slivre2,reset),
          bandeira(robo2,elivre1,reset).
```

Este procedimento indica que, ao ocorrer o evento "inic_mov4", as bandeiras sinalizadoras de espaço comum livre são levadas ao estado "reset".

```
fim_mov5([robo2,rot1]) :-
        bandeira(robo2,stemp,set).
```

Este procedimento indica que, ao ocorrer o evento "fim_mov5", a bandeira "stemp" tem seu estado levado a "set".

```
inic_mov7([robo2,rot1]) :-
        bandeira(robo2,slivre2,reset),
        bandeira(robo2,elivre1,reset).
```

Este procedimento indica que, ao ocorrer o evento "inic_mov7", as bandeiras sinalizadoras de espaço comum livre são levadas ao estado "reset".

```
fim_mov7([robo2,rot1]) :-
        bandeira(robo2,stemp,reset).
```

Este procedimento indica que, ao ocorrer o evento "fim_mov7", a bandeira "stemp" tem seu estado levado a "reset".

```
fim_mov9([robo2,rot1]) :-
        altera_porta(robo2,sbuf2,num,[+]).
```

Este procedimento indica que, ao ocorrer o evento "fim_mov9", a porta "sbuf2" tem seu valor acrescido de um lote, de acordo com a definição da porta.

```
fim_mov10([robo2,rot1]) :-
        bandeira(robo2,slivre2,set).
```

Este procedimento indica que, ao ocorrer o evento "fim_mov10", a bandeira "slivre2" tem seu estado levado a "set".

B1.A - Comentários à modelagem das entidades

O robô 2 foi modelado, procurando-se otimizar a definição dos eventos. Isto pode ser observado em relação à sequência entre os eventos "inic_mov2" "fim_mov3". Este conjunto de quatro eventos é repetido em duas distintas si-

tuações: no início das operações, quando nenhum cartão ainda foi montado; na sequência normal de operações, quando há cartões esperando para serem soldados.

Na modelagem apresentada acima, não se repetiu esta sequência, mas precisou-se da definição de uma bandeira de controle "stemp", para verificação da existência de cartões temporáriamente colocados no buffer de cartões prontos, à espera de serem soldados. O estado desta bandeira é verificado logo após a ocorrência do evento 6, em regras de decisão de fluxo.

Uma alteração possível, nesta modelagem, seria a repetição da sequência de eventos, no trecho do evento "inic_mov2" até o evento "fim_mov3", com outros nomes e identificadores. Isto simplificaria a apresentação da sequência da rotina, eliminando decisões de fluxo e também eliminando as regras de decisão de fluxo correspondentes.

A decisão de qual dentre os modelos deva ser adotado depende dos interesses do usuário modelador.

B2 - SISTEMA

Uma representação gráfica do sistema desejado é apresentada na figura 5.1 abaixo. Esta configuração indica os canais de comunicação entre as entidades, representados pelas bandeiras, e utilizados para controle de operações das entidades. A bandeira "stemp" da entidade "robo2" é utilizada apenas para controle próprio da entidade. As portas "ebuf1" e "sbuf2" representam os buffers da célula e podem ser utilizados para controle dos cartões virgens e prontos, respectivamente.

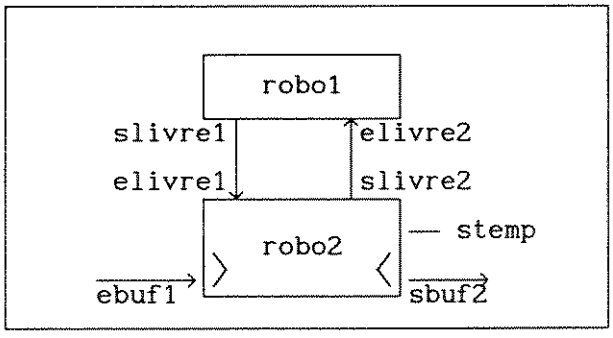


Fig. 5.1 - Representação gráfica do sistema Célula1

O frame, as regras e os procedimentos, que formam a composição do sistema a ser simulado, são apresentados a seguir.

STATUS

```
nome       : <valor> → status
versão     : <valor> → 1
data_hora  : <valor> → 01-12-89
tipo       : <valor> → celula1
membros    : <valor> → []
entidades  :
           robo1 : <valor> → []
           robo2 : <valor> → []
matriz_ligação      :
     portas    : <valor> → []
     bandeiras : <valor> → [slivre1_robo1=elivre1_robo2,
                            slivre2_robo2=elivre2_robo1]
condições_iniciais  :
     portas    : <valor> → [ebuf1_robo2=1000,
                            sbuf2_robo2=0]
     bandeiras : <valor> → [slivre1_robo1=reset,
                            slivre2_robo2=reset,
                            elivre1_robo2=reset,
                            elivre2_robo1=reset]
condições_sistema   : <regra> → [bcelula1]
condições_término   : <regra> → [bcel1trm]
clock               : <valor> → [0]
```

As regras referentes ao sistema não são necessárias neste exemplo. Isto ocorre, devido a que todo o controle de operação entre entidades da célula foi efetuado através das bandeiras de controle.

Neste caso, na base "bcelula1", tem-se um conjunto de fatos, ao invés de um conjunto de regras. Isto faz com que a ocorrência dos eventos seja inquestionável, após seu escalonamento. Para cada evento, de cada uma das duas entidades, apresenta-se um fato como o abaixo, referente ao evento "inic_mov1" da entidade "robo1". Este evento corresponde ao evento de número 1, referente à rotina 1.

```
fato_1 ind bcelula1_robo1 : [executar,inic_mov1,1,robo1,rot1]
```

As condições de término são apresentadas pelo conjunto de regras denominado "bcel1trm", apresentado abaixo.

bcell1trm

```
operadores ([]).

executar(sys,Meta,X) :-
          case([Meta = 'avalia a porta ebuf1_robo2'
              → X = [verif_valor,ebuf1,robo2,igual,0]
              ; X = nao_atrib]).


regra_1 ind bcell1trm_sys_rev :
              se 'avalia a porta ebuf1_robo2'
              e  ebuf1 eh igual_a_0,
              entao 'terminar'.
```

Esta regra indica que a condição de terminação da simulação corresponde ao esvaziamento da porta "ebuf1", onde se considera armazenados os cartões virgens.

5.3 - REDE DE COMPUTADORES

Este exemplo, mais complexo que o anterior, procura salientar tanto a versatilidade como a flexibilidade da possibilidade de modelagem, utilizando-se o modelo declarativo de informação proposto.

O exemplo corresponde à modelagem de uma rede local de computadores em ambiente industrial, muito parecido com o apresentado em [99]. Esta modelagem destina-se à realização de simulações, com o intuito de serem avaliadas algumas características de comportamento de estações de trabalho, denominadas "End Systems", frente a certa metodologia de gerenciamento da rede e a diferentes volumes de mensagens a serem transmitidas entre os "End Systems".

a) DESCRIÇÃO DA REDE

A rede a ser modelada é composta por computadores (ES) interconectados por um barramento, com operação de transmissão gerenciada por um único módulo de gerenciamento da rede (LAN). A configuração do modelo da rede em consideração é mostrada na figura 5.2 abaixo, para um protocolo de acesso ao meio de transmissão do tipo "TOKEN-BUS".

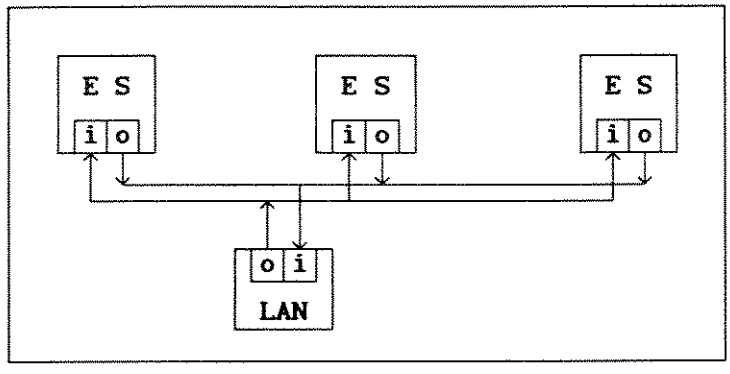


Fig. 5.2 - Configuração de conexão da rede

As principais considerações a respeito da rede são as citadas a seguir.

As cargas geradas nos ES correspondem a mensagens de mesmo tamanho. No entanto, as frequêmcias de geração de carga variam em função tanto do ES origem como dos ESs destino. Devido ao tipo de análise desejada, não se considerada limite de capacidade na armazenagem de mensagens nos ESs origem.

A comunicação entre ESs realiza-se após o estabelecimento de uma conexão. Esta é estabelecida após o envio de um pedido de conexão por um ES e posterior confirmação da conexão pelo outro ES. O término de uma conexão resume-se a uma liberação da conexão, efetuada por qualquer um dos dois ES, quando não mais houver mensagens a serem enviadas para o outro.

Não se considera a ocorrência de priorização entre mensagens a serem enviadas, embora estas só sejam enviadas após a resposta a todos os pedidos de conexão serem confirmados e os pedidos de conexão serem enviados. O tempo de transmissão dos pedidos de conexão, ou sua confirmação, são desprezíveis frente ao tempo de transmissão da mensagem padrão, bem como frente ao tempo de posse do token pelo ES.

Cada ES tem a si atribuído um determinado tempo de posse do token, dentro do qual realiza seus serviços na rede. O controle de posse de token para os ESs é realizado por um gerenciador da rede.

No caso de haver expirado o prazo de posse do token, antes que todas as mensagens sejam enviadas, o ES aguarda uma nova concessão para a continuação de seus serviços. No caso de um ES encerrar as suas transmissões, antes que tenha expirado seu prazo de posse do token, este sinaliza liberando o token. No caso de um ES não necessitar da utilização da rede para a transmissão de mensagens, o token não lhe é concedido.

Para a modelagem dessa rede, há a necessidade do reconhecimento das entidades envolvidas e o estabelecimento de suas características e comportamento.

Considera-se a existência de um End System genérico, cujas instâncias correspondem às entidades representantes das diferentes estações de trabalho que compõem a rede.

Para cada ES, consideram-se entidades geradoras de carga, uma para cada ES com o qual a estação de trabalho possa estabelecer comunicação. Com isto, tem-se uma grande flexibilidade no estabelecimento do volume de comunicações a ser gerado para cada ES remoto. As mensagens destinadas a distintos ESs remotos são armazenadas em buffers distintos, no Es origem.

O gerenciamento da rede é desenvolvido por uma entidade denominada Lan. Esta efetua o controle do tempo de token a ser concedido a cada ES, utilizando-se da ajuda de uma outra entidade denominada "Timer". A existência desse Timer faz-se presente, uma vez que o simulador utilizado é direcionado a eventos e não a tempo. Dessa forma, necessita-se de uma entidade que simule o comportamento de um relógio, para a geração da sequencialidade do tempo.

Em função dos resultados desejados, são considerados apenas três ES, operando simultaneamente. Neste caso, consideram-se apenas dois canais de comunicação para cada ES.

O comportamento de cada uma destas entidades é detalhadamente apresentado a seguir.

END SYSTEM

- verifica se existem mensagens a serem transmitidas, ou confirmação de pedidos de conexão recebidos (connect_request). Em caso negativo, repete este passo.
- solicita o token.
- verifica a concessão do token. Em caso negativo, repete este passo.
- encaminha as confirmações de conexão (connect_confirm).
- verifica a existência de mensagens para ES remotos com os quais não haja conexão já estabelecida.
- encaminha pedido de conexão a estes ESs.
- verifica se ainda se encontra de posse do token. Em caso negativo, retorna a quatro passos acima nesta sequência.
- verifica se ainda existem mensagens a serem transmitidas. Em caso negativo, libera o token e retorna ao primeiro passo desta sequência.
- verifica os buffers de mensagens, de forma cíclica, para constatar a existência de mensagens a serem transmitidas para ES com os quais exista conexão já estabelecida.
- envia uma mensagem para o ES remoto.
- retorna a quatro passos acima nesta sequência para a verificação da posse do token.

Em relação ao desenvolvimento desta sequência, reconhecem-se os seguintes eventos, condições e atitudes, para cada instância do ES.

1 - pede_token - efetua a indicação de solicitação de token, quan do houver mensagem a enviar, ou conexão a ser confirmada. Após a ocorrência deste evento, podem ocorrer os eventos de número: 2, 3, 5, 6, 7, 8.

2 - connect_confirmA - efetua a indicação de confirmação de conexão no primeiro canal de comunicação, se estiver de posse do token. Após a ocorrência

deste evento, podem ocorrer os eventos de número: 3, 6, 7, 8, 10, 4.

3 - connect_confirmB - efetua a indicação de confirmação de conexão no segundo canal de comunicação, se estiver de posse do token. Após a ocorrência deste evento, podem ocorrer os eventos de número: 5, 7, 8, 9, 2, 4.

4 - libera_token - efetua a indicação de liberação do token, no caso de estar de posse do token e não mais houver mensagens a serem transmitidas. Após a ocorrência deste evento, pode ocorrer apenas o evento de número 1.

5 - connect_requestA - efetua indicação de solicitação de conexão no primeiro canal de comunicação, se estiver de posse do token. Após a ocorrência deste evento, podem ocorrer os eventos de número: 3, 6, 7, 8, 10, 4.

6 - connect_requestB - efetua indicação de solicitação de conexão no segundo canal de comunicação, se estiver de posse do token. Após a ocorrência deste evento, podem ocorrer os eventos de número: 2, 5, 7, 8, 9, 4.

7 - envia_mensagemA - efetua a indicação do envio de uma mensagem pelo primeiro canal de comunicação, se estiver de posse do token. Também sinaliza a retirada de uma mensagem do buffer de mensagens a serem enviadas no primeiro canal de comunicação. Após a ocorrência deste evento, podem ocorrer os eventos de número: 3, 6, 8, 7, 9, 10, 4.

8 - envia_mensagemB - efetua a indicação do envio de uma mensagem pelo segundo canal de comunicação, se estiver de posse do token. Também sinaliza a retirada de uma mensagem do buffer de mensagens a serem enviadas no segundo canal de comunicação. Após a ocorrência deste evento, podem ocorrer os eventos de número: 2, 5, 7, 8, 9, 10, 4.

9 - disconnectA - efetua a indicação de desconexão do primeiro canal de comunicação, se estiver de posse do token. Após a ocorrência deste evento, podem ocorrer os eventos de número: 3, 5, 6, 7, 8, 10, 4.

10- disconnectB - efetua a indicação de desconexão do segundo canal de comunicação, se estiver de posse do token. Após a ocorrência deste evento, podem ocorrer os eventos de número: 2, 5, 6, 7, 8, 9, 4.

LAN

- faz com que a indicação do ES a receber o token seja o primeiro do anel lógico.
- verifica se este ES está com solicitação de token. Em caso negativo, vai para a indicação do próximo ES no anel lógico, seis passos abaixo, nesta

sequência.

- concede o token e inicializa a contagem de tempo de posse de token para este ES.
- verifica se o prazo de concessão de token já está expirado. Caso afirmativo, vai para a retirada do token, dois passos abaixo, nesta sequência.
- verifica se o ES ainda deseja o token. Caso afirmativo, retorna à verificação do tempo de token, no passo anterior desta sequência.
- retira o token.
- verifica se a indicação de posse de token estava com o último ES do anel lógico. Em caso afirmativo, retorna ao primeiro passo desta sequência.
- passa a indicação de token para o próximo ES do anel lógico.
- retorna ao segundo passo desta sequência para a verificação de solicitação do token.

Em relação ao desenvolvimento desta sequência, reconhecem-se os seguintes eventos, condições, e atitudes, para cada instância do ES.

1 - da_token1 - efetua a indicação de concessão de token ao primeiro ES do anel lógico. Também indica a inicialização da contagem do tempo de posse do token. Após a ocorrência deste evento, apenas o evento de número 2 pode ocorrer.

2 - retira_token1 - indica a retirada do token do primeiro ES do anel lógico, se o tempo de posse de token expirou, ou o ES indicou a liberação do token. Após a ocorrência deste evento, podem ocorrer os eventos de número: 3, 5, 1.

3 - da_token2 - efetua a indicação de concessão de token ao segundo ES do anel lógico. Também indica a inicialização da contagem do tempo de posse do token. Após a ocorrência deste evento, apenas o evento de número 4 pode ocorrer.

4 - retira_token2 - indica a retirada do token do segundo ES do anel lógico, se o tempo de posse de token expirou, ou o ES indicou a liberação do token. Após a ocorrência deste evento, podem ocorrer os eventos de número: 5, 1, 3.

5 - da_token3 - efetua a indicação de concessão de token ao terceiro ES do anel lógico. Também indica a inicialização da contagem do tempo de posse do token. Após a ocorrência deste evento, apenas o evento de número 6 pode ocorrer.

4 - retira_token3 - indica a retirada do token do terceiro ES do anel lógico, se o tempo de posse de token expirou, ou o ES indicou a liberação do token. Após a ocorrência deste evento, podem ocorrer os eventos de número: 1, 3, 5.

TIMER

O Timer apenas gera sinais de relógio, estabelecendo um ciclo equivalente a um relógio.

Assim sendo, o Timer executa um único evento:

1 - set_tempo - indica um incremento no buffer de controle de tempo.

GERADOR DE CARGA

Cada gerador de carga gera sinais de carga variável, representando as mensagens geradas para cada ES remoto.

Apresenta um único evento:

1 - gera_carga - indica um incremento no buffer de cargas geradas.

B - MODELAGEM DA REDE

A modelagem das entidades (frames, regras e procedimentos) que compõem a rede, denominada rede1, são apresentados a seguir.

B1 - ENTIDADES

ES1

nome : <valor> → es1
versão : <valor> → 1
data_hora : <valor> → 02-08-90
tipo : <valor> → rede1
membros : <valor> → []
portas :
ectA : <valor> → [num,1,1,0,_,_]
ectB : <valor> → [num,1,1,0,_,_]
mtA : <valor> → [num,1,1,0,_,_]
mtB : <valor> → [num,1,1,0,_,_]

```
    smtA     : <valor> → [num,1,1,0,_,_]
    smtB     : <valor> → [num,1,1,0,_,_]
  bandeiras :
    pede_token    : <valor> → [reset]
    tem_token     : <valor> → [reset]
    crA           : <valor> → [reset]
    crB           : <valor> → [reset]
    ccA           : <valor> → [reset]
    ccB           : <valor> → [reset]
  informações : <valor> → []

  eventos   :
     pede_token        : <valor> → [1,pede_token,_,_,hab,
                                    0,0,_,_,T]
     connect_confirmA  : <valor> → [2,cca,_,_,hab,
                                    1,0,_,_,T]
     connect_confirmB  : <valor> → [3,ccb,_,_,hab,
                                    1,0,_,_,T]
     libera_token      : <valor> → [4,libera_token,_,_,hab,
                                    0,0,_,_,T]
     connect_requestA  : <valor> → [5,crA,_,_,hab,
                                    1,0,_,_,T]
     connect_requestB  : <valor> → [6,crB,_,_,hab,
                                    1,0,_,_,T]
     envia_mensagemA   : <valor> → [7,env_mensA,_,_,hab,
                                   12,0,_,_,T]
     envia_mensagemB   : <valor> → [8,env_mensB,_,_,hab,
                                   12,0,_,_,T]
     disconnectA       : <valor> → [9,disconnectA,_,_,hab,
                                    0,0,_,_,T]
     disconnectB       : <valor> → [10,disconnectB,_,_,hab,
                                    0,0,_,_,T]
  rotinas :
     rot1 :
       sequência : <valor> → [4>1>/>2>/>3>/>5>/>6>/>7>
                              />8>/>9>/>10>/]
       decisão_fluxo : <regra> → [res1]
       regras_entidade : <regra> → [bes1]
     evento_rotina : <valor> → [_]
```

As regras referentes ao ES 1 são apresentadas a seguir:

- regras de entidade (referente aos eventos)

ES1

```
operadores ([]).

  executar(bes1,Meta,X) :-
        case([Meta = 'verifica a flag conn_reqA'
            → X = [verif_flag,es1,conn_reqA],
              Meta = 'verifica a flag conn_reqB'
            → X = [verif_flag,es1,conn_reqB],
              Meta = 'verifica se ectA maior que 0'
            → X = [comp_valor,es1,ectA,maior,0],
```

```
    Meta = 'verifica se ectB maior que 0'
  --> X = [comp_valor,es1,ectB,maior,0],
    Meta = 'verifica a flag tem_token'
  --> X = [verif_flag,es1,tem_token]
  ; X = nao_atrib]).
```

Este cabeçalho apresenta a definição descritiva de todos os procedimentos utilizados nas regras de decisão de fluxo abaixo. Como todos os operadores nelas utilizados são os já previamente definidos, não há a necessidade de definição de novos operadores, motivo pelo qual, no início do cabeçalho, a definição de operadores apresenta-se vazia.

```
regra_1 ind bes1_es1_rev :
            se 'verifica a flag conn_reqA',
            e  conn_reqA esta set,
            entao
                 [executar,pede_token,1,es1,rot1].

regra_2 ind bes1_es1_rev :
            se 'verifica a flag conn_reqB',
            e  conn_reqB esta set,
            entao
                 [executar,pede_token,1,es1,rot1].

regra_3 ind bes1_es1_rev :
            se 'verifica se ectA maior que 0',
            e  ectA eh maior_que_0,
            entao
                 [executar,pede_token,1,es1,rot1].

regra_4 ind bes1_es1_rev :
            se 'verifica se ectB maior que 0',
            e  ectB eh maior_que_0,
            entao
                 [executar,pede_token,1,es1,rot1].
```

Estas regras indicam que, se existir alguma mensagem a ser enviada (ectA ou ectB maior que zero), ou se existir algum pedido de conexão (crA ou crB no estado set), pode ser executado o evento "pede_token", de número 1, na rotina 1.

```
regra_5 ind bes1_es1_rev :
            se 'verifica a flag tem_token',
            e  tem_token esta set,
            entao
                 [executar,connect_confirmA,2,es1,rot1].
```

```
regra_6 ind bes1_es1_rev :
            se 'verifica a flag tem_token',
            e  tem_token esta set,
            entao
                 [executar,connect_confirmB,3,es1,rot1].
```

Estas duas regras indicam que, para a execução dos eventos "connect_confirmA" e "connect_confirmB", tem-se, como única condição, que o ES deve estar de posse do token (flag tem_token no estado "set").

```
regra_7 ind bes1_es1_rev :
            se 'verifica a flag tem_token',
            e  tem_token esta set,
            e  'verifica se ectA maior que 0',
            e  ectA eh maior_que_0,
            e  'verifica se ectB maior que 0',
            e  ectB eh maior_que_0,
            entao
                 [executar,libera_token,4,es1,rot1].
```

Esta regra indica que a execução do evento libera_token está condicionada a que o ES esteja de posse do token e que não exista carga para nenhum outro ES (ectA e ectB com valor 0).

```
regra_8 ind bes1_es1_rev :
            se 'verifica a flag tem_token',
            e  tem_token esta set,
            entao
                 [executar,connect_requestA,5,es1,rot1].

regra_9 ind bes1_es1_rev :
            se 'verifica a flag tem_token',
            e  tem_token esta set,
            entao
                 [executar,connect_requestB,6,es1,rot1].

regra_10 ind bes1_es1_rev :
            se 'verifica a flag tem_token',
            e  tem_token esta set,
            entao
                 [executar,envia_mensagemA,7,es1,rot1].

regra_11 ind bes1_es1_rev :
            se 'verifica a flag tem_token',
            e  tem_token esta set,
            entao
                 [executar,envia_mensagemB,8,es1,rot1].
```

```
regra_12 ind bes1_es1_rev :
             se 'verifica a flag tem_token',
             e  tem_token esta set,
             entao
                  [executar,disconnectA,9,es1,rot1].

regra_13 ind bes1_es1_rev :
             se 'verifica a flag tem_token',
             e  tem_token esta set,
             entao
                  [executar,disconnectB,10,es1,rot1].
```

Estas últimas seis regras indicam que, para a execução dos eventos "connect_requestA", "connect_requestB", "envia_mensagemA", "envia_mensagemB", "disconnectA" e "disconnectB", tem-se, como única condição, que o ES deve estar de posse do token (flag tem_token no estado "set").

- regras de decisão de fluxo

res1

```
operadores ([]).
    executar(res1,Meta,X) :-
          case([Meta = 'escreve evento 1 - rot1'
              --> X = [escreve,es1,rot1,1],
                Meta = 'escreve evento 2 - rot1'
              --> X = [escreve,es1,rot1,2],
                Meta = 'escreve evento 3 - rot1'
              --> X = [escreve,es1,rot1,3],
                Meta = 'escreve evento 4 - rot1'
              --> X = [escreve,es1,rot1,4],
                Meta = 'escreve evento 5 - rot1'
              --> X = [escreve,es1,rot1,5],
                Meta = 'escreve evento 6 - rot1'
              --> X = [escreve,es1,rot1,6],
                Meta = 'escreve evento 7 - rot1'
              --> X = [escreve,es1,rot1,7],
                Meta = 'escreve evento 8 - rot1'
              --> X = [escreve,es1,rot1,8],
                Meta = 'escreve evento 9 - rot1'
              --> X = [escreve,es1,rot1,9],
                Meta = 'escreve evento 10 - rot1'
              --> X = [escreve,es1,rot1,10],
                Meta = 'verifica se ectA maior que 0'
              --> X = [verif_valor,ectA,es1,maior,0],
                Meta = 'verifica se ectB maior que 0'
              --> X = [verif_valor,ectB,es1,maior,0],
                Meta = 'verifica se ectA igual 0'
```

```
                --> X = [verif_valor,ectA,es1,igual,0],
                  Meta = 'verifica se ectB igual 0'
                --> X = [verif_valor,ectB,es1,igual,0]
                ; X = nao_atrib]).
```

Este cabeçalho apresenta a definição descritiva de todos os procedimentos utilizados nas regras de decisão de fluxo abaixo. Como todos os operadores nelas utilizados são os já previamente definidos, não há a necessidade de definição de novos operadores, motivo pelo qual, no início do cabeçalho, a definição de operadores apresenta-se vazia. Nota-se, aqui, a flexibilidade obtida pela representação de regras adotada, ao ter-se uma mesma composição tanto para regras de entidade com para regras de decisão de fluxo.

```
regra_1 ind res1_es1_rev :
            se 'verifica a flag crA',
            e  crA esta set,
            e  'escreve evento 2 - rot1',
            entao
                 [decisao,pede_token,1,es1,rot1].
```

Esta regra indica que, uma vez ocorrido o evento "pede_token", se existe pedido de conexão para o primeiro canal de comunicação (bandeira crA no estado "set"), então o próximo evento a ocorrer é o de número 2, da rotina 1.

```
regra_2 ind res1_es1_rev :
            se 'verifica a flag crB',
            e  crB esta set,
            e  'escreve evento 3 - rot1',
            entao
                 [decisao,pede_token,1,es1,rot1].
```

Esta regra indica que, uma vez ocorrido o evento "pede_token", se existe pedido de conexão para o segundo canal de comunicação (bandeira crB no estado "set"), então o próximo evento a ocorrer é o de número 3, da rotina 1.

```
regra_3 ind res1_es1_rev :
            se 'verifica se ectA maior que 0',
            e  ectA eh maior_que_0,
            e  'verifica a flag ccA',
            e  ccA esta reset,
            e  'escreve evento 5 - rot1',
            entao
                 [decisao,pede_token,1,es1,rot1].
```

Esta regra indica que, uma vez ocorrido o evento "pede_token", se existe pedido mensagem a ser mandada pelo primeiro canal de comunicação (ectA maior

que 0) e não existe conexão neste canal (bandeira ccA no estado "reset"), então o próximo evento a ocorrer é o de número 5, da rotina 1.

```
regra_4 ind res1_es1_rev :
            se 'verifica se ectB maior que 0',
            e  ectB eh maior_que_0,
            e  'verifica a flag ccB',
            e  ccB esta reset,
            e  'escreve evento 6 - rot1',
            entao
                 [decisao,pede_token,1,es1,rot1].
```

Esta regra tem a mesma composição que a anterior, apenas referindo-se ao segundo canal de comunicação, cuja decisão leva ao evento "connect_requestB", de número 6.

```
regra_5 ind res1_es1_rev :
            se 'verifica se ectA maior que 0',
            e  ectA eh maior_que_0,
            e  'verifica a flag ccA',
            e  ccA esta set,
            e  'escreve evento 7 - rot1',
            entao
                 [decisao,pede_token,1,es1,rot1].

regra_6 ind res1_es1_rev :
            se 'verifica se ectB maior que 0',
            e  ectB eh maior_que_0,
            e  'verifica a flag ccB',
            e  ccB esta set,
            e  'escreve evento 8 - rot1',
            entao
                 [decisao,pede_token,1,es1,rot1].
```

Estas duas regras concluem pela ocorrência dos eventos "envia_mensagemA" ou "envia_mensagemB", no caso de haver mensagem a ser encaminhada e existir conexão estabelecida.

Estas seis regras são utilizadas para a decisão de qual dos seis possíveis eventos deva ocorrer após a ocorrência do evento "pede_token".

A composição das demais regras de decisão de fluxo, para aplicação após a ocorrências de cada um dos outros eventos, apresenta a mesma forma das apresentadas acima, em relação ao primeiro evento. No propósito de evitar-se repetições, é apresentado abaixo apenas um exemplo de cada, para a situação em que a conclusão seja para indicação de ocorrência dos eventos 9, 10, e 4.

```
regra_7 ind res1_es1_rev :
              se 'verifica se ectA igual 0',
              e  ectA eh igual_a_0,
              e  'verifica a flag ccA',
              e  ccA esta set,
              e  'escreve evento 9 - rot1',
              entao
                   [decisao,connect_confirmB,3,es1,rot1].


regra_8 ind res1_es1_rev :
              se 'verifica se ectB igual 0',
              e  ectB eh igual_a_0,
              e  'verifica a flag ccB',
              e  ccB esta set,
              e  'escreve evento 10 - rot1',
              entao
                   [decisao,connect_confirmA,2,es1,rot1].

regra_9 ind res1_es1_rev :
              se 'verifica se ectA igual 0',
              e  ectA eh igual_a_0,
              e  'verifica se ectB igual 0',
              e  ectB eh igual_a_0,
              e  'escreve evento 4 - rot1',
              entao
                   [decisao,connect_confirmA,2,es1,rot1].
```

Os procedimentos de eventos. referentes à entidade End System 1, são apresentados a seguir.

Proced_es1

```
pede_token([es1,rot1]) :- bandeira(es1,pede_token,set).

ccA([es1,rot1]) :- bandeira(es1,ccA,set).

ccB([es1,rot1]) :- bandeira(es1,ccB,set).

libera_token([es1,rot1]) :-
                   bandeira(es1,pede_token,reset).

crA([es1,rot1]) :- bandeira(es1,crA,set).

crB([es1,rot1]) :- bandeira(es1,crB,set).
```

```
env_mensA([es1,rot1]) :-
                altera_porta(es1,ectA,num,[-]),
                altera_porta(es1,mtA,num,[+]).

env_mensB([es1,rot1]) :-
                altera_porta(es1,ectB,num,[-]),
                altera_porta(es1,mtB,num,[+]).

disconnectA([es1,rot1]) :-
                bandeira(es1,crA,reset),
                bandeira(es1,ccA,reset).

disconnectB([es1,rot1]) :-
                bandeira(es1,crB,reset),
                bandeira(es1,ccB,reset).
```

Estes procedimentos apresentam as operações efetuadas sobre portas e bandeiras da entidade, como consequência da ocorrência de cada evento. Como os procedimentos são os mesmos que os já explicados anteriormente, apenas com mudança de argumentos, não apresentamos aqui uma explicação detalhada de cada um.

LAN

```
nome       : <valor> → lan
versão     : <valor> → 1
data_hora  : <valor> → 02-08-90
tipo       : <valor> → rede1
membros    : <valor> → []
portas     :
  tempo    : <valor> → [num,1,1,0,_,_]
  tempo1   : <valor> → [num,1,1,0,60,_]
  tempo2   : <valor> → [num,1,1,0,60,_]
  tempo3   : <valor> → [num,1,1,0,60,_]
bandeiras :
  pede_token1  : <valor> → [reset]
  pede_token2  : <valor> → [reset]
  pede_token3  : <valor> → [reset]
  da_token1    : <valor> → [reset]
  da_token2    : <valor> → [reset]
  da_token3    : <valor> → [reset]
informações : <valor> → []
eventos    :
   da_token1         : <valor> → [1,da_token1,_,_,des,
                                  1,0,_,_,T]
   retira_token1     : <valor> → [2,retira_token1,_,_,hab,
                                  1,0,_,_,T]
   da_token2         : <valor> → [3,da_token2,_,_,des,
```

```
                                        1,0,_,_,T]
      retira_token2    : <valor> ⟶ [4,retira_token2,_,_,hab,
                                        1,0,_,_,T]
      da_token3        : <valor> ⟶ [5,da_token3,_,_,des,
                                        1,0,_,_,T]
      retira_token3    : <valor> ⟶ [6,retira_token3,_,_,hab,
                                        1,0,_,_,T]

   rotinas :
      rot1 :
        sequência      : <valor> ⟶ [1>2>/>3>4>/>5>6>/]
        decisão_fluxo  : <regra> ⟶ [rlan]
        regras_entidade : <regra> ⟶ [blan]
      evento_rotina : <valor> ⟶ [_]
```

As regras referentes à LAN são apresentadas a seguir:

- regras de entidade (referente aos eventos)

```
   blan
  operadores ([]).
      executar(blan,Meta,X) :-
            case([Meta = 'verifica a bandeira pede_token1'
                ⟶ X = [verif_flag,lan,pede_token1],
                Meta = 'verifica a bandeira pede_token2'
                ⟶ X = [verif_flag,lan,pede_token2],
                Meta = 'verifica a bandeira pede_token3'
                ⟶ X = [verif_flag,lan,pede_token3],
                  Meta = 'compara limite superior tempo'
                ⟶ X = [comp_limite,lan,sup,tempo]
                ; X = nao_atrib]).
```

Este cabeçalho apresenta a definição descritiva dos procedimentos utilizados nas regras de entidade. O procedimento de comparação de limite faz parte do conjunto de procedimentos propostos para a avaliação do argumento "valor para decisão", constante da definição de cada porta. Este valor, neste caso, é utilizado para a verificação do prazo de posse do token.

A cada concessão do token, copia-se, para este argumento da porta tempo, o limite expresso na porta tempo1, tempo2, ou tempo3, referentes aos ES ao qual foi concedido o token. Dessa forma, apenas a porta "tempo" é utilizada como controle do tempo de token, recebendo, no devido tempo, os valores limites.

```
  regra_1 ind blan_lan_rev :
              se 'verifica a bandeira pede_token1',
              e  pede_token1 esta reset,
              entao
                   [executar,retira_token1,3,lan,rot1].
```

```
regra_2 ind blan_lan_rev :
            se 'compara limite superior tempo',
            e  limite esta vencido,
            entao
                 [executar,retira_token1,3,lan,rot1].
```

Estas duas regras apresentam as duas possíveis condições de ocorrência do evento "retira_token2". Estas condições são: o ES1 não mais deseja o token (bandeira "pede_token1" em estado "reset"), ou o tempo de token da entidade expirou (valor máximo da porta "tempo" foi atingido). A execução do procedimento "comp_lim" retorna uma informação, agregada à memória de trabalho de inferências, do tipo "limite esta vencido", ou "limite 'nao esta' vencido".

As regras referentes aos eventos "retira_token2" e "retira_token3" são parecidas com estas, apenas sendo alterado o nome das respectivas bandeiras, portas e dos eventos.

Os eventos "da_token1", "da_token2" e "da_token3" não apresentam condições de execução, por conseguinte não habilitando a consulta às regras de entidade. Isto se encontra especificado em suas definições, no frame da entidade.

- regras de decisão de fluxo

rlan

```
operadores ([]).
   executar(rlan,Meta,X) :-
         case([Meta = 'escreve evento 1 - rot1'
             --> X = [escreve,lan,rot1,1],
               Meta = 'escreve evento 3 - rot1'
             --> X = [escreve,lan,rot1,3],
               Meta = 'escreve evento 5 - rot1'
             --> X = [escreve,lan,rot1,5],
               Meta = 'verifica a bandeira pede_token1'
             --> X = [verif_flag,lan,pede_token1],
               Meta = 'verifica a bandeira pede_token2'
             --> X = [verif_flag,lan,pede_token2],
               Meta = 'verifica a bandeira pede_token3'
             --> X = [verif_flag,lan,pede_token3]
             ; X = nao_atrib]).
```

Este cabeçalho apresenta a definição descritiva de todos os procedimentos utilizados nas regras de decisão de fluxo abaixo. Como todos os operadores nelas utilizados são os anteriormente definidos, não há a necessidade de defi-

nição de novos operadores, motivo pelo qual, no início do cabeçalho, a definição de operadores apresenta-se vazia.

```
regra_1 ind rlan_lan_rev :
            se 'verifica a bandeira pede_token2',
            e  pede_token2 esta set,
            e  'escreve evento 3 - rot1',
            entao
                 [decisao,retira_token1,2,lan,rot1].

regra_2 ind rlan_lan_rev :
            se 'verifica a bandeira pede_token3',
            e  pede_token3 esta set,
            e  'escreve evento 5 - rot1',
            entao
                 [decisao,retira_token1,2,lan,rot1].

regra_3 ind rlan_lan_rev :
            se 'verifica a bandeira pede_token1',
            e  pede_token1 esta set,
            e  'escreve evento 1 - rot1',
            entao
                 [decisao,retira_token1,2,lan,rot1].
```

Estas três regras indicam que, uma vez ocorrido o evento "retira_token1", o próximo evento a ocorrer pode ser um dentre os de número 3, 5, ou 1. Esta ocorrência depende de qual esteja solicitando o token (bandeira pede_token no estado "set"). Como a ordem de aparecimento das regras, na base de conhecimento, induz a uma prioridade, o sequenciamento de apresentação das regras se faz em uma ordem cíclica. Com esta ordem, procura-se apresentar uma sequência que represente o anel lógico estabelecido.

Novas condições de inferência para decisão de fluxo se apresentam para os eventos de número 4 e 6. Nestes casos, as regras são semelhantes às apresentadas acima, apenas com troca na apresentação cíclica das regras e no nome do evento ocorrido anteriormente à decisão de fluxo.

Os procedimentos de eventos, referentes à entidade LAN são apresentados a seguir.

Proced_lan

```
da_token1([lan,rot1]) :-
          bandeira(lan,da_token1,set),
          altera_porta(lan,tempo,num,[w,0]),
          transf_limite(lan,tempo1,tempo).
```

```
retira_token1([lan,rot1]) :-
          bandeira(lan,da_token1,reset).


da_token2([lan,rot1]) :-
          bandeira(lan,da_token2,set),
          altera_porta(lan,tempo,num,[w,0]),
          transf_limite(lan,tempo2,tempo).


retira_token2([lan,rot1]) :-
          bandeira(lan,da_token2,reset).


da_token3([lan,rot1]) :-
          bandeira(lan,da_token3,set),
          altera_porta(lan,tempo,num,[w,0]),
          transf_limite(lan,tempo3,tempo).


retira_token3([lan,rot1]) :-
          bandeira(lan,da_token3,reset).
```

Estes procedimentos apresentam as operações efetuadas sobre portas e bandeiras da entidade, como consequência da ocorrência de cada evento. Os procedimentos do tipo "altera_porta" e "bandeira" são semelhantes aos já explicados anteriormente, apenas com mudança de argumentos.

O procedimento "transf_limite" é responsável pela transferência do valor do argumento "valor para decisão" de uma porta para outra. Este valor, atribuído à porta descrita como seu primeiro argumento, é copiado para a porta descrita como seu segundo argumento.

De uma forma genérica, ao ser concedido o token a um ES, procede-se à execução dos seguintes procedimentos: coloca-se a bandeira "da_tokenX", relativa ao ES referido, no estado "set"; escreve-se zero no valor atual da porta "tempo", reiniciando a contagem de tempo de posse do token; transfere-se o "valor para decisão" da porta definida para cada ES para a porta contadora de tempo.

TIMER

```
nome      : <valor> --> timer
versão    : <valor> --> 1
data_hora : <valor> --> 02-08-90
tipo      : <valor> --> rede1
membros   : <valor> --> []
```

```
portas     :
  tempo    : <valor> ⟶ [num,1,1,0,_,_]
bandeiras  : <valor> ⟶ []

informações : <valor> ⟶ []

eventos    :
   set_tempo        : <valor> ⟶ [1,set_tempo,_,_,des,
                                  1,0,_,_,T]
rotinas :
   rot1 :
     sequência      : <valor> ⟶ [1>/]
     decisão_fluxo  : <regra> ⟶ [rtimer]
     regras_entidade : <regra> ⟶ []
   evento_rotina : <valor> ⟶ [_]
```

Devido à característica de operação da entidade Timer, esta não apresenta regras de entidade. Isto porque seu único evento ocorre incondicionalmente, tendo apresentado desabilitação da base de regras, em sua definição.

A única regra de decisão de fluxo, apresentada abaixo, aparece para representar um retorno ao único evento, não para uma verdadeira decisão de fluxo.

```
   rtimer
operadores ([]).
   executar(rtimer,Meta,X) :-
         case([Meta = 'escreve evento 1 - rot1'
             ⟶ X = [escreve,timer,rot1,1]
             ; X = nao_atrib]).

regra_1 ind rtimer_timer_rev :
             se 'escreve evento 1 - rot1',
             entao
                  [decisao,set_tempo,1,timer,rot1].
```

O único procedimento de evento da entidade TIMER é apresentado a seguir:

```
Proced_timer
   set_tempo([timer,rot1]) :-
                 altera_porta(timer,tempo,num,[+]).
```

MÓDULO DE CARGA

Para cada ES podem existir vários módulos de carga, um para cada canal de comunicação a ser estabelecido. No entanto, todos se apresentam como diferen-

tes instâncias de uma mesma entidade. A única alteração corresponde à frequência e volume de carga gerada para o outro ES, definido nos argumentos da porta denominada "carga".

Assim sendo, apresenta-se apenas um dos módulos de carga, denominado C1T2, ou seja, gerador de carga do ES 1 para o ES 2.

C1T2

```
nome       : <valor> → c1t2
versão     : <valor> → 1
data_hora  : <valor> → 02-08-90
tipo       : <valor> → rede1
membros    : <valor> → []
portas     :
  carga    : <valor> → [num,1,1,0,_,_]
bandeiras  : <valor> → []
informações : <valor> → []
eventos    :
   gera_carga  : <valor> → [1,gera_carga,_,_,des,
                            3,0,_,_,T]
rotinas :
   rot1 :
     sequência      : <valor> → [1>/]
     decisão_fluxo  : <regra> → [rc1t2]
     regras_entidade : <regra> → []
   evento_rotina : <valor> → [_]
```

Devido à característica de operação da entidade geradora de carga C1T2, esta não apresenta regras de entidade. Isto porque seu único evento ocorre incondicionalmente, tendo apresentado desabilitação da base de regras, em sua definição.

A única regra de decisão de fluxo, apresentada abaixo, aparece para representar um retorno ao único evento, não para uma verdadeira decisão de fluxo.

```
   rc1t2
operadores ([]).
   executar(rc1t2,Meta,X) :-
         case([Meta = 'escreve evento 1 - rot1'
             → X = [escreve,c1t2,rot1,1],
             ; X = nao_atrib]).

regra_1 ind rc1t2_c1t2_rev :
             se 'escreve evento 1 - rot1',
             entao
                  [decisao,gera_carga,1,c1t2,rot1].
```

O único procedimento de evento da entidade C1T2 é apresentado a seguir:

```
Proced_c1t2
     gera_carga([c1t2,rot1]) :-
                    altera_porta(c1t2,carga,num,[+]).
```

B2 - SISTEMA

Uma representação gráfica do sistema desejado é apresentada na figura 5.3 abaixo. Esta configuração indica os canais de comunicação física e de controle entre as entidades, representados pelas ligações entre diferentes portas e bandeiras.

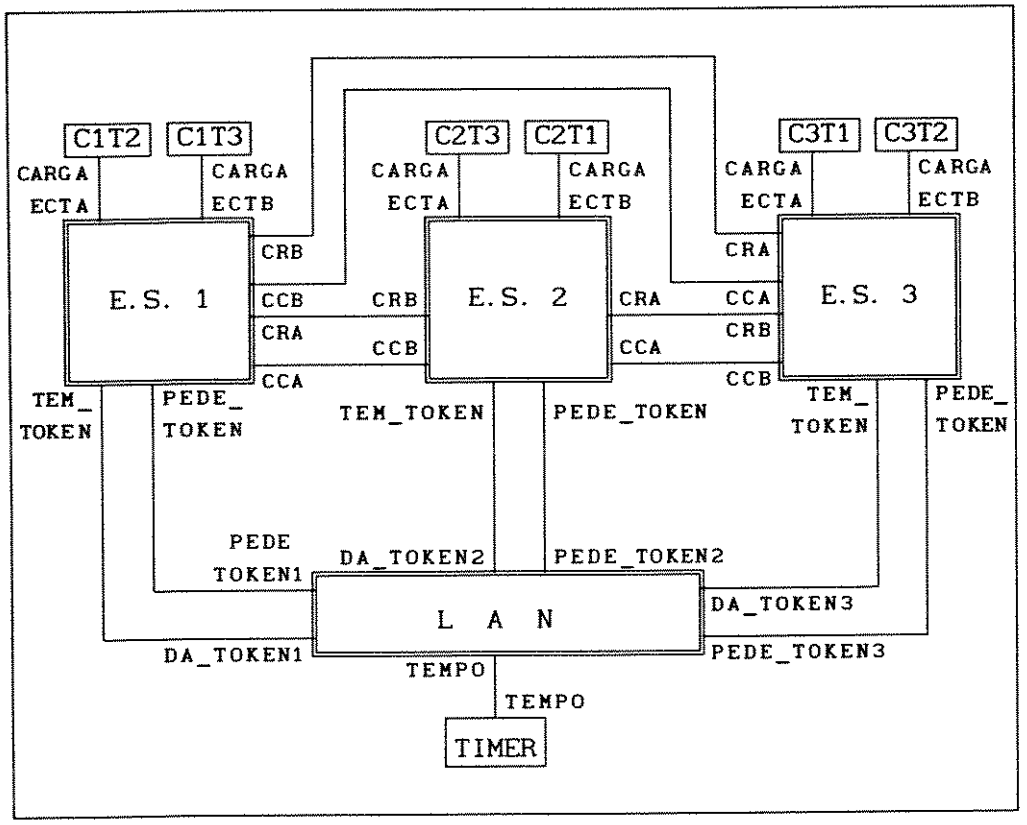


Fig. 5.1 - Representação gráfica do sistema Rede1

O frame, as regras e os procedimentos, que formam a composição do sistema a ser simulado, são apresentados abaixo.

STATUS

```
     nome       : <valor> ⟶ status
     versão     : <valor> ⟶ 1
     data_hora  : <valor> ⟶ 02-08-90
     tipo       : <valor> ⟶ rede1
     membros    : <valor> ⟶ []
     entidades  :
                es1  : <valor> ⟶ []
                es2  : <valor> ⟶ []
                es3  : <valor> ⟶ []
                lan  : <valor> ⟶ []
                c1t2 : <valor> ⟶ []
                c1t3 : <valor> ⟶ []
                c2t3 : <valor> ⟶ []
                c2t1 : <valor> ⟶ []
                c3t1 : <valor> ⟶ []
                c3t2 : <valor> ⟶ []
                timer: <valor> ⟶ []
     matriz_ligação :
          portas     : <valor> ⟶  [ecta_es1=carga_c1t2,
                                   ectb_es1=carga_c1t3,
                                   ecta_es2=carga_c2t3,
                                   ectb_es2=carga_c2t1,
                                   ecta_es3=carga_c3t1,
                                   ectb_es3=carga_c3t2,
                                   tempo_lan=tempo_timer]
          bandeiras : <valor> ⟶  [cra_es1=crb_es2,
                                   cca_es1=ccb_es2,
                                   crb_es1=cra_es3,
                                   ccb_es1=cca_es3,
                                   cra_es2=crb_es3,
                                   cca_es2=ccb_es3,
                                   pede_token_es1=pede_token1_lan,
                                   tem_token_es1=da_token1_lan,
                                   pede_token_es2=pede_token2_lan,
                                   tem_token_es2=da_token2_lan,
                                   pede_token_es3=pede_token3_lan,
                                   tem_token_es3=da_token3_lan]
     condições_iniciais  :
          portas     : <valor> ⟶  [ecta_es1=0, ectb_es1=0,
                                   ecta_es2=0, ectb_es2=0,
                                   ecta_es3=0, ectb_es3=0,
                                   mta_es1=0, mtb_es1=0,
                                   mta_es2=0, mtb_es2=0,
                                   mta_es3=0, mtb_es3=0,
                                   tempo1_lan=0, tempo2_lan=0,
                                   tempo3_lan=0, tempo_lan=0,
                                   carga_c1t2=0, carga_c1t3=0,
                                   carga_c2t3=0, carga_c2t1=0,
                                   carga_c3t1=0, carga_c3t2=0,
                                   tempo_timer=0]
          bandeiras : <valor> ⟶  [cra_es1=reset, crb_es2=reset,
                                   cca_es1=reset, ccb_es2=reset,
                                   crb_es1=reset, cra_es3=reset,
                                   ccb_es1=reset, cca_es3=reset,
```

```
                                    cra_es2=reset, crb_es3=reset,
                                    cca_es2=reset, ccb_es3=reset,
                                    pede_token_es1=reset,
                                    pede_token1_lan=reset,
                                    tem_token_es1=reset,
                                    da_token1_lan=reset,
                                    pede_token_es2=reset,
                                    pede_token2_lan=reset,
                                    tem_token_es2=reset,
                                    da_token2_lan=reset,
                                    pede_token_es3=reset,
                                    pede_token3_lan=reset,
                                    tem_token_es3=reset,
                                    da_token3_lan=reset]
condições_sistema    : <regra> ⟶ [brede1]
condições_término    : <regra> ⟶ [bred1trm]
clock                : <valor> ⟶ [0]
```

As regras referentes ao sistema não são necessárias neste exemplo. Isto ocorre, uma vez que todo o controle de operação entre entidades da célula foi efetuado através das bandeiras de controle.

Neste caso, na base "brese1" tem-se um conjunto de fatos, ao invés de um conjunto de regras. Isto faz com que a ocorrência dos eventos seja inquestionável, após seu escalonamento. Para cada evento, de cada uma das duas entidades, apresenta-se um fato como o abaixo.

```
fato_1 ind brede1_es1 : [executar,pede_token,1,es1,rot1]
```

As condições de término são apresentadas pelo conjunto de regras denominado "bred1trm", apresentado abaixo.

bred1trm

```
operadores ([]).
executar(sys,Meta,X) :-
         case([Meta = 'avalia a porta mta_es1'
             ⟶ X = [verif_valor,mta,es1,igual,10000]
             ; X = nao_atrib]).

regra_1 ind bcel1trm_sys_rev :
             se 'avalia a porta mta_es1'
             e  mta eh igual_a_10000,
             entao 'terminar'.
```

Esta regra indica que a condição de término da simulação corresponde à totalização de 10000 mensagens enviadas do ES1 para o ES2.

5.4 - CÉLULA FLEXÍVEL DE PROCESSAMENTO COM DUAS MÁQUINAS

Este exemplo corresponde à modelagem de uma célula flexível de processamento, operando com duas máquinas distintas. Estas máquinas podem processar peças de duas diferentes classes.

A caracterização deste exemplo é muito semelhante à apresentada em [92], na qual foram acrescentadas questões de tempo real, conforme [100].

O propósito principal deste exemplo é a apresentação de como se constitui um modelo de simulação, baseado no modelo declarativo de informação, a partir de especificações de planta e controlador apresentados em GRTTL. A metodologia usada para a realização da transformação de uma especificação em outra é a já descrita no capítulo 4.

a) DESCRIÇÃO DA CÉLULA

A operação desta célula corresponde ao processamento de dois diferentes tipos de peças, denomindas C1 e C2, por dois tipos de máquinas distintas, denominadas M1 e M2.

As restrições à operação da célula são as seguintes:

R1 - Toda peça, de ambas as classes C1 e C2, devem visistar ambas as máquinas, M1 e M2, não importando qual máquina seja visitada primeiro.

R2 - As peças são processadas em exclusão mútua: duas peças não podem ser processadas na mesma máquina, ao mesmo tempo. Em outras palavras, peças são processadas uma por vez, em cada máquina.

R3 - As peças têm a seguinte distribuição para acessar as máquinas: peças da classe C1 podem visitar a máquina M1 com probabilidade p_1 e a máquina M2 com probabilidade p_2; peças de classe C2 apresentam probabilidade complementar de visita às máquinas M1 e M2.

R4 - Sendo considerado igual a Z o tempo de processamento, nas duas máquinas, o intervalo de tempo entre os processamentos da peça pelas duas máquinas

nunca pode ser maior que 2Z.

A figura 5.4 abaixo pode dar uma idéia das transições do sistema.

Como descrição dos estados tem-se:

O - A peça está fora da estação de trabalho.
W - A peça está esperando para ser processada.
P - A peça está sendo processada pela máquina 1.
Q - A peça está sendo processada pela máquina 2.
C - A peça está esperando para ser processada pela máquina 1.
D - A peça está esperando para ser processada pela máquina 2.

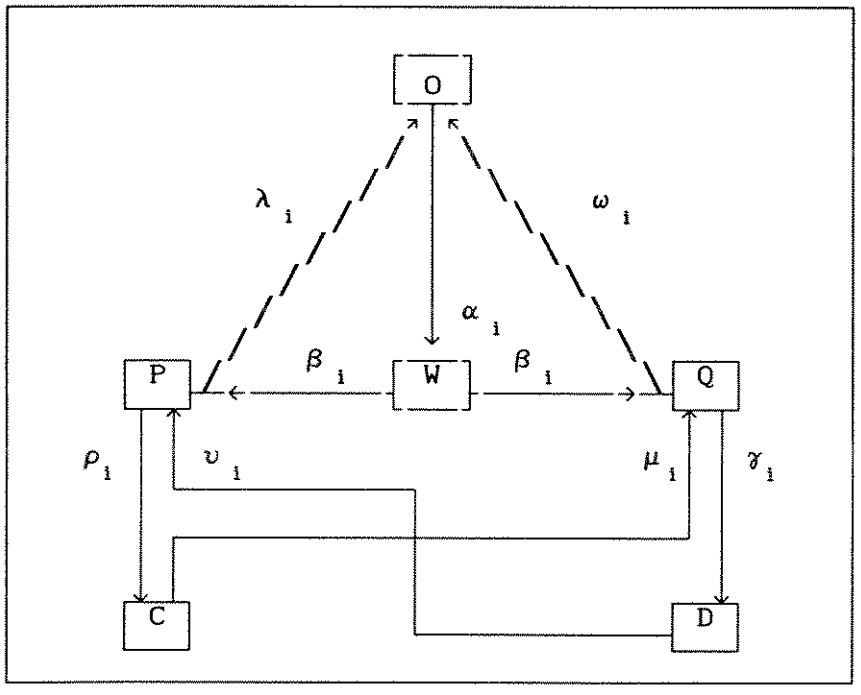


Fig. 5.4 - Transições de estado efetuadas na célula

Como descrição dos eventos tem-se:

α_i - Chegada de uma peça.
β_i - Escolha estocástica para processamento por M1 ou M2.
γ_i - Entrada na fila de espera para processamento pela máquina M1.
ρ_i - Entrada na fila de espera para processamento pela máquina M2.
υ_i - Começo de processamento pela máquina M1.
μ_i - Começo de processamento pela máquina M2.

λ_i - Término do processamento na máquina M1 e saída da estação de trabalho.

ω_i - Término do processamento na máquina M2 e saída da estação de trabalho.

b) ESPECIFICAÇÃO DA CÉLULA EM GRTTL

AXIOMAS DA PLANTA

P1) $\Box[\underline{\bigvee}_{i=1}^{n}(\delta=\alpha_i \vee \delta=\beta_i \vee \delta=\rho_i \vee \delta=\gamma_i \vee \delta=\mu_i \vee \delta=\upsilon_i \vee \delta=\lambda_i \vee \delta=\omega_i)]$

Esta fórmula apresenta os eventos passíveis de ocorrerem.

P2) $\Box\{\bigwedge_{i=1}^{n}[\delta=\alpha_i \Rightarrow x_i=0 \wedge (\odot x_i)=W \wedge (\bigwedge_{i\neq j=1}^{n}(\odot x_j)=x_j)]\}$

Esta fórmula descreve o efeito da chegada da i-ésima peça. Seu estado muda de O para W, enquanto os estados das outras peças permanecem o mesmo.

P3) $\Box\{\bigwedge_{i=1}^{n}[\delta=\beta_i \Rightarrow x_i=W \wedge [(\odot x_i)=P \vee (\odot x_i)=Q] \wedge \Delta(\odot x_i)=P \wedge \Delta(\odot x_i)=Q$

$\wedge (\bigwedge_{i\neq j=1}^{N}(\odot x_j)=x_j) \wedge (l_P=M \wedge u_P=M) \wedge (l_Q=M \wedge u_Q=M)]\}$

Esta fórmula descreve as mudanças de estado, motivadas pelas escolhas estocásticas efetuadas pela célula de trabalho, referente a qual máquina deva trabalhar a peça. Se o evento foi β_i, o próximo estado tanto pode ser P (máquina M1) como Q (máquina M2). Todas as outras peças mantém-se com seus estados inalterados. O intervalo de tempo para ocorrência destas transições é livre.

P4) $\Box\{\bigwedge_{i=1}^{n}[\delta=\rho_i \Rightarrow x_i=P \wedge t=T \wedge (\odot x_i)=C \wedge t=T+Z \wedge (\bigwedge_{i\neq j=1}^{n}(\odot x_j)=x_j)]\}$

P5) $\Box\{\bigwedge_{i=1}^{n}[\delta=\gamma_i \Rightarrow x_i=Q \wedge t=T \wedge (\odot x_i)=D \wedge t=T+Z \wedge (\bigwedge_{i\neq j=1}^{n}(\odot x_j)=x_j)]\}$

As fórmulas P4 e P5 descrevem o efeito da ocorrência dos eventos ρ_i e γ_i. Se o evento for $\rho_i(\gamma_i)$, a peça termina o processamento na máquina M1(M2), ou

seja, sairá do estado P(Q) e irá para o estado C(D). Os outros estados mantém-se constantes e o tempo entre a ocorrência de β_i e ρ_i (β_i e γ_i) é de Z unidades de tempo.

$$P6)\Box\{\bigwedge_{i=1}^{n}\ [\delta=\mu_i \Rightarrow x_i=C \wedge (\odot x_i)=Q \wedge (\bigwedge_{i\neq j=1}^{n}(\odot x_j)=x_j) \wedge (l_Q=M \wedge u_Q=M)]\}$$

$$P7)\Box\{\bigwedge_{i=1}^{n}\ [\delta=\upsilon_i \Rightarrow x_i=D \wedge (\odot x_i)=P \wedge (\bigwedge_{i\neq j=1}^{n}(\odot x_j)=x_j) \wedge (l_P=M \wedge u_P=M)]\}$$

As fórmulas P6 e P7 descrevem o efeito da ocorrência dos eventos μ_i e υ_i. Se o evento for $\mu_i(\upsilon_i)$, a peça é direcionada para a máquina M2(M1), ou seja, sai do estado C(D) e vai para o estado Q(P). Os outros estados ficam inalterados e o tempo decorrido entre os eventos ρ_i e μ_i (γ_i e υ_i) é livre.

$$P8)\Box\{\bigwedge_{i=1}^{n}\ [\delta=\lambda_i \Rightarrow x_i=P \wedge (\odot x_i)=O \wedge (\bigwedge_{i\neq j=1}^{n}(\odot x_j)=x_j) \wedge (l_O=M \wedge u_O=M)]\}$$

$$P9)\Box\{\bigwedge_{i=1}^{n}\ [\delta=\omega_i \Rightarrow x_i=Q \wedge (\odot x_i)=O \wedge (\bigwedge_{i\neq j=1}^{n}(\odot x_j)=x_j) \wedge (l_O=M \wedge u_O=M)]\}$$

As fórmulas P8 e P9 descrevem o efeito da ocorrência dos eventos λ_i e ω_i. Se o evento for $\lambda_i(\omega_i)$, a peça sai da máquina M1(M2), ou seja, sai do estado P(Q) e vai para o estado O. Os outros estados permanecem inalterados e o tempo decorrido entre os eventos υ_i e λ_i (μ_i e ω_i) é livre.

$$P10)\Box[O\neq W \wedge O\neq P \wedge O\neq Q \wedge O\neq C \wedge O\neq D \wedge W\neq P \wedge W\neq Q \wedge W\neq C \wedge W\neq D \wedge P\neq Q \wedge$$
$$\wedge P\neq C \wedge P\neq D \wedge Q\neq C \wedge Q\neq D \wedge C\neq D]$$

A fórmula P10 indica que todos os estados são distintos.

$$P11)\bigwedge_{i=1}^{n} x_i=O.$$

A fórmula P11 indica que todas as peças estão fora da célula de trabalho, no estado inicial.

ESPECIFICAÇÕES DE MALHA FECHADA EM GRTTL

São as seguintes as fórmulas que apresentam as especificações de malha fechada.

$$MF1)\Box[\overset{n}{\underset{i=1}{\wedge}}(x_i=P \Rightarrow \overset{n}{\underset{i\neq j=1}{\wedge}} x_j\neq P)]$$

$$MF2)\Box[\overset{n}{\underset{i=1}{\wedge}}(x_i=Q \Rightarrow \overset{n}{\underset{i\neq j=1}{\wedge}} x_j\neq Q)]$$

Estas fórmulas indicam que as peças são processadas uma por vez, em cada máquina.

$$MF3)\Box[\overset{n}{\underset{j=1}{\wedge}}(\overset{n}{\underset{i=1}{\vee}}(x_i=C \vee x_i=D)=>\delta=\beta_j)]$$

A fórmula MF3 indica que, se existem peças já processadas pela máquina M1 esperando para processamento, a escolha de peças não se processa de forma estocástica, e sim estas peças têm prioridade de escolha para processamento. Peças já processadas por uma das máquinas são chamadas de "peças-pf".

$$MF4)\ \Box[\overset{n}{\underset{\substack{i=1\\ j=1\\ i\neq j}}{\wedge}}(x_i\neq C => ((\delta=\rho_i\ \mathbb{P}\ \delta=\rho_j) => (\delta=\mu_i\ \mathbb{P}\ \delta=\mu_j)))]$$

$$MF5)\ \Box[\overset{n}{\underset{\substack{i=1\\ j=1\\ i\neq j}}{\wedge}}(x_i\neq D => ((\delta=\gamma_i\ \mathbb{P}\ \delta=\gamma_j) => (\delta=\upsilon_i\ \mathbb{P}\ \delta=\upsilon_j)))]$$

As fórmulas MF4 e MF5 mostram que as peças já processadas uma vez (peças-pf) são reprocessadas na ordem em que chegam: Se a próxima chegada da peça *i* precede a próxima chegada da peça j, então a peça *i* é enviada antes da peça j.

$$MF6)\Box[((\overset{n}{\underset{j=1}{\wedge}}(x_i=0 \vee x_i=W)) \wedge \neg(\overset{n}{\underset{k=1}{\wedge}} x_k=0)) =>\overset{n}{\underset{j=1}{\vee}}\delta=\beta_j]$$

A fórmula MF6 indica que, se todas as peças estão fora da estação, ou todas estão esperando para serem processadas, mas nem todas estão fora da estação, então um evento tipo β_j deve ocorrer.

$$MF7)\Box[\wedge(((\delta=\gamma_i \vee \delta=\rho_i)\wedge t=T)=>(\Diamond(\delta=\upsilon_i \vee \delta=\mu_i)\wedge t\leq T+2Z))]$$

A fórmula MF7 indica que, se uma peça teve terminado seu processamento por qualquer uma das máquinas e vai para a fila de espera de reprocessamento, ela será reprocessada em um tempo máximo de 2Z unidades de tempo.

ESPECIFICAÇÃO DO CONTROLADOR

Nesta especificação, são usados alguns símbolos adicionais para representar as informações armazenados pelo controlador. Os símbolos p e q representam as filas de espera para processamento nas máquinas M2 e M1, respectivamente.

A cada elemento destas filas, são associados números inteiros, variando de 1 a N. Também pode-se associar às filas o símbolo ϵ, que denota fila vazia. O símbolo variável local $\underline{c}$ representa o número de peças na célula de trabalho, que ainda não foram processadas (note-se que estas não são peças-pf). Este símbolo assume valores inteiros não negativos.

Neste exemplo, além dos requisitos de controle, expressos pelas fórmulas MF1 a MF6, são apresentados requisitos de tempo real, na fórmula MF7. As considerações a respeito de requisitos de tempo real podem ser cumpridas pelo partilhar de eventos entre a planta e o controlador: em relação à saída da peça do primeiro processamento na máquina M1 (M2) e em relação aos eventos relativos à entrada da peça no segundo processamento na máquina M2 (M1). O detalhe é que, para o controlador, estas transições tem um limite inferior de Z e um limite superior de 2Z unidades de tempo.

Estas considerações podem ser representadas graficamente pela figura 5.5 abaixo.

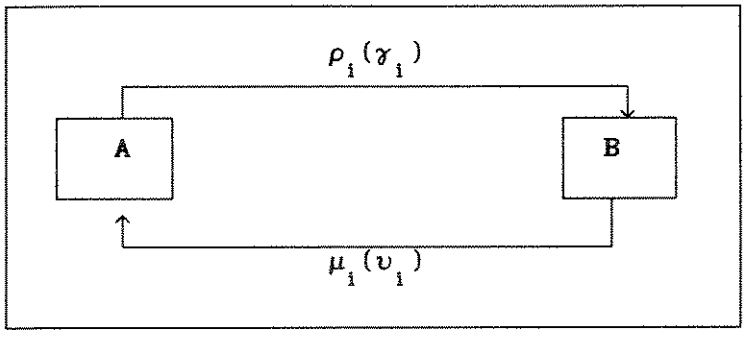


Fig. 5.5 - Representação dos estados do controlador

O diagrama da figura 5.5 apresenta os estados da variável Y_i do controlador, que sincroniza o início do segundo processamento. Quando o evento ρ_i ou γ_i ocorre, a Y_i vai para o estado **A**. A partir deste estado, o controlador impõe que o evento partilhado μ_i ou υ_i ocorra entre Z e 2Z unidades de tempo. Isto garante que a especificação MF7 seja cumprida.

As equações do controlador podem ser apresentadas como:

$$C1)\Box[\overset{n}{\underset{i=1}{\wedge}}(\delta=\rho_i=>((Y_i=\mathbf{A}) \wedge (\odot Y_i=\mathbf{B}) \wedge (\odot p)=p^*i \wedge (\odot c)=c-1)) \wedge l_B=Z \wedge u_B=2Z)]$$

$$C2)\Box[\overset{n}{\underset{i=1}{\wedge}}(\delta=\gamma_i=>((Y_i=\mathbf{A}) \wedge (\odot Y_i=\mathbf{B}) \wedge (\odot p)=p^*i \wedge (\odot c)=c-1))] l_B=Z \wedge u_B=2Z)]$$

As fórmulas C1 e C2 expressam, respectivamente, que se uma peça-pf, identificada como i-ésima peça, vem da máquina M1 (M2), o número "i" é colocado na fila p (q), e o valor do contador "c" é decrementado de 1. Estas fórmulas também expressam as cotas inferior "l_B" e superior "u_B" de tempo, para a tansição de um estado desativado **A**, para outro em que o controle de tempo está ativado **B** no controlador. Esta ativação do controlador corresponde ao desenvolvimento do acompanhamento da variável tempo que, para outras transições, não é necessária.

$$C3)\Box[\overset{n}{\underset{i=1}{\wedge}}((\delta=\mu_i)=>((Y_i=B) \wedge (\odot Y_i=\mathbf{A}) \wedge i < p \wedge (\odot p)=p \wedge (\odot c)=c)]$$

$$C4)\Box[\overset{n}{\underset{i=1}{\wedge}}((\delta=\upsilon_i)=>((Y_i=B) \wedge (\odot Y_i=\mathbf{A}) \wedge i < p \wedge (\odot p)=p \wedge (\odot c)=c)]$$

As fórmulas C3 e C4 expressam, respectivamente, que uma peça-pf é enviada para a máquina M2 (M1) apenas quando seu identificador corresponder à cabeça da fila p (q), sendo que a variável de estado do controlador muda de 1 para 0. Expressam, ainda, que o conteúdo da própria fila e do contador "c" não são alterados.

$$C5)\Box[\overset{n}{\underset{i=1}{\wedge}}(\delta=\omega_i=>(\odot p)=p|^1\wedge(\odot c)=c)]$$

$$C6)\Box[\overset{n}{\underset{i=1}{\wedge}}(\delta=\lambda_i=>(\odot q)=q|^1\wedge(\odot c)=c)]$$

As fórmulas C5 e C6, respectivamente, expressam que, se uma peça sai da máquina M2 (M1), a fila p (q) tem retirado seu primeiro elemento e o valor do contador fica inalterado.

$$C7) \Box[\bigwedge_{i=1}^{n}(\delta=\beta_i => (q=\epsilon \wedge (\odot q)=q \wedge p=\epsilon \wedge (\odot p)=p \wedge (\odot c)=c))]$$

A fórmula C7 expressa que uma escolha estocástica é feita só se as filas p e q estão vazias, à época desta escolha. Também que o valor do contador e o conteúdo das filas permanecem inalterados.

$$C8) \Box[\bigwedge_{i=1}^{n}(\delta=\alpha_i => \wedge\ (\odot q)=q \wedge (\odot p)=p \wedge (\odot c)=c+1)]$$

A fórmula C8 expressa que, se uma peça chega na célula de processamento, o valor do contador "c" é incrementado de 1 e que o conteúdo das filas p e q permanecem inalterados.

$$C9)\ q=\epsilon \wedge p=\epsilon \wedge c=0\ (\bigwedge_{i=1}^{n} Y_i = \mathbf{A}_i)$$

A fórmula C9 simplemente apresenta o estado inicial do controlador. No estado inicial, tem-se: as filas p e q estão vazias; o contador "c" possui o valor zero; e os estados das variáveis Y_i são inicializados como $\mathbf{A}_i$, significando que o controle do tempo de espera está desativado.

Com tais fórmulas, a descrição formal do controlador se completa.

c) MODELAGEM DA CÉLULA

O desenvolvimento do modelo da célula, a ser utilizado para simulação, segue os princípios estabelecidos nos itens 4.2-4 e 4.2-5 para mapeamento e transformação de um tipo de especificação em outro.

Para o desenvolvimento do modelo da célula para simulação, foram consideradas as expressões descritivas da planta, fórmulas P1 a P11 e as expressões descritivas do controlador, fórmulas C1 a C9. O propósito de executar-se a simulação da célula é a verificação do cumprimento das especificações de malha fechada, expressas pelas fórmulas MF1 a MF7.

Quanto às entidades, o exame das especificações leva a um inicial reco-

nhecimento de quais são as entidades da célula. Pode-se considerar toda a célula como apenas uma entidade ativa, que atua sobre as entidades passivas, as peças a serem processadas.

Por sua própria natureza, as especificações em GRTTL não se preocupam com o sequenciamento de ocorrências da célula de processamento, conforme entendido no modelo declarativo de informação. Estas especificações preocupam-se, sim, com o comportamento da célula, uma vez ocorrido cada evento, conforme descrito nas explicações das fórmulas P1 a P11.

Devido a isto, o sequenciamento de eventos da célula apresenta-se, nas especificações em GRTTL, completamente indeterminado. Com o propósito de guardar um íntimo relacionamento entre as diferentes especificações, considera-se que uma entidade externa indique, de forma aleatória, um dos possíveis eventos da célula para ocorrência.

Esta entidade externa, denominada Gerador de Eventos, incumbe-se apenas da indicação de um dos eventos da célula para ocorrência. O evento indicado para provável ocorrência é então avaliado, segundo os condicionamentos estabelecidos pela descrição da própria célula, e pela descrição do controlador.

A adoção desta entidade externa à célula, como entidade participante da simulação, permite uma maior flexibilidade de utilização do modelo, guardando, ainda, uma estreita relação com as especificações originais em GRTTL.

Da própria conceituação de um controlador de planta e de uma análise das fórmulas do controlador, percebe-se que estas fórmulas englobam dois tipos de assertivas: condicionantes que restringem o comportamento da planta e atitudes de controle. Devido a isto, as fórmulas do controlador podem ser expressas por meio de regras e procedimentos.

As regras expressam as condicionantes de ocorrência de cada evento, enquanto os procedimentos expressam as atitudes de controle do controlador. Ainda mais, considerando que nas especificações em GRTTL os eventos são considerados partilhados entre planta e controlador, também não ouve distinção entre eles na transformação das especificações.

Assim, o sistema proposto para a simulação da célula apresenta-se composto por duas entidades: a própria célula e o Gerador de Eventos.

A transformação de um tipo de especificação em outra, leva a uma descrição da célula como a apresentada abaixo, através de frames, regras e procedimentos, para a descrição do sistema, denominado "duasmaq".

c.1 - ENTIDADES

Célula

```
nome      : <valor> → celula
versão    : <valor> → 1
data_hora : <valor> → 25-11-91
tipo      : <valor> → duasmaq
membros   : <valor> → []
portas    :
     tipo      : <valor> → [num,1,1,0,_,_]
     entrada   : <valor> → [num,1,1,0,_,_]
     e1        : <valor> → [num,1,1,0,_,_]
     e2        : <valor> → [num,1,1,0,_,_]
     espera1   : <valor> → [num,1,1,0,_,_]
     espera2   : <valor> → [num,1,1,0,_,_]
     saida1    : <valor> → [num,1,1,0,_,_]
     saida2    : <valor> → [num,1,1,0,_,_]
     controle  : <valor> → [num,1,1,0,_,_]
     contador  : <valor> → [num,1,1,0,_,_]
     proc2m1   : <valor> → [num,1,1,0,_,_]
     proc2m2   : <valor> → [num,1,1,0,_,_]
     filap     : <valor> → [alfa,_,_,[],_,_]
     filaq     : <valor> → [alfa,_,_,[],_,_]

bandeiras : <valor> → []

informações : <valor> → []

eventos   :
   chega_peca  : <valor> → [1,chega_peca,_,_,hab,
                            0,0,_,_,T]
   escolhe_maq : <valor> → [2,escolhe_maq,_,_,hab,
                            Z,1,1,_,T]
   ini_esp_2pM2: <valor> → [3,ini_esp_2pM2,_,_,hab,
                            0,0,_,_,T]
   ini_esp_2pM1: <valor> → [4,inic_esp_2pM1,_,_,hab,
                            0,0,_,_,T]
   inic_2procM2: <valor> → [5,inic_2procM2,_,_,hab,
                            Z,0,_,_,T]
   inic_2procM1: <valor> → [6,inic_2procM1,_,_,hab,
                            Z,0,_,_,T]
   fim_2procM2 : <valor> → [7,fim_2procM2,_,_,hab,
                            0,0,_,_,T]
   fim_2procM1 : <valor> → [8,fim_2procM1,_,_,hab,
                            0,0,_,_,T]

rotinas :
   rot1 :
     sequência : <valor> → [/>1>/>2>/>3>/>4>/>5>/>
                            6>/>7>/>8>/]
     decisão_fluxo : <regra> → [rcelula]
     regras_entidade : <regra> → [bcelula]
   evento_rotina : <valor> → [_]
```

```
- regras de entidade (referente à descrição da célula)
    bcelula
  operadores ([]).
    executar(bcelula,Meta,X) :-
          case([Meta = 'avalia a porta entrada_celula'
              --> X = [verif_valor,entrada,celula,maior,0],
                Meta = 'avalia a porta e2_celula'
              --> X = [verif_valor,e2,celula,maior,0],
                Meta = 'avalia a porta e1_celula'
              --> X = [verif_valor,e1,celula,maior,0],
                Meta = 'avalia a porta espera2_celula'
              --> X = [verif_valor,espera2,celula,maior,0],
                Meta = 'avalia a porta espera1_celula'
              --> X = [verif_valor,espera1,celula,maior,0],
                Meta = 'avalia a porta proc2M2_celula'
              --> X = [verif_valor,proc2M2,celula,maior,0],
                Meta = 'avalia a porta proc2M1_celula'
              --> X = [verif_valor,proc2M1,celula,maior,0]
              ; X = nao_atrib]).

  regra_1 ind bcelula_celula_rev :
              se 'avalia a porta entrada_celula',
              e  entrada eh maior_que_0,
              entao
                   [executar,escolhe_maq,2,celula,rot1].

  regra_2 ind bcelula_celula_rev :
              se 'avalia a porta e2_celula',
              e  e2 eh maior_que_0,
              entao
                   [executar,ini_esp_2pM2,3,celula,rot1].

  regra_3 ind bcelula_celula_rev :
              se 'avalia a porta e1_celula',
              e  e1 eh maior_que_0,
              entao
                   [executar,ini_esp_2pM1,4,celula,rot1].

  regra_4 ind bcelula_celula_rev :
              se 'avalia a porta espera2_celula',
              e  espera2 eh maior_que_0,
              entao
                   [executar,inic_2procM2,5,celula,rot1].

  regra_5 ind bcelula_celula_rev :
              se 'avalia a porta espera1_celula',
              e  espera1 eh maior_que_0,
              entao
                   [executar,inic_2procM1,6,celula,rot1].
```

```
regra_6 ind bcelula_celula_rev :
            se 'avalia a porta proc2M2_celula',
            e  proc2M2 eh maior_que_0,
            entao
                 [executar,fim_2procM2,7,celula,rot1].

regra_7 ind bcelula_celula_rev :
            se 'avalia a porta proc2M1_celula',
            e  proc2M1 eh maior_que_0,
            entao
                 [executar,fim_2procM1,8,celula,rot1].
```

Estas regras apresentam apenas as condições internas da planta, para a ocorrência de cada evento. Expressam, principalmente, que: para escolher uma máquina há que existir peças esperando para processamento; para uma máquina operar deve existir peça disponível para tal; para concluir um processamento este deveria ter-se iniciado.

- regras de decisão de fluxo

rcelula

```
operadores ([]).
    executar(rcelula,Meta,X) :-
          case([Meta = 'escreve evento 1 - rot1'
              --> X = [escreve,celula,rot1,1],
                Meta = 'escreve evento 2 - rot1'
              --> X = [escreve,celula,rot1,2],
                Meta = 'escreve evento 3 - rot1'
              --> X = [escreve,celula,rot1,3],
                Meta = 'escreve evento 4 - rot1'
              --> X = [escreve,celula,rot1,4],
                Meta = 'escreve evento 5 - rot1'
              --> X = [escreve,celula,rot1,5],
                Meta = 'escreve evento 6 - rot1'
              --> X = [escreve,celula,rot1,6],
                Meta = 'escreve evento 7 - rot1'
              --> X = [escreve,celula,rot1,7],
                Meta = 'escreve evento 8 - rot1'
              --> X = [escreve,celula,rot1,8],
                Meta = 'le a porta tipo_celula'
              --> X = [le_valor,tipo,celula],
                Meta = 'verifica fila filap'
              --> X = [verif_fila,filap,celula],
                Meta = 'verifica fila filaq'
              --> X = [verif_fila,filaq,celula]
              ; X = nao_atrib]).
```

```
regra_1 ind rcelula_celula_rev :
            se 'le a porta tipo_celula'
            e  tipo vale 1
            e  'escreve evento 1 - rot1',
            entao
                 [decisao,_,_,celula,rot1].

regra_2 ind rcelula_celula_rev :
            se 'le a porta tipo_celula'
            e  tipo vale 2
            e  'verifica fila filap'
            e  filap eh []
            e  'verifica fila filaq'
            e  filaq eh []
            e 'escreve evento 2 - rot1',
            entao
                 [decisao,_,_,celula,rot1].

regra_3 ind rcelula_celula_rev :
            se 'le a porta tipo_celula'
            e  tipo vale 3
            e  'escreve evento 3 - rot1',
            entao
                 [decisao,_,_,celula,rot1].

regra_4 ind rcelula_celula_rev :
            se 'le a porta tipo_celula'
            e  tipo vale 4
            e  'escreve evento 4 - rot1',
            entao
                 [decisao,_,_,celula,rot1].

regra_5 ind rcelula_celula_rev :
            se 'le a porta tipo_celula'
            e  tipo vale 5
            e  'escreve evento 5 - rot1',
            entao
                 [decisao,_,_,celula,rot1].

regra_6 ind rcelula_celula_rev :
            se 'le a porta tipo_celula'
            e  tipo vale 6
            e  'escreve evento 6 - rot1',
            entao
                 [decisao,_,_,celula,rot1].

regra_7 ind rcelula_celula_rev :
            se 'le a porta tipo_celula'
            e  tipo vale 7
            e  'escreve evento 7 - rot1',
            entao
                 [decisao,_,_,celula,rot1].
```

```
regra_8 ind rcelula_celula_rev :
            se 'le a porta tipo_celula'
            e  tipo vale 8
            e  'escreve evento 8 - rot1',
            entao
                 [decisao,_,_,celula,rot1].
```

Estas regras apresentam que, para um evento ser escalonado para ocorrência, seu número deve ter sido indicado pelo gerador de tipos de eventos. Como única condição além destas, na regra_2, apresenta-se uma condição do controlador que indica a necessidade de ambas as filas de peças, já inicialmente processadas (peças pf), devem estar vazias. Esta condição é expressa pela fórmula C7 do controlador.

Os procedimentos de eventos, referentes à entidade célula, são apresentados a seguir. Em cada procedimento de evento abaixo descrito, o símbolo "*" indica os procedimentos básicos agregados, devido às ações de controle do controlador, expressas pelas atitudes de controle descritas nas fórmulas do controlador. Os outros referem-se às ações da própria célula.

Proced_celula

```
chega_peca([(celula,rot1)] :-
          altera_porta(celula,entrada,num,[+]),
          altera_porta(celula,contador,num,[+]),    *
          altera_porta(celula,controle,num,[+]).    *
```

Os procedimentos de controle, aqui incluídos, são consequência da fórmula C7. A informação contida na porta "controle" refere-se à implementação do identificador "i". Embora aparente repetição, as portas "entrada" e "contador" foram mantidas para uma distinção entre planta e controlador.

```
escolhe_maq([(celula,rot1)] :-
          gera_valor(aleat,int,1,2,valor),
          testa_prob(valor,p1,p2,maq),
          concat(e,maq,emaq),
          altera_porta(celula,emaq,num,[+]).

ini_esp_2pM2([(celula,rot1)] :-
          altera_porta(celula,e1,num,[-]),
          altera_porta(celula,espera1,num,[+]),
          concat_cauda(filap,controle),             *
          altera_porta(celula,contador,num,[-]).    *
```

```
ini_esp_2pM1([(celula,rot1)] :-
          altera_porta(celula,e2,num,[-]),
          altera_porta(celula,espera2,num,[+]),
          concat_cauda(filaq,controle),          *
          altera_porta(celula,contador,num,[-]).  *
```

Os procedimentos de controle, aqui incluídos, são consequência das fórmulas C1 e C2 do controlador.

```
inic_2procM2([(celula,rot1)] :-
     altera_porta(celula,espera2,num,[-]),
     altera_porta(celula,proc2M2,num,[+]),
     avalia_cabeca(filap,cabeca),                *
     concat_cabeca(celula,proc2M2,alfa,[cabeca]]).*

inic_2procM1([(celula,rot1)] :-
     altera_porta(celula,espera1,num,[-]),
     altera_porta(celula,proc2M1,num,[+]),
     avalia_cabeca(filaq,cabeca),                *
     concat_cabeca(celula,proc2M1,alfa,[cabeca]). *
```

Os procedimentos de controle, aqui incluídos, são consequência das fórmulas C3 e C4 do controlador.

```
fim_2procM2[(celula,rot1)] :-
          altera_porta(celula,proc2M2,num,[-]),
          altera_porta(celula,saida2,num,[+]),
          retira_cabeca(celula,filap,alfa,[cabeca]).   *

fim_2procM1[(celula,rot1)] :-
          altera_porta(celula,e2proc1,num,[-]),
          altera_porta(celula,saida1,num,[+]),
          retira_cabeca(celula,filaq,alfa,[cabeca]).   *
```

Os procedimentos de controle, aqui incluídos, são consequência das fórmulas C5 e C6 do controlador.

O modelo do gerador de tipos de eventos é apresentado pelo conjunto de frame, regras e procedimentos, abaixo.

Gerador

```
     nome       : <valor> ⟶ gerador
     versão     : <valor> ⟶ 1
     data_hora  : <valor> ⟶ 25-11-91
     tipo       : <valor> ⟶ celula
     membros    : <valor> ⟶ []
     portas     :
         tipo   : <valor> ⟶ [num,1,1,0,_,_]
     bandeiras  : <valor> ⟶ []
     informações : <valor> ⟶ []
     eventos    :
          gera_tipo   : <valor> ⟶ [1,gera_tipo,_,_,des,
                                   1,0,_,_,T]
     rotinas :
          rot1 :
            sequência : <valor> ⟶ [1>/]
            decisão_fluxo : <regra> ⟶ [rgerador]
            regras_entidade : <regra> ⟶ [bgerador]
          evento_rotina : <valor> ⟶ [_]
```

As regras de eventos, referentes ao gerador, são apresentadas a seguir:

- regras de entidade (referente aos eventos)

```
    bgerador
  operadores ([]).
  fato_1 ind bgerador_gerador :
              [executar,gera_tipo,1,gerador,rot1].
```

O único evento do gerador é incondicional; dessa forma sua ocorrência é apresentada como um fato. Embora a consulta a esta base tenha sido desabilitada pela definição do evento, aqui é apresentada como fato, facilitando futuras alterações.

- regras de decisão de fluxo

```
    rgerador
  operadores ([]).
  executar(rgerador,Meta,X) :-
          case([Meta = 'escreve evento 1 - rot1'
              ⟶ X = [escreve,gerador,rot1,1]
              ; X = nao_atrib]).

  regra_1 ind rgerador_gerador_rev :
              se 'escreve evento 1 - rot1',
              entao
                   [decisao,gera_tipo,1,gerador,rot1].
```

Esta regra é utilizada para forçar a repetição do evento responsável pela geração do tipo de evento a ser considerado na célula.

O procedimento do único evento da entidade gerador é definido como:

Proced_gerador

```
gera_tipo([(gerador,rot1)] :-
          gera_valor(aleat,int,1,8,valor),
          altera_porta(gerador,tipo,num,[w,valor]).
```

Este procedimento indica que a geração de tipo de evento corresponde à geração de um número inteiro, aleatório, no intervalo de 1 a 8. O valor gerado é escrito na porta "tipo" do gerador. Como esta porta se comunica com a porta "tipo" da célula, este número é transferido à célula para escolha de qual evento deva ocorrer.

B2 - SISTEMA

Uma representação gráfica do sistema desejado é apresentada na figura 5.6 abaixo. Esta configuração indica a porta "tipo", como o único canal de comunicação entre a célula e o gerador de tipos de eventos. Esta porta é utilizada para a transmissão da mensagem com o tipo de evento a ser testado pela célula, conseguindo-se, assim, o indeterminismo de sequenciamento sugerido nas especificações em GRTTL.

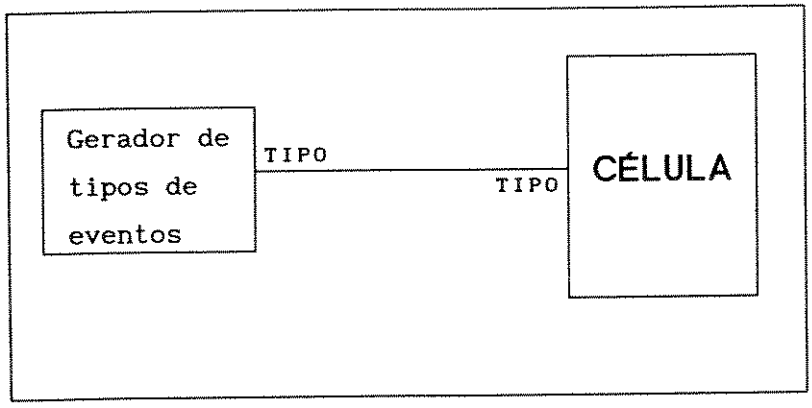


Fig. 5.6 - Representação Gráfica da célula de processamento com duas máquinas

O frame, as regras e os procedimentos, que formam a composição do sistema a ser simulado, são apresentados abaixo.

STATUS

```
nome       : <valor> ⟶ status
versão     : <valor> ⟶ 1
data_hora  : <valor> ⟶ 25-11-91
tipo       : <valor> ⟶ duasmaq
membros    : <valor> ⟶ []
entidades :
          celula    : <valor> ⟶ []
          gerador   : <valor> ⟶ []
matriz_ligação      :
     portas    : <valor> ⟶ [tipo_celula=tipo_gerador]
     bandeiras : <valor> ⟶ []

condições_iniciais  :
     portas    : <valor> ⟶ [tipo_gerador=0, tipo_celula=0,
                            entrada_celula=0, e1_celula=0,
                            e2_celula=0, espera1_celula=0,
                            espera2_celula=0,
                            saida1_celula=0,
                            saida2_celula=0],
                            controle_celula=0,
                            contador_celula=0,
                            proc2M1_celula=0,
                            proc2M2_celula=0,
                            filap_celula=[],
                            filaq_celula=[].
     bandeiras : <valor> ⟶ []

condições_sistema   : <regra> ⟶ [bduasmaq]
condições_término   : <regra> ⟶ [bduasmaqtrm]
clock               : <valor> ⟶ [0]
```

Este sistema, conforme modelado, não necessita das "regras de sistema", para interrelacionamento entre as entidades. Neste caso, na base "bduasmaq", tem-se um conjunto de fatos, ao invés de um conjunto de regras. Isto faz com que a ocorrência dos eventos seja inquestionável, após seu escalonamento. Para cada evento, de cada uma das duas entidades, apresenta-se um fato, habilitando sua execução.

As condições de término são apresentadas pelo conjunto de regras denominado "bduasmaqtrm", apresentado abaixo.

bduasmaqtrm

```
operadores ([]).

executar(sys,Meta,X) :-
          case([Meta = 'avalia a porta entrada_celula'
              --> X = [verif_valor,entrada,celula,igual,1000]
              ; X = nao_atrib]).


regra_1 ind bduasmaqtrm_sys_rev :
              se 'avalia a porta entrada_celula'
              e  entrada eh igual_a_1000,
              entao 'terminar'.
```

Esta regra indica que a condição de término da simulação é atingida quando são geradas mil peças para processamento no sistema.

6. CONCLUSÕES

6.1 - INTRODUÇÃO

Esta pesquisa foi direcionada no sentido do entendimento das questões relativas ao apoio à tomada de decisão, no ambiente da Manufatura Integrada por Computador.

A integração prática e eficiente das informações necessárias a um processo gerencial integrado das diferentes áreas organizacionais da manufatura não tem existido, a despeito da grande quantidade de sistemas computacionais de apoio operacional a estas áreas. Esta falta de integração deve-se, principalmente, à falta de padronização das informações, pois cada sistema adota uma estrutura para as informações necessárias à sua operação.

Dentre as ferramentas disponíveis para o apoio gerencial, as linguagens de simulação têm-se destacado como de grande valia. Nestas linguagens, a estruturação da informação, utilizada para a criação de modelos de simulação, as torna de difícil aprendizado, além de levar a modelos muito complexos, quando se procura representar sistemas flexíveis de manufatura.

Também devido à flexibilidade dos atuais sistemas de manufatura, a atividade decisória tornou-se muito complexa, já que trabalha com uma considerável multiplicidade de alternativas para a análise dos mesmos. Como elemento agravante, tem-se o fato de que o processo decisório se apresenta compartimentado, devido à falta de integração de informações entre as diferentes áreas organizacionais.

A meta deste trabalho tornou-se, então, o desenvolvimento de um ambiente de apoio à tomada de decisão, que integrasse, operacionalmente, a realização de simulação e de inferências lógicas. Foram estabelecidos como requisitos fundamentais do sistema que:

- sua operação fosse simples o suficiente, para que profissionais responsáveis pela tomada de decisão pudessem operá-lo diretamente;
- o modelo de informação utilizado fosse único para qualquer área organizacional da manufatura, criando-se uma base integrada de conhecimento.

Este ambiente deveria incluir as características de auxílio à especificação de modelos, presentes nas interfaces homem-máquina construídas como aperfeiçoamento das linguagens de simulação. Deveria, no entanto, apresentar-se

livre das restrições motivadas pelo estreito domínio de aplicação destas interfaces e também livre daquelas restrições motivadas pela própria estrutura das linguagens de simulação.

6.2 - RESULTADOS E AVALIAÇÃO

Desenvolveram-se, neste trabalho, o estudo, a especificação e a implementação de um Ambiente Integrado de Apoio à Tomada de Decisão.

Este ambiente destina-se a ser utilizado como instrumento de análise e fornecimento de informações que possam contribuir no projeto, planejamento, operação e gerenciamento de sistemas a eventos discretos. Oferece ferramentas computacionais para a realização de modelagens, simulações, previsões de comportamento e inferências lógicas, todas operando de maneira integrada e cooperativa.

O desenvolvimento do ambiente baseou-se nas premissas de operação cooperativa de duas ferramentas básicas de análise, um simulador a eventos discretos e um processador configurável de conhecimento, e na utilização de um mesmo modelo de informação, especificado tanto de maneira declarativa informal como de maneira formal.

A obediência às premissas básicas e aos requisitos impostos ao ambiente levou à criação de um Modelo Abstrato de Informação, contribuição central deste trabalho. Neste modelo, desenvolveu-se uma estrutura hierárquica de informação, capaz de ser utilizada de forma configurável pelo usuário, de maneira a representar complexos sistemas a eventos discretos de modo simples, sendo, ainda, que a informação, agregada a esta estrutura, pode ser diretamente processada por ambas as ferramentas básicas de análise do ambiente.

Destacam-se, assim, como principais características deste modelo, o ser: configurável pelo usuário, determinando a forma de apresentação da informação; incremental, facilitando a inclusão de novas informações; flexível, permitindo seu uso para diferentes propósitos; hierárquico, permitindo a composição da informação em vários níveis de detalhamento.

A abordagem utilizada para o desenvolvimento do modelo de informação diferiu, em parte, da metodologia tradicional de modelagem orientada a objetos, visto combinar uma forma flexível de representação do conhecimento, composta por três tipos de estruturas, frames, regras e procedimentos [95 a 101],

com as vantagens de uma programação simbólica e declarativa, mediante o uso da linguagem Prolog.

Os frames, coleção de slots organizados em estrutura de árvore, foram utilizados para a representação hierarquizada da informação. A cada slot corresponde uma parte da informação, contendo um dado, a indicação de uma regra (ou conjunto de regras) ou indicação de um procedimento (ou conjunto de procedimentos).

As regras, mais particularmente regras de produção, foram usadas para a representação de conhecimento dinâmico, expressando atitudes comportamentais, ou apresentando condicionamentos genéricos.

Os procedimentos foram usados, também, na representação de conhecimento dinâmico, para a definição de comportamentos específicos e de métodos computacionais.

Com o modelo de informação, assim composto, e com a possibilidade de configuração de cada uma destas estruturas e do conteúdo das mesmas, pôde-se representar a informação de maneira informal declarativa, quase coloquial. O mesmo modelo de informação adequa-se a representar informações expressas de maneira formal pela Lógica Temporal de Tempo Real Generalizada (GRTTL). Uma metodologia de transformação da representação formal do conhecimento para a representação informal declarativa foi desenvolvida, permitindo a conversão de fórmulas escritas em GRTTL para a estrutura de frames, regras e procedimentos.

A possibilidade de realização de uma especificação formal de entidades e posteriores simulações sobre estas especificações, abre a possibilidade de novas metodologias para o estudo de sistemas da manufatura, em particular nos estudos de sistemas de controle. Com isto pode-se, primeiramente, simular modelos de especificações para sistemas, antes de procurar-se sua prova formal. Simulações satisfatórias de certos modelos os habilitam a terem suas especificações consideradas para provas formais, que são complicadas e demoradas. Simulações que comprovem o não cumprimento dos requisitos estabelecidos para o comportamento de certos modelos evita o dispêndio de esforço com a tentativa de prova de um modelo inadequado.

A composição estrutural do ambiente foi desenvolvida de maneira modular, baseada em vários elementos funcionais projetados segundo o paradigma de objetos. Isto permite sua adequação ao uso em configuração distribuída e multiusuário, podendo este, então, operar como um Servidor de Apoio à Tomada de

Decisão. No entanto, em face das disponibilidades computacionais, muito pouco pôde ser realizado no sentido de avaliação desta capacidade.

Todos os elementos funcionais do ambiente operam sobre o mesmo modelo de informação, obtendo-se, assim, uma operação integrada e cooperativa, principalmente para as duas ferramentas básicas de análise. Tanto resultados de simulações podem ser utilizados como informação dinâmica, a ser aproveitada em novas inferências, como resultados de inferências podem ser usados como informação adicional para a realização de novas simulações. Esta possibilidade de realização de inferências lógicas sobre o resultado das simulações, ou vice-versa, apresenta-se como uma importante colaboração à análise de simulações, em muito contribuindo para o apoio à tomada de decisão.

Ambas as ferramentas de análise tiveram seu desenvolvimento influenciado pelo modelo de informação desenvolvido.

O Processador de Conhecimento apresenta uma estruturação específica para o tratamento dos frames, regras e procedimentos, motivado pelas características estruturais e de configurabilidade do modelo de informação. O processamento do conhecimento é simbólico, utilizando-se da linguagem Prolog.

O Simulador foi projetado de forma a privilegiar o aproveitamento da manipulação de informações realizada pelo Processador de Conhecimento, levando ao desenvolvimento de uma Lógica de Simulação [102 a 104], especialmente desenvolvida para a realização de simulações sobre informações estruturadas segundo o modelo adotado. Sua própria estruturação também apresenta um caráter modular, com sua operacionalidade distribuída por diversos elementos funcionais.

A operação do ambiente mostrou-se satisfatória, dentro dos parâmetros estabelecidos para seu desenvolvimento. Simulações realizadas utilizando-se o ambiente mostraram-se equivalentes àquelas realizadas com algumas linguagens de simulação. No entanto, os modelos utilizados no ambiente mostraram-se consideravelmente mais simples em sua construção e entendimento em relação àqueles desenvolvidos para linguagens de simulação.

Apesar de cumprirem-se os requisitos estabelecidos como metas para o desenvolvimento do ambiente, a avaliação de seu desempenho operacional sugere algumas considerações. O processamento simbólico, adotado através da linguagem Prolog, apresenta-se, por natureza, de certa forma lento, quando comparado em

relação a outras linguagens de programação. A configurabilidade atribuída ao modelo de informação adotado, por sua vez, requer certo volume de processamento prévio, até que um conjunto de informações se torne utilizável, internamente ao ambiente. Isto também contribui para um maior tempo de processamento durante a realização de simulações.

Entretanto, as características de processamento simbólico e de configurabilidade da informação, por parte do usuário, foram conscientemente escolhidas como características do ambiente, quando de sua especificação, em detrimento de seu desempenho computacional.

Apesar de toda a flexibilidade atribuída ao ambiente, sua estruturação o torna passível de utilização em sistemas computacionais de baixo custo. Os testes de operação do ambiente foram realizados em computadores do tipo Personal Computer (PC), em configuração comum ao ambiente empresarial a que se destina, apresentando resultados altamente satisfatórios.

6.3 - FUTUROS TRABALHOS

A título de trabalhos futuros, algumas considerações podem ser feitas, em diferentes sentidos: na complementação da implementação de alguns elementos funcionais do ambiente, na melhoria do desempenho computacional, e em aperfeiçoamentos ao ambiente através de novos desenvolvimentos.

Em termos de complementação da implementação do ambiente, cita-se o completo desenvolvimento das funções do Analisador de Resultados que, embora tendo sua funcionalidade já especificada, ainda não foi completamente implementado.

Em termos de melhoria do desempenho computacional do ambiente, duas linhas distintas de trabalho se apresentam. Uma, no sentido de portá-lo para novas plataformas computacionais, com maior desempenho, tais como as atuais estações de trabalho, as quais apresentam um consideravel potencial computacional, além de muito bom desempenho em processamento gráfico. Outra linha corresponde à otimização das inferências realizadas pelo Processador de Conhecimento. Uma avaliação das bases de regras, em relação à maneira em que foram construídas pelo usuário, poderia levar à eliminação de redundâncias, aumen-

penho na realização das inferências.

Em termos de aperfeiçoamentos ao ambiente através de novos desenvolvimentos, várias alternativas podem ser consideradas. Estes aperfeiçoamentos podem ser realizados em diferentes elementos do AIATD, dentre os quais citam-se, como mais importantes, aperfeiçoamentos no Aquisitor de Conhecimento, Máquina Generalizada de Inferência e Interface de Comunicação.

Quanto ao Aquisitor de Conhecimento, os principais aperfeiçoamentos a serem desenvolvidos são: quanto à execução de uma transformação semi-automática, ou mesmo automática, de especificações formais para modelos declarativos de simulação; quanto à criação de interfaces inteligentes para aprendizado automático e sugestão de soluções em diferentes áreas gerenciais.

A transformação de especificação formal em GRTTL para o modelo declarativo teve sua metodologia desenvolvida no capítulo 4. No entanto, esta transformação ainda requer, em muito, a atuação do usuário. Estudos são necessários tanto a respeito da GRTTL como a respeito do modelo declarativo, para uma melhor adequação de um ao outro, facilitando a realização de transformação automática.

A criação de interfaces inteligentes apresenta-se como outro ramo promissor para o desenvolvimento do AIATD. Uma aquisição automática de conhecimento poderia ser desenvolvida através do acompanhamento da realização de inferências, ou de simulações. No caso em que certas metas, ou condições, genéricas fossem estabelecidas, o Aquisitor de Conhecimento poderia avaliar resultados de passos intermediários de inferências, ou de simulações, e enviá-los à base de conhecimento, quando cumpridas algumas exigências préviamente estabelecidas. O conhecimento, assim adquirido, poderia ser utilizado tanto em novas simulações e inferências como no acompanhamento e orientação de modelagens futuras, a serem realizadas para entidades similares.

Quanto à Máquina de Inferência Generalizada, um aperfeiçoamento a ser considerado seria o de melhorar-se o gerenciamento das inferências, agregando-lhe certa quantidade de conhecimento a respeito da própria inferência em desenvolvimento. Isto poderia ser implementado através do uso da filosofia de "Blackboard" [34, 56, 57]. Isto, além de melhorar a eficiência da realização de inferências, usufruiria da capacidade de operação distribuída do ambiente, utilizando, eficientemente, o conhecimento distribuído. Outra direção de aperfeiçoamento corresponde à manipulação e inferência de conhecimento com certo grau de incerteza. A agregação de níveis de incerteza ao conhecimento adquiri-

do poderia aumentar bastante a similaridade das inferências com o raciocínio humano, principalmente em se tratando de inferências com encadeamento direto.

Em relação à Interface de Comunicação, o principal aperfeiçoamento a ser realizado corresponde à sua adequação ao trabalho em redes de comunicação onde se disponha de um Sistema de Processamento Distribuído Aberto (ODP). Neste caso, esta interface deveria também poder realizar serviços de negociação para a prestação de serviços de simulação e inferência, operando, então, como um Servidor de Apoio à Tomada de Decisão. Também a adequação aos padrões de ODP permitiria a utilização de recursos de Multimídia, para trabalho cooperativo de profissionais que estivessem desenvolvendo uma mesma utilização do Servidor de Apoio à Tomada de Decisão.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] - Abed, S. Y.; T. A. Barta, K. L. McRoberts. 1985. *"A Qualitative Comparison of Three Simulation Languages: GPSS/H, SLAM, SIMSCRIPT"*. Computers and Industrial Engineering, vol. 9 nº1, 35-43.

[2] - Abed, S. Y.; T. A. Barta, K. L. McRoberts. 1985. *"A Quantitative Comparison of Three Simulation Languages: GPSS/H, SLAM, SIMSCRIPT"*. Computers and Industrial Engineering, vol. 9 nº1, 45-66.

[3] - Abdin, M. F.; N. S. Mohamed. 1986. *"The Role of Simulation in Design of FMSs"*. Computers and Industrial Engineering, vol. 11 nº1-4, 372-376.

[4] - Adelsberger, H. 1984. *"PROLOG as a Simulation Language"*. In Proceedings of 1984 Winter Simulation Conference (Dallas, TX).

[5] - Alptekin, S.; D. L. Webber. 1988. *"A Systematic Transition to Flexible Manufacturing"*. In Proceedings of 10th Annual Conference on Computers and Industrial Engineering. Computers and Industrial Engineering, vol. 15 nº1-4, 8-13.

[6] - Amaral, W. C.; S. Bingulac; P. Ferreira; W. Fontanini, F. Gomide. 1988. *"A Knowledge Based Environment for Computer Aided Control Engineering"*. In Proceedings of the 4th IFAC CADCS, Beijing - China, 542-547.

[7] - Antonelli, C. J.; R. A. Volz; T. N. Mudge. 1986. *"Hierarchical Decomposition and Simulation of Manufacturing Cells Using ADA"*. Simulation, vol. 46 nº4, 141-152.

[8] - Bagrodia, R. L.; K. M. Chandi; J. Misra. 1987. *"A Message-Based Approach to Discrete-Event Simulation"*. IEEE Transactions on Software Engineering, vol. SE-13 nº6, 654-665.

[9] - Barbosa, L. M. S. 1987. *"Protocolos: Da Especificação ao Protótipo"*. RT - Universidade do Minho.

[10] - Barr, A.; E. Feigenbaum. 1982. *"The Handbook of Arificial Intelligence"* Vol. 2. William Kaufmann, Los Altos, CA.

[11] - Bell, P. C.; R. M. O'Keefe. 1987. *"Visual Interactive Simulation: History, Recent Developments, and Major Issues"*. Simulation, vol.49 nº3, 109-116.

[12] - Bensana, E; M. Correge; G. Bel; D. Dubois. 1986. *"An Expert System Approach in Industrial Job-Shop Scheduling"*. In Proceedings of 1986 IEEE International Conference on Robotics and Automation, 1645-1650.

[13] - Berenji, H. R.; B. Khoshnevis. 1986. *"Use of Artificial Intelligence in Automated Process Planning"*. Computer in Mechanical Engineering, vol. 9, 47-55.

[14] - Blackburn, M. R. 1989. *"Using Expert Systems to Construct Formal Specifications"*. IEEE Expert, vol. 4 nº1, 62-74.

[15] - Bobrow, D. J.; M. Stefik. 1986. *"Object-Oriented Programming: Themes and Variations"*. AI Magazine, vol. 6 nº4, 40-62.

[16] - Bruno, B.; A. Elia; P. Laface. 1986. *"A Rule Based System to Schedule Production"*. IEEE Computer, vol. 19 nº7, 32-40.

[17] - Bu-Hulaiga, M. I.; A. K. Chakravarty. 1988. *"An Object-Oriented Knowledge Representation for Hierarchical Real-Time Control of Flexible Manufacturing"*. International Journal of Production Research, vol. 26 nº5, 777-793.

[18] - Burns, J. R. 1990. *"A Specification Language for Generating Intelligent Discrete Next-Event Simulations"*. Information and Decision Technologies, vol. 16, 3-13.

[19] - Burns, J. R., J. D. Morgeson. 1988. *"An Object-Oriented World-View for Intelligent, Discrete, Next-Event Simulation"*. Management Science, vol. 34 nº12, 1425-1440.

[20] - Casanova, M. A. 1987. *"Programação em Lógica e a Linguagem PROLOG"*. Editora Edgard Blucher.

[21] - Chandrasekharan, M. P.; R. Rajagopalan. 1987. *"ZODIAC - An Algorithm for Concurrent Formation of Part-Families and Machine-Cells"*. International Journal of Production Research, vol. 25 nº6, 835-850.

[22] - Chang, T. C.; R. A. Wysk. 1985. *"An Introduction to Process planning Systems"*. Prentice Hall, Englewood Cliffs, N.J.

[23] - Chang, Y. L. 1986. *"Lot Sizing in a Flexible Assembly System"*. In Proceedings of 2th ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Methods and Applications, K. E. Stecke and R. Suri - eds, Elsevier, New York, 359-369.

[24] - Chaundry, S. S.; T.Khidhathong. 1988. *"Resource Management Using Microcomputers"*. Computers and Industrial Engineering, vol. 15 nº1-4, 166-171.

[25] - Cheng, T. C. E. 1985. *"Simulation of Flexible Manufacturing Systems"*. Simulation, vol. 45 nº6, 299-302.

[26] - Chorafas, D. S. 1967. *"System and Simulation"*. In Mathematics in Science and Engineering, vol. 14. Academic Press, New York.

[27] - Cleary, J; K.-S. Goh; B. Unger. 1985. *"Discrete Event Simulation in PROLOG"*. In Artificial Intelligence, Graphics, and Simulation. Proceedings of 1985 SCS Multiconference, 8-13, Birtwistle - ed., SCS.

[28] - Clocksin, F.; C. S. Mellish. 1984. *"Programming in Prolog"*. Springer Verlag.

[29] - Dahl, O. J.; K. Nygaard. 1966. *"SIMULA - An ALGOL-Based Simulation Language"*. Communications of the ACM, vol. 9 nº9, 671-678.

[30] - Dixon, J. R.; M. K. Simmons. 1983. *"Computers that Design: Expert Systems for Mechanical Engineers"*. Computers in Mechanical Engineering, vol. 2 nº3, 10-18.

[31] - Egbelu, P. J. 1986. *Planning for Machining Multijob, Multimachine Manufacturing Environment"*. Journal of Manufacturing Systems, vol. 5 nº1, 1-14.

[32] - Eid, M. S.; C. Poirier. 1988. *"An Expert System for Workshop Simulation"*. In Proceedings of 10th Annual Conference on Computers and Industrial Engineering. Computers and Industrial Engineering, vol. 15 nº1-4, 91-97.

[33] - Eloranta, E.; M. Syrjanen; S. Torma. 1990. *"Knowledge Based Tool for Manufacturing Systems Design"*. Computer-Integrated Manufacturing Systems, vol. 3 nº3, 163-170.

[34] - Engelmore, R. S.; A. J. Morgan. 1988. *"Blackboard Systems"*, Addison Wesley.

[35] - Evans, J. R.; D. R. Dolk; S. M. Shaffer. 1987. *"An Intelligent System for Production Planning Modeling and Optimization"*. In Proceedings of the 9th International Conference on Production Research, 1689-1693.

[36] - Fan, I.S.;P.J.Sackett. 1988. *"A PROLOG Simulator for Interactive Flexible Manufacturing Systems Control"*. Simulation, vol. 50 nº6, 239-247.

[37] - Fishman, G. S. 1973. *"Concepts and Methods in Discrete Event Simulation"*. John Wiley and Sons, New York, N.Y.

[38] - Fishwick, P. A. *"Process Abstraction in Simulation Modeling"*. 1989. In Artificial Intelligence Simulation and Modeling, L. E. Widman, K. A. Loparo, N. R. Nielsen - eds, John Wiley and Sons, New York.

[39] - Ford, D. R.; B. J. Schroer. 1987. *"An Expert Manufacturing Simulation System"*. Simulation, vol. 48 nº5, 193-200.

[40] - Foulds, L. R. 1983. *"Techniques for Facilities Layout: Deciding Which Pairs of Activities Should Be Adjacent"*. Management Science, vol. 29 nº12, 1414-1426.

[41] - Fox, M. S.; N. Husain; M. McRoberts; Y. V. Reddy. 1989. *"Knowledge Based Simulation: An Artificial Intelligence Approach to System Modeling and Automating the Simulation Life Cycle"*. In Artificial Intelligence Simulation and Modeling, L. E. Widman, K. A. Loparo, N. R. Nielsen - eds, John Wiley and Sons, New York.

[42] - Fox, M. S.; N. Sathi; V. Baskaran; J. Bouer. 1986. *"Simulation Craft: An Expert System for Discrete Event Simulation"*. in Simulators III: Proceedings of the SCS Simulators Conference, SCS, San Diego CA.

[43] - Ghosh, B. K.; R.A. Wysk. 1987. *"Modeling of Computer and Communication Networks in Flexible Manufacturing"*. In. Proceedings of 9th Annual Conference on Computers and Industrial Engineering. Computers and Industrial Engineering, vol. 13 nº1-4, 49-54.

[44] - Giozza, W. F. et alli. 1986. *"Redes Locais de Computadores"*. McGraw Hill - EMBRATEL.

[45] - Goldberg, A.; D. Robson. 1983. *"SMALLTALK-80: The Language and Its Implementation"*. Addison-Wesley, Reading, M.A.

[46] - Golden, D. G. 1985. *"Software Engineering Considerations for the Design of Simulation Languages"*. Simulation, vol. 45 nº4, 169-178.

[47] - Gordon, G. 1978. *"System Simulation"*. 2nd edition, Prentice Hall Inc. New Jersey.

[48] - Grant, T. J. 1986. *"Lessons for O. R. From A. I.: A Scheduling Case Study"*. Journal of Operational Research, vol. 37 nº1, 41-57.

[49] - Griesmeyer, J. M. 1989. *"Generalized Simulation Environment for Factory Systems"*. In Tools for the Simulation Profession, SCS, 20-28.

[50] - Groover, M. P.; E. W. Zimmers. 1984. *"CAD/CAM: Computer-Aided Design and Manufacturing"*. Prentice Hall, Englewood Cliffs, N.J.

[51] - Guariso, G.; M. Hitz; H. Werthner. 1989. *"An Intelligent Simulation Model Generator"*. Simulation, vol. 53 nº2, 57-66.

[52] - Gutschke, W.; K. Mertins. 1987. *"CIM: Competitive Edge in Manufacturing"*. Robotics and Computer-Integrated Manufacturing, vol. 3 nº1, 77-87.

[53] - Haddock, J. 1987. *"An Expert System Framework Based on a Simulation Generator"*. Simulation, vol. 48 nº2, 45-53.

[54] - Haddock, J. 1988. *"A Simulation Generator for Flexible Manufacturing Systems Design and Control"*. IIE Transactions, vol. 20 nº1, 22-31.

[55] - Hailpern, B. T. 1982. *"Verifying Concurrent Processes Using Temporal Logic"*. Lectures Notes on Computer Science, vol. 129, Springer Verlag.

[56] - Hayes-Roth, B. 1973. *"A Blackboard Architecture for Control"*. Artificial Intelligence, vol. 26, 251-321.

[57] - Hayes-Roth, B.; M. Hewett. 1988. *"BBI: An Implementation of the Blackboard Control Architecture"*. In Blackboard Systems, R. Engelmore and T. Morgan -eds, Addison Wesley.

[58] - Henderson, M. R.; S. Musti. 1988. *"Automated Group Technology Part Coding from a Three-Dimensional CAD Database"*. Transaction of the ASME, vol. 110

[59] - Henricksen, J. O.; R. C. Crain. 1983. *"GPSS/H User's Manual"*. Wolverine Software Corporation, VA.

[60] - Heragu, S.; A. Kusiak. 1988. *"Machine Layout Problem in Flexible Manufacturing Systems"*. Operations Research, vol. 36 nº2, 258-268.

[61] - Hitz, K. 1987. *"Flexible Integrated Computer-Aided Manufacturing Systems Increase Productivity"*. Robotics and Computer-Integrated Manufacturing, vol. 3 nº1, 123-128.

[62] - Hooper, J. M. 1986. *"Strategy-Related Characteristics of Discrete-Event Languages and Models"*. Simulation, vol. 46 nº4, 153-159.

[63] - Howe, A. E.; P. R. Cohen; J. R. Dixon; M. K. Simmons. 1988. *"Dominic: A Domain-Dependent Program for Mechanical Engineering Design"*. In Expert Systems in Engineering, D. T. Pham - ed, IFS Publications, Springer Verlag, New York, 361-371.

[64] - Hutchinson, G. K. 1984. *"Flexible Manufacturing Systems and Simulation"*. In Flexible Manufacturing Systems published by SME, 222-223.

[65] - Intellicorp. 1986. *"KEE - Software Development System"*. Intellicorp ed.

[66] - ISO TC97/CS16 - 1981. *"Open System Interconnection - Basic Reference Model"*. Computer Communication Review, nº2.

[67] - ISO-7492. 1982. *"Information Processing Systems - Open Interconnection, Basic Reference Model"*. International Standard Organization.

[68] - Issa, T. N.; Z. J. Czajkiewicz. 1987. *"MRP II - Manufacturing Resource Planning: System Development, Implementation, and Its Impact on Productivity at the Factory Level"*. In Modern Production Management Systems, A. Kusiak - ed, Elsevier, New York, 711-727.

[69] - Iwata, K.; N. Sugimura. 1985 *"An Integrated CAD/CAPP System with Know-Hows on Machining Accuracies of Parts"*. In Computer Aided Intelligent Process Planning, A.S.M.E., C. R. Liu, T. C. Chang, R. Komanduri - eds, 121-130.

[70] - Jackson, M. A. 1975. Principles of Program Design. Academic-Press, New York, N.Y.

[71] - Jones, G. K.; M. A. Greene. 1989. *"A Prototype Implementation of GPSS in SAS"*. Simulation, vol. 52 nº1, 10-17.

[72] - Joshi, S.; N. N. Vissa, T. C. Chang. 1988. *"Expert Process Planning System with Solid Model Interface"*. International Journal of Production Research, vol. 26 nº5, 863-885.

[73] - Kanet, J. J.; H. H. Adelsberger. 1987. *"Expert Systems in Production Scheduling"*. European Journal of Operational Research, vol. 29 nº1, 51-59.

[74] - Kaye, M. M.; Q. Sun. 1990. *"Data Manipulation for the Integration of Simulation with Online Production Control"*. Computer-Integrated Manufacturing Systems, vol. 3 nº1, 19-26.

[75] - Ketcham, M. G.; R. E. Shannon; G. L. Hogg. 1989. *"Information Structure for Simulation Modeling of Manufacturing Systems"*. Simulation, vol. 52 nº2, 59-67.

[76] - Kidd, A. L. 1987. *"Knowledge Acquisition for Expert Systems - A Practical Handbook"*. Plenum Press, New York.

[77] - King, J. R.; V. Nakornchai. 1982. *"Machine-Component Group Formation in Group Technology: Review and Extensions"*. International Journal of Production Resarch, vol. 20 nº2, 117-133.

[78] - Kiviat, P. J. 1971. *"Simulation Languages"*. In Computer Simulation Experiments with Models of Economic Systems, T. H. Taylor - ed, John Wiley and Sons, New York, N.Y.

[79] - Komorowsky, H. J.; J. Maluzzynsky. 1987. *"Logic Programming and Rapid Prototyping"*. Science of Computer Programming, vol. 9 nº2, 179-205.

[80] - Kramer, B.; H. Liebowitz. 1987. *"Intelligent Materials Processing"*. Robotics and Computer-Integrated Manufacturing, vol. 3 nº2, 141-145.

[81] - Kulkarni, V. M.; J. R. Dixon; M. K. Simmons, J. E. Sunderland. 1985. *"Expert Systems for Design: The Design of Heat Fins as an Example of Conflicting subgoals and the Use of Dependencies"*. Proceeedings of the ASME Computers in Engineering Conference.

[82] - Kumar, K. R.; A. Vannelli. 1986. *"Grouping of Parts and Components in Flexible Manufacturing Systems"*. International Journal of Production Research, vol. 25 nº12, 1715-1728.

[83] - Kumara, S. R. T.; R. L. Kashyap; C. L. Moodie. 1988. *"Application of Expert Systems and Pattern Recognition Methodologies to Facilities Layout Planning"*. International Journal of Production Research, vol. 26 nº5, 905-930.

[84] - Kusiak, A. 1990. *"Intelligent Manufacturing Systems"*. Prentice Hall, Englewood Cliffs, NJ.

[85] - Kusiak, A.; A. Vannelli; K. R. Kumar. 1985. *"Grouping Problem in Flexible Manufacturing Systems"*. Robotica, vol. 3, 245-252.

[86] - Kusiak, A,; G. Finke. 1987. *"Modeling and Solving the Flexible Forging Module Scheduling Problem"*. Engineering Optimization, vol. 12 nº1, 1-12.

[87] - Kusiak, A,; G. Finke. 1988. *"Selection of Process Plans in Automated Manufacturing Systems"*. IEEE Journal of Robotics and Automation, vol. 4 nº4, 397-402.

[88] - Kusiak, A.; S. S. Heragu. 1987. *"The Facility Layout Problem"*. European Journal of Operational Research, vol. 29 nº3, 229-253.

[89] - Kusiak, A.; W.S.Chow. 1987. *"Efficient Solving of the Group Technology Problem"*. Journal of Manufacturing Systems, vol. 6 nº2, 117-124.

[90] - Lehmann, D.; S. Shelah. 1982. *"Reasoning With Time and Chance"*. Information and Control, vol. 53, 165-198.

[91] - Levas, A.; R. Jayaraman. 1989. *"WADE: An Object-Oriented Environment for Modeling and Simulation of Workcell Applications"*. IEEE Transaction on Robotics and Automation, vol. 5 nº3, 324-335.

[92] - Lin, J. Y.; D. Ionescu. 1990. *A Generalized Temporal Logic Approach for Control Problems of a Class of Nondeterministic Discrete Event Systems"*. In Proceedings of CDC-90, Honolulu, 3440-3445.

[93] - Link, C. H. 1976. *"CAPP-CAM-I Automated Process Planning System"*. Proceedings of the 13th Numerical Control Society Annual Meeting and Technical Conference.

[94] - Lorimy, B.; P. Russel. 1986. *"CIM-OSA Strategic and Management Issues"*. In ESPRIT '88 - Putting the Technology to Use. Proceedings of the 5th Annual ESPRIT Conference, 1777-1793.

[95] - Loyolla, W. P. D. C. 1991. *"Uma Especificação Formal para Modelos de Simulação a Eventos Discretos"*. In Proceedings II Simpósio de Pesquisa e Pós-Graduação em Ciências Exatas e Engenharias, Campos do Jordão-SP.

[96] - Loyolla, W. P. D. C.; A. Signoretti, F. Gomide; M. J. Mendes. 1990. *"Modelagem de Entidades de Manufatura para Simulação"*. RT-DCA-006/90, DCA-FEE-UNICAMP.

[97] - Loyolla, W. P. D. C.; A. Signoretti, F. Gomide; M. J. Mendes. 1990. *"Um Ambiente Baseado em Conhecimento para Simulação de Sistemas"*. RT-004/90, DCA-FEE-UNICAMP.

[98] - Loyolla, W. P. D. C.; A. Signoretti; F. Gomide; M. J. Mendes. 1991. *"A Knowledge Based Simulation Server"*. In Proceedings 13th IMACS World Conference on Computation and Applied Mathematics, Dublin, Ireland.

[99] - Loyolla, W. P. D. C.; A. Signoretti; F. Gomide; M. J. Mendes; S. Bingulac. 1991. *"A Knowledge Based Simulation Environment"*. In Proceedings 5th IFAC/IMACS Symposium on Computer Aided Design in Control Systems CADCS 91, Swansea - United Kingdom.

[100] - Loyolla, W. P. D. C.; B. I. Silva Jr; R. Mendes; M. J. Mendes. 1992. *"Aplicação de Lógica Temporal de Tempo Real Generalizada na Modelagem e Simulação de Sistems a Eventos Discretos"*. In Proceedings do 9º Congresso Brasileiro de Automática - 9º CBA, Vitória-ES.

[101] - Loyolla, W. P. D. C.; F. Gomide; M. J. Mendes. 1992. *"A Knowledge Based Simulation Server"*. In Artificial Intelligence, Expert Systems, and Symbolic Computation, E. N. Houstis and J. R. Rice - eds, North Holland.

[102] - Loyolla, W. P. D. C.; M. J. Mendes. 1990. *"Lógica Temporal na Especificação de Simuladores para Sistemas Discretos da Manufatura"*. In Proceedings do 8º Congresso Brasileiro de Automática - 8º CBA, Belém-PA.

[103] - Loyolla, W. P. D. C.; M. J. Mendes. 1990. *"Lógica Temporal na Especificação e Verificação de Processos Concorrentes"*. RT-DCA-010/90, DCA-FEE-UNICAMP.

[104] - Loyolla, W. P. D. C.; M. J. Mendes. 1990. *"Arquitetura de um Servidor de Simulação em Redes Industriais da Manufatura"*. In Proceedings 1º Simpósio de Automação Integrada, Curitiba-PR.

[105] - Magnenat-Thalmann, N. 1987. *"Procedural Animation Blocks in Discrete Simulation"*. Simulation, vol. 49 nº3, 102-108.

[106] - Malmborg, C. J.; M. H. Agee; G. R. Simons; J. Y. Choudry. 1988. *"An Expert System for Selection of Material Handling Equipment"*. In Artificial Intelligence: Implications for Computer Integrated Manufacturing, A. Kusiak - ed, IFS Publications, Springer Verlag, New York, 484-504.

[107] - Mamalis, A. G.; N. G. Bilalis; M. J. Konstantinidis. 1987. *"On Simulation Modeling for FMS"*. Simulation, vol. 48 nº1, 19-23.

[108] - Manna,Z.;A.Pnueli. 1981. *"Verification of Concurrent Programs: Temporal Proof Principles"*. Lecture Notes on Computer Science vol. 131.

[109] - Manna, Z.; A. Pnueli. 1983. *"Proving Precedence Properties: The Temporal Way"*.Lectures Notes on Computer Science, vol. 154, Springer Verlag.

[110] - Manna, Z.; P. Wolper. 1981. *"Syntesis of Communications Processes from Temporal Logic Specifications"*. Lectures Notes on Computer Science, vol. 131.

[111] - McGlennon, J. M.; G. Cassidy; J. Browne. 1988. *"ROBOSPEC - A Prototype Expert System for Robot Selection"*. In Artificial Intelligence: Implications for Computer Integrated Manufacturing, A. Kusiak - ed, IFS Publications, Springer Verlag, New York, 505-515.

[112] - Mellichamp, J. M.; A. F. A. Wahab. 1987. *"An Expert System for FMS Design"*. Simulation, vol. 48 nº5, 201-208.

[113] - Mellichamp, J. M.; A. F. A. Wahab. 1987. *"Process Planning Simulation: An FMS Modeling Tool for Engineers"*. Simulation, vol. 48 nº5, 186-192.

[114] - Merchant, M. E. 1985. *"Computer-Integrated Manufacturing as the Basis for the Factory Of The Future"*. Robotics and Computer-Integrated Manufacturing, vol. 2 nº2, 89-99.

[115] - Merchant, M. E. 1988. *"The Precepts and Sciences of Manufacturing"*. Robotics and Computer-Integrated Manufacturing, vol. 4 nº1-2, 1-6.

[116] - Meyer, B. 1987. *"Reusability: The Case for Object-Oriented Design"*. IEEE Software, 50-64.

[117] - Meyer, W.; R. Isenberg. 1988. *"Aspects of Knowledge-Based Factory Supervision Systems"*. In ESPRIT '88 - Putting the Technology to Use. Proceedings of the 5th Annual ESPRIT Conference, 1671-1691.

[118] - Miller, D. M.; R. P. Davis. 1978. *"A Dynamic Resource Allocation Model for a Machine Requirements Problem"*. IIE Transactions, vol. 10 nº3, 237-243.

[119] - Milner, R. 1980. *"A Calculus of Communications Systems"*. Lectures Notes on Computer Science, vol. 92.

[120] - Minsky, M. 1975. *"A Framework for Representing Knowledge"*. In The Psychology of Computer Vision, P. H. Winston - ed., McGraw Hill, New York.

[121] - Mittal, S.; C. Dym; M. Morjaria. 1986. *"PRIDE: An Expert System for Design of Paper Handling Systems"*. Computer, vol. 6.

[122] - Moser, J. G. 1986. *"Integration of Artificial Intelligence and Simulation in a Comprehensive Decision-Support System"*. Simulation, vol. 47 nº6, 223-229.

[123] - Murchio, P.; et alli. 1988. *"Experiments and Demonstrations Scenario of the CIRCE Center for System Integratio in CIM"*. In ESPRIT '88 - Putting the Technology to Use. Proceedings of the 5th Annual ESPRIT Conference, 1545-1560.

[124] - Murray, K. J.; S. V. Sheppard. 1988. *"Knowledge-Based Simulation Model Specification"*. Simulation, vol. 50 nº3, 112-119.

[125] - Murthy, S.; S. Addanki. 1987. *"PROMPT: An Innovative Design Tool"*. Proceedings of the 7th AAAI Conference, 637-642.

[126] - Nau, D. S.; T. C. Chang. 1985. *"A Knowledge-Based Approach to Generative Process Planning"*. In Computer-Aided Intelligent Process Planning, A.S.M.E., C. R. Liu, T. C. Chang, R. Komanduri - eds, 65-71.

[127] - O'Keefe, R. 1986. *"Simulation and Expert Systems - A Taxonomy and Some Examples"*. Simulation, vol. 46 nº1, 10-16.

[128] - Ostroff, J. S. 1989. *"Synthesis of Controllers for Real-Time Discrete Event Sistems"*. In Proceedings of the 28th Conference on Decision and Control, Tampa-FA.

[129] - Ostroff, J. S.; 1989. *"Temporal Logic for Real-Time Systems"*. Advanced Software Development Series, Press Limited.

[130] - Ostroff, J. S.; W. M. Wonham. 1990. *"A Framework for Real-Time Discrete Event Control"*. IEEE Transactions on Automatic Control, vol. 35 nº4, 386-397.

[131] - Partsch, H; R. Steinbruggen. 1983. *"Program Transformation Systems"*. Computing Surveys, vol. 15 nº3, 199-236.

[132] - Pegden, C. D. 1985. *"Introduction to SIMAN"*. Systems Modeling Corporation, State College, Pa.

[133] - Phillips, R. H.; X. D. Zhou, C. B. Mouleeswaran. 1984. *"An Artificial Intelligence Approach to Integrating CAD and CAM Through Generative Process Planning"*. In Proceedings of the 4th ASME International Computers in Engineering Conference, vol. 2, 459-463.

[134] - Pnueli, A. 1981. *"The Temporal Semantics of Concurrent Programs"*. In Theoretical Computer Science, vol 13, 45-60, North Holland.

[135] - Pnueli, A. 1986. *"Applications of Temporal Logic to the Specification of Reactive Systems. A Survey of Current Trends"*. Lecture Notes on Computer Science, vol. 224.

[136] - Pressman, R. S. 1987. *"Software Engineering. A Practitioner's Approach"*. McGraw Hill, New York.

[137] - Pritsker, A. A. B. 1984. *"Introduction to Simulation and SLAM"*. Halsted Press, New York, 2nd edition.

[138] - Pruett, J. M.; V. K. Vasudev. 1990. *"MOSES: Manufacturing Organization Simulation and Evaluation System"*. Simulation, vol. 54 nº1, 37-45.

[139] - Quade, E. S. 1985. *"Modeling Techniques"*. In Handbook of Systems Analysis, H. J. Miser, E. S. Quade - ed, North Holland, Amsterdam.

[140] - Ranson, G. M. 1972. *"Group Technology"*. McGraw Hill, London.

[141] - Reisch, D. 1987. *"Total CIM Concept Embracing Logistics"*. Robotics and Computer-Integrated Manufacturing, vol. 3 nº1, 105-122.

[142] - Rembold, U.; C. Blume; R. Dillmann. 1985. *"Computer Integrated Manufacturing Technology and Systems"*. Marcel Dekker, New York.

[143] - Ruiz-Mier, S.; J. Talavage. 1987. *"A Hybrid Paradigm for Modeling of Complex Systems"*. Simulation, vol. 48 nº4, 135-141.

[144] - Sabuncuoglu, I; D. Hommertzheim. 1988. *"Expert Systems and Simulation in Flexible Manufacturing Systems"*. In Proceedings of 10th Annual Conference on Computers and Industrial Engineering. Computers and Industrial Engineering, vol. 15 nº1-4, 1-7.

[145] - Sarin, S. C.; R. R. Salgame. 1990. *"Development of a Knowledge-Based System for Dynamic Scheduling"*. International Journal of Production Research, vol. 28 nº8, 1499-1512.

[146] - Sarker, B. R. 1989. *"Simulating a Just-In-Time Production System"*. Computers and Industrial Engineering, vol. 16, nº1, 127-137.

[147] - Schaffer, G. 1980. *"GT Via Automated Process Planning"*. American Machinist, nº5, 119-122.

[148] - Schriber, T. J. 1974. In *"Simulation Using GPSS"*. Wiley, New York.

[149] - Schroer, B. J. 1989. *"A Simulation Assistant for Modeling Manufacturing Systems"*. Simulation, vol. 53 nº5, 201-206.

[150] - Schroer, B. J. 1989. *"Improving the Manufacturing Simulation Modeling Environment"*. Manufacturing Review, vol. 2 nº4, 283-289.

[151] - Schroer, B. J.; F. T. Tseng. 1988. *"Modeling Complex Manufacturing Systems Using Discrete Event Simulation"*. Computers and Industrial Engineering, vol. 14 nº4, 455-464.

[152] - Schroer, B. J.; F. T. Tseng. 1989. *"An Intelligent Assistant for Manufacturing System Simulation"*. International Journal of Production Research, vol. 27 nº10, 1665-1683.

[153] - Seifoddini, H; P. M. Wolfe. 1986. *"Application of the Similarity Coefficient Method in Group Technology"*. IIE Transactions, vol. 18 nº3, 271-277.

[154] - Seila, A. F. 1988. *"SIMTOOLS: A Software Tool Kit for Discrete Event Simulation in Pascal"*. Simulation, vol. 50 nº3, 93-99.

[155] - Shannon, R. E. 1975. *"System Simulation: The Art and Science"*. Prentice Hall, New Jersey.

[156] - Shannon, R. E. 1988. *"Knowledge Based Simulation Techniques for Manufacturing"*. International Journal of Production Research, vol. 26 nº5, 953-973.

[157] - Shannon, R. E.; R. Mayer; H. H. Adelsberger. 1985. *"Expert Systems and Simulation"*. Simulation, vol. 44 nº6, 275-284.

[158] - Shapiro, S. F. 1986. *"CIM: The Right Medicine for the Electronic Manufacturing ? "*. Computer Design, Nov.,73-88.

[159] - Shapiro, E. 1986. *"The Art of PROLOG"*. The MIT Press.

[160] - Shaw, M. J.; A. B. Whinston. 1989. *"An Artificial Intelligence Approach to the Scheduling of Flexible Manufacturing Systems"*. IIE Transactions, vol. 21 nº2, 170-183.

[161] - Signoretti, A. 1990. *"SMAC - Sistema de Manipulação e Armazenagem de Conhecimento"*. Tese de Mestrado, DCA - FEE - UNICAMP.

[162] - Signoretti, A.; M. Rezende; F. Gomide; S. Bingulac. 1989. *"Implementation of Knowledge Based Environment for Computer Aided Control Engineering"*. IEEE National Aerospace and Electronics Conference - NAECON, vol. 1.

[163] - Silva Jr., B. I. 1992. *"Lógica Temporal de Tempo Real Generalizada"*. Tese de Mestrado, DCA - FEE - UNICAMP.

[164] - Sistla, A. P. 1984. *"Can Message Buffers Be Axiomatized in Linear Temporal Logic?"*. Information and Control, nº63.

[165] - Smith, R. L.; L.Platt. 1987. *"Benefits of Animation in Simulation of Machining and Assembly Line"*. Simulation, vol. 48 nº1, 28-30.

[166] - Srihari, K.; T. J. Greene. 1988. *"Alternate Routings in CAPP Implementation in a FMS"*. In Proceedings of 10th Annual Conference on Computers and Industrial Engineering. Computers and Industrial Engineering, vol. 15 nº1-4, 41-50.

[167] - Stallings. 1984. *"Local Area Networks"*. ACM - Computing Survey, vol. 16 nº1.

[168] - Standridge, C. R. 1985. *"Performing Simulation Projects with The Extended Simulation System (TESS)"*. Simulation, vol. 46 nº6, 283-291.

[169] - Stylianides, C. 1987. *"Use of Simulation in the Analysis of Shop Floor Operations"*. Computers and Industrial Engineering, vol. 13 nº1-4, 144-148.

[170] - Takahashi, T.; H. K. E. Liesenberg. 1990. *"Programação Orientada a Objetos"*. VII Escola de Computação, IME - USP.

[171] - Thistle, J. G.; W. M. Wonham. 1986. *"Control Problems in a Temporal Logic Framework"*. International Journal of Control, vol.44 nº4, 943-976.

[172] - Tocher, K. D. 1963. *"The Art of Simulation"*. The English Universities Press, London.

[173] - Turner, D. H. 1986. *"Manufacturing Simulation Comes To Age"*. CIM Technology, vol. 5, 16-17.

[174] - Vannelli, A.; K. R. Kumar. 1986. *"A Method for Finding Minimal Bottle-Neck Cells for Grouping Part-Machine Families"*. International Journal of Production Research, vol. 24 nº2, 387-400.

[175] - Wang, H-P (Ben). 1988. *"A Layered Architecture for Manufacturing Operation Planning"*. Computers and Industrial Engineering, vol. 14 nº2, 201-210.

[176] - Wang, Q.; J. Zhou; J. Yu. 1988. *"A Chain-Drive Design Expert System and CAD SYstem"*. In Expert Systems in Engineering, D. T. Pham - ed, IFS Publications, Springer Verlag, New York.

[177] - Widman, L. E.; K. A. Loparo. 1989. *"Artificial Intelligence, Simulation, and Modeling: A Critical Survey"*. In Artificial Intelligence Simulation and Modeling, L. E. Widman, K. A. Loparo, N. R. Nielsen - eds, John Wiley and Sons, New York.

[178] - Wolper, P. 1983. *"Temporal Logic Can Be More Expressive"*. Information and Control, nº56, 72-79.

[179] - Young, R. E.; M. A. Rossi. 1988. *"Toward Knowledge-Based Control of Flexible Manufacturing Systems"*. IIE Transactions, vol. 20 nº1, 36-41.

[180] - Zeigler, B. P. 1987. *"Hierarchical, Modular Discrete-Event Modeling in an Object-Oriented Environment"*. Simulation, vol. 49 nº5, 219-230.