Universidade Federal de Santa Catarina
Departamento de Engenharia de Produção e Sistemas
Programa de Pós-Graduação em Engenharia de Produção

Uma Abordagem Distribuída no Desenvolvimento e Implementação do Software de Controle de Chão-de-Fábrica em Sistemas de Manufatura Celular

Tese Submetida à Universidade Federal de Santa Catarina para a Obtenção do Título de Doutor em Engenharia de Produção

Luis Fernando Friedrich

Florianópolis, Fevereiro de 1996

UMA ABORDAGEM DISTRIBUÍDA NO DESENVOLVIMENTO DO SOFTWARE DE CONTROLE DE CHÃO-DE-FÁBRICA EM SISTEMAS DE MANUFATURA CELULAR

Luis Fernando Friedrich

Esta tese foi julgada adequada para a obtenção do título de
DOUTOR EM ENGENHARIA DE PRODUÇÃO.

Prof. Ricardo Miranda Barcia Ph.D.
Coordenador do Curso

BANCA EXAMINADORA:

Prof. Plínio Stange, PhD
(Orientador)

Prof. Osama K. Eyada, PhD
Examinador Externo

Prof. Simão S. Toscani, Dr.
Examinador Externo

Prof. Miguel Fiod, Dr.

Prof. Fernando Álvaro Ostuni Gauthier, Dr.

Rogério Bastos, Dr. (Moderador)

A Bi,
Fernanda,
Duda e
Luciano

AGRADECIMENTOS

À minha família pela paciência e incentivo durante todo o tempo.

À Universidade Federal de Santa Catarina (UFSC) e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pela ajuda financeira durante o período de estudos nos Estados Unidos da América.

Ao Departamento de Informática e Estatístca (INE) da UFSC por acreditar e permitir a realização deste trabalho.

Ao Curso de Pós Graduação em Engenharia da Produção da UFSC por fornecer os meios necessários para a realização deste trabalho.

Ao professor Plínio Stange pela sua orientação segura e correta sem a qual seria difícil terminar o trabalho.

Ao professor Osama K. Eyada pela sua valiosa orientação durante o período de estudos na Virgínia Polytechnic Institute and State University.

Aos amigos e colegas, em especial o Thadeu e o Mazzucco, pelo incentivo e apoio durante as várias fases deste trabalho.

RESUMO

O uso das células flexíveis de manufatura (FMC) tem representado para a indústria uma estratégia efetiva no sentido de responder rapidamente às mudanças impostas pelo mercado. Considerando que o objetivo das células de manufatura é a possibilidade de rápida adaptação às mudanças mercadológicas, o software de controle (que é o que realmente controla a célula) deve estar habilitado a receber as modificações necessárias para atender as constantes mudanças de requisitos. Entretanto, a falta de sistemas de desenvolvimento de software adequados ao controle de células de manufatura faz com que o desenvolvimento de sistemas de controle de chão-de-fábrica (células e centros de trabalho) represente hoje um dos maiores problemas na implantação da manufatura celular flexível. Uma boa parcela desta dificuldade pode ser atribuída aos altos custos envolvidos no desenvolvimento e manutenção do software de controle e à dificuldade de se conseguir definir a forma de integração dos sistemas de chão-de-fábrica.

O objetivo principal da abordagem proposta neste trabalho é oferecer uma estrutura para o projeto do software de controle de chão-de-fábrica em sistemas de manufatura celular (modelo distribuído), e um ambiente de suporte para a sua implementação. Desta forma a geração do controle para sistemas específicos será facilitada com base em estruturas e procedimentos genéricos que caracterizam os componentes do sistema. Para isto é adotado um modelo de controle onde os componentes se comunicam para realizar as tarefas. Além disto é apresentado um modelo de implementação que atende e suporta as características do modelo de controle. Um protótipo do ambiente de implementação é mostrando com o objetivo de validar as características de configurabilidade e extensibilidade da abordagem proposta.

ABSTRACT

Flexible Manufacturing Cells (FMC's) are today a good way for manufacturing industry to deal with changes wich are imposed by the market. Considering that manufacturing cells are capable of dealing with these changes, the control software (wich is actually what controls the cell) must be able to change following the requirements. However, most of software development systems do not have the proper capabilities to deal with the manufacturing cell requirements. Thus, the development of shop floor control systems is one of the major problems for implementation of flexible manufacturing cells. Part of the problems are assigned to the high software development costs and the difficulty in defining how the integration between shop floor systems (equipment) would be.

In this work we propose an approach providing a framework for shop floor control software development and implementation. The approach is based on a distributed model. Thus, the control software that will be used in order to control real systems is implemented by using generic structures and procedures that caracterize the components of the system. In order to do that, we use a control model where the components communicate each other to perform the tasks. In addition, an implementation model is proposed in order to support the components and the commnunication between them. A prototype, which includes a set of functions that support the implementation model, is implemented. Also, the prototype has a user interface that gives the user the capabilities to configurate the system and to specify the control components.

ÍNDICE

LISTA DE FIGURAS

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

CAPÍTULO 1
INTRODUÇÃO

A indústria manufatureira tem uma participação muito importante na economia brasileira. Ela é responsável por aproximadamente 40% do produto interno bruto (PIB) e tem uma participação de 30% nas exportações [BRA91]. A busca de maior integração e participação no mercado mundial tem sido uma tendência da indústria nacional. Assim sendo, a indústria tem procurado melhorar atividades como projeto de produto, planejamento de fabricação, processos de fabricação, controle da fabricação e distribuição de produtos, objetivando a conquista e manutenção de mercados. Neste quadro, os sistemas flexíveis de manufatura (FMS's - Flexible Manufacturing Systems) têm recebido grande atenção como sendo uma estratégia efetiva no sentido de as indústrias responderem mais rapidamente às mudanças impostas pelo mercado. FMS promete muitos benefícios, incluindo aumento da utilização das máquinas, estoques intermediários reduzidos, número de máquinas reduzido, qualidade do produto mais consistente, espaços menores, e redução dos tempos de preparação [PAL87] [GRE86]. Entretanto, a implementação destes sistemas é bastante complexa. Com o objetivo de gerenciar a dificuldade e complexidade do desenvolvimento de FMS, estes sistemas são decompostos em diferentes tipos de células flexíveis de manufatura (FMC - Flexible Manufacturing Cell). Geralmente, uma FMC é um grupo de uma ou mais máquinas-ferramenta, robôs, e um sistema de manuseio interno de material. Um dos principais componentes das FMC's é o controlador, ou sistema de controle da célula (SCC). Considerando que o objetivo das células de manufatura é a possibilidade de rápida adaptação às mudanças mercadológicas, o software de controle (que é o que realmente controla a célula) deve estar habilitado a receber as modificações necessárias para atender as constantes mudanças de requisitos. Entretanto, os métodos tradicionais de desenvolvimento de software de controle não têm

conseguido atender os requisitos destes sistemas de maneira eficiente e produtiva. Acredita-se que a falta de sistemas de desenvolvimento de software adequados ao controle de células de manufatura representa uma das maiores restrições para o crescimento da automação fabril [LAR89]. Nesta pesquisa, a meta é a definição de uma estrutura para o desenvolvimento do software de controle de células/centros de trabalho e um ambiente de suporte para sua implementação.

O restante deste capítulo apresenta uma caracterização dos sistemas de manufatura celular e aspectos relevantes ao controle destes sistemas. Em seguida são apresentados alguns problemas que devem ser resolvidos com respeito ao controle de sistemas de manufatura celular e ao desenvolvimento do software de controle que constituem a motivação deste trabalho e estabelecem os objetivos da pesquisa.

1.1 Manufatura Celular - uma caracterização

De um modo geral um sistema de manufatura (SM) é uma coleção de dispositivos físicos, computadores e pessoas, que de forma cooperativa realizam algum processo de manufatura. A complexidade de um sistema de manufatura varia desde simples máquinas e ferramentas operadas manualmente até sofisticados sistemas de manufatura integrada por computador (CIM-Computer Integrated Manufacturing).

A manufatura discreta de peças pode ser classificada segundo a atividade de produção (quantidade de produto fabricado) e em função do *layout* da fábrica [GRO90]. Considerando a atividade de produção, existem três tipos principais de sistemas de produção: pequena produção (*Job shop*), produção em lotes (*Batch*), linha de produção (*Mass*).

O tipo *Job shop* é caracterizado pelo baixo volume de produção e pela grande variação de peças fabricadas, por exemplo, usualmente o tamanho do lote de manufatura é pequeno (uma peça). A produção em linha, *Mass production*, é caracterizada pelas altas taxas de produção, com equipamentos que são completamente dedicados à produção de um produto em particular; a fábrica normalmente é projetada com o propósito exclusivo de fabricar um produto específico. A produção em lotes, *Batch*, é caracterizada pela fabricação de lotes de tamanho médio de forma contínua ou intercalada com o propósito de satisfazer a demanda do usuário por um determinado produto. Nos EUA, estima-se que 75% da produção na indústria de peças discretas deriva da produção em lotes de aproximadamente 50 peças [GRO90]. Desta forma, a produção em lotes representa uma parte significativa na atividade de manufatura.

Por outro lado, a classificação em função do *layout* da fábrica refere-se ao arranjo físico dos equipamentos de fabricação. Existem três tipos principais de layout de fábrica associados aos sistemas de produção de peças discretas: posição fixa (*fixed-position layout*), orientado a processo (*process layout*) e orientado ao fluxo do produto (*product-flow layout*). No primeiro tipo o termo posição fixa se refere ao produto. Em função do tamanho e do peso do produto, o mesmo permanece em um local e os equipamentos usados na sua fabricação são trazidos até ele. Usualmente este tipo de *layout* é associado a sistemas de produção do tipo *Job shop*. No *layout* orientado a processo, as máquinas de produção são arranjadas em grupos de acordo com o tipo geral de processo de manufatura. Por exemplo, os tornos estão em um departamento, as furadeiras em outro e equipamentos de acabamento em outro. Este tipo de *layout* é comum em sistemas de produção do tipo *Job shop* e *Batch*. Finalmente, quando a fábrica é especializada para a produção de um, ou uma classe de produtos em grande escala, os equipamentos são arranjados da

forma mais eficiente para a produção do mesmo. Este tipo de *layout* normalmente é associado a sistemas de produção do tipo *Mass production.*

A manufatura baseada em células (manufatura celular), representa uma tentativa de combinar a eficiência do *layout* orientado a fluxo (*product-flow layout*) com a flexibilidade do *layout* orientado a processo (*process layout*) em sistemas de produção em *Batch.* Na manufatura celular, o sistema de manufatura é decomposto em um conjunto de centros de trabalho ou células. Cada centro de trabalho é uma coleção de equipamentos e processos dedicados para o atendimento dos requisitos de processamento de uma família de peças (peças com requisitos de fabricação similares). A Figura 1.1 mostra a estratégia de manufatura tradicional e a celular. Como resultado da utilização da organização celular, os tempos de transporte de material podem ser significativamente reduzidos. Na busca de flexibilidade e automação, a manufatura celular utiliza equipamentos tais como robôs, máquinas-ferramenta numéricamente controladas, sistemas automáticos de inspeção e sistemas de transporte de material.

No chão-de-fábrica, algumas funções básicas devem ser realizadas para que o seu objetivo principal seja atingido, ou seja, a transformação de matéria prima em produto final. Estas funções envolvem *processamento, montagem, armazenamento e manuseio de material, inspeção e teste,* e *controle.* As primeiras quatro funções dizem respeito às atividades físicas que relacionam-se diretamente com o produto sendo fabricado. A função de controle é necessária para coordenar e regular as atividades físicas dos vários dispositivos que existem no chão-de-fábrica.

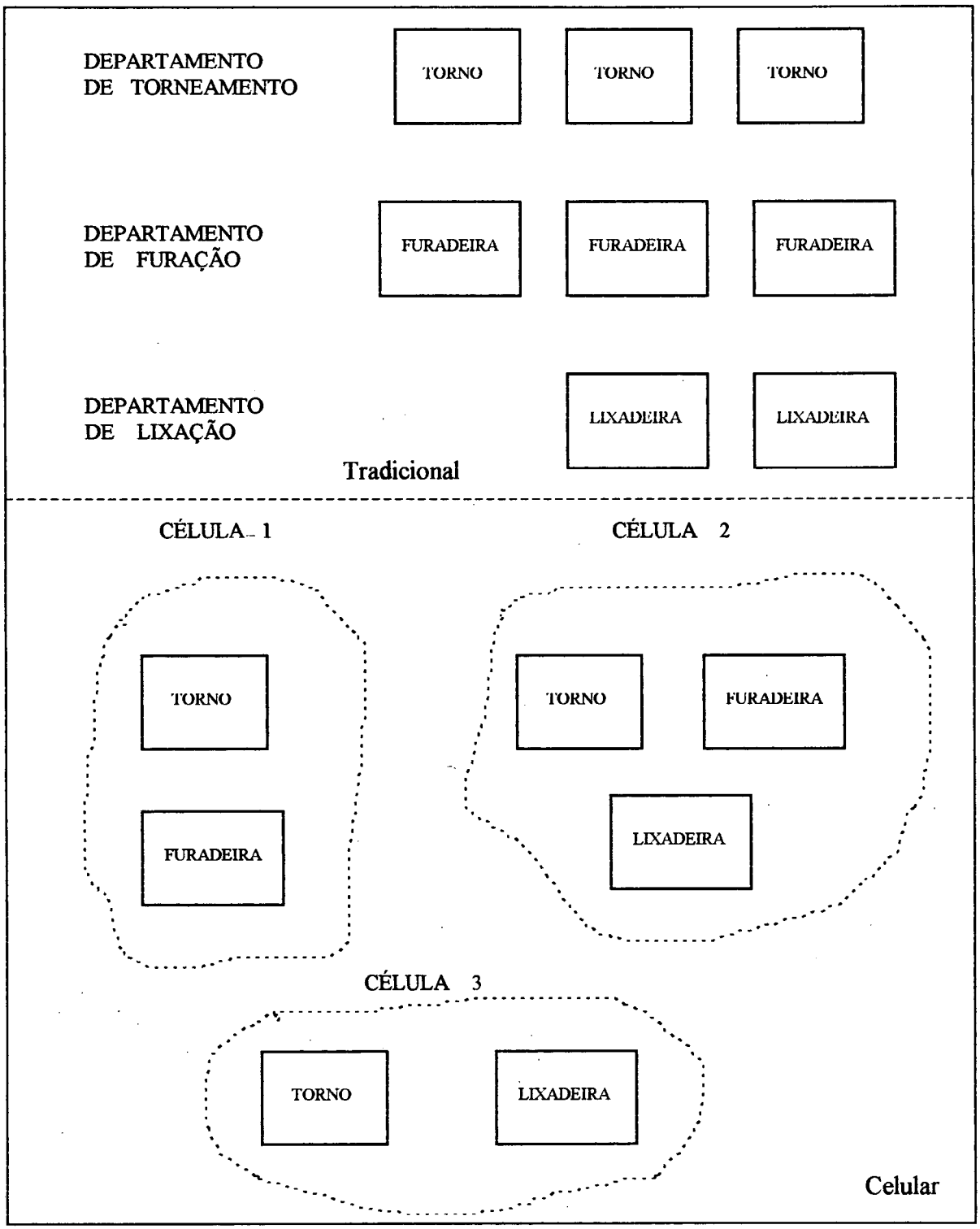


Figura 1.1 : Estratégias de Manufatura - Tradicional e Celular.
Fonte: Adaptação de [VIT94]

A funcionalidade e definição exatas de uma célula de trabalho vem sofrendo evoluções, mas a idéia fundamental de célula é a de controlar e coordenar um conjunto relativamente pequeno de dispositivos em um ambiente de produção. Os dispositivos são agrupados de forma que possam, sob o controle de um computador, configurar-se para a produção de diferentes peças. A figura 1.2 mostra um exemplo de um sistema de manufatura a nível de chão-de-fábrica utilizando o conceito de células ou centros de trabalho. Neste caso, as células são formadas por equipamentos de manufatura com capacidade de processar, armazenar, transportar e montar peças.

1.2 O Controle em Manufatura Celular

O controle em sistemas de manufatura celular usualmente é exercido nos seguintes níveis de controle : *nível de controle local*, *nível de controle de supervisão*, *nível de planejamento*.

O *controle local* envolve um processo interativo entre um controlador e uma máquina. Este processo abrange todas as funções necessárias para o controle da máquina. O processo envolve o recebimento de comandos de outras entidades (controlador) e o posterior envio de comandos específicos para as máquinas.

O *controle de supervisão* envolve um processo interativo entre o controlador, ou operadores humanos e um grupo de máquinas. O objetivo do controle é mover as peças de uma máquina para outra em uma sequência ótima de processamento executando o plano de processo de cada peça. O processo abrange as funções necessárias para o controle da produção de um tipo de produto.

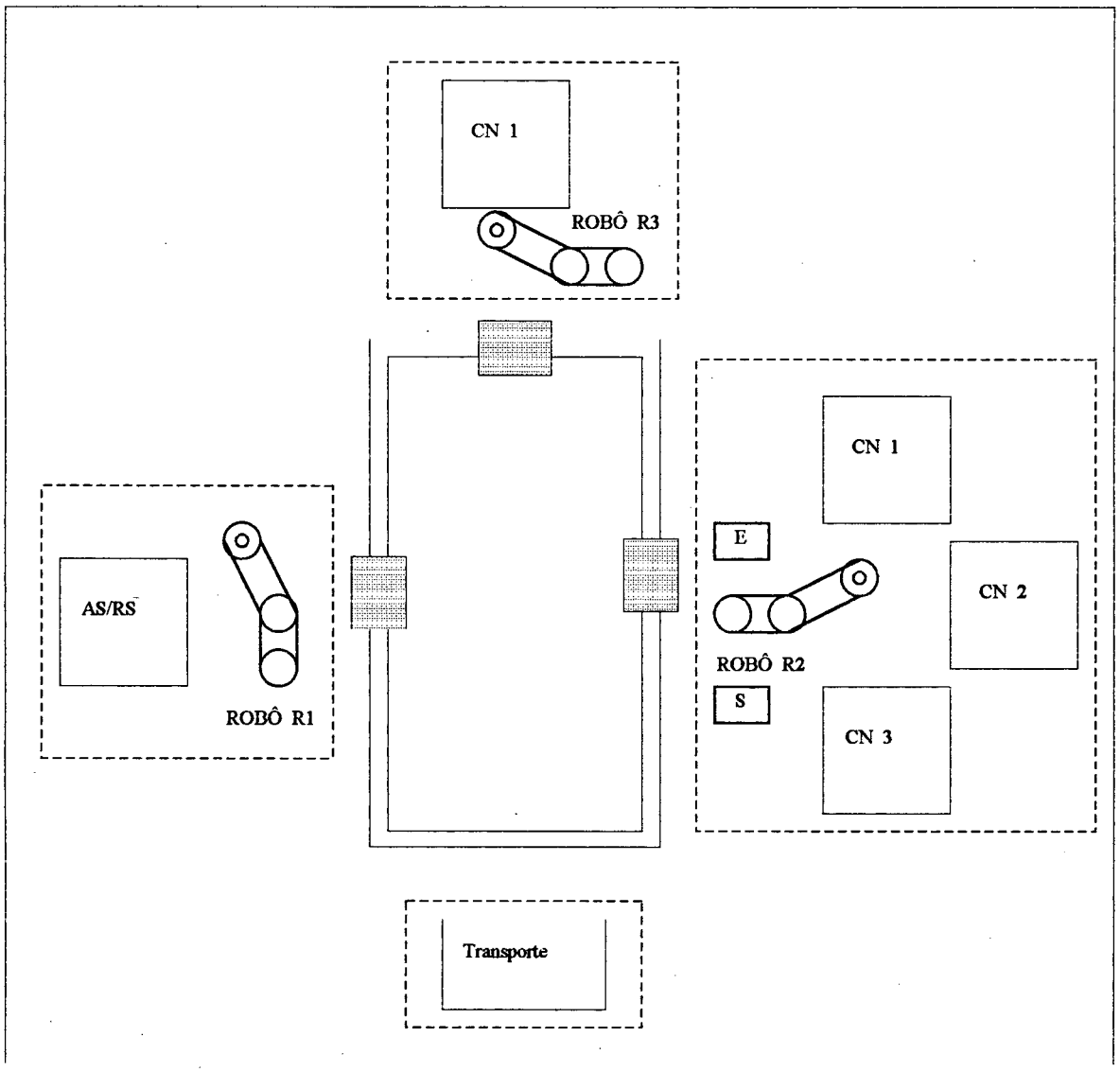


Figura 1.2 : Exemplo de um sistema de manufatura celular (chão-de-fábrica).

O *nível de planejamento* é a conexão de todos os controles de supervisão envolvidos na fabricação do mesmo produto. Funções deste nível envolvem cálculo de capacidades das máquinas e planejamento de produção.

O termo **controlador** é usado de uma forma genérica para descrever o hardware, software, e outros componentes que influenciam o comportamento do chão-de-fábrica. Alguns controladores são compostos apenas por componentes de hardware, como por exemplo um conjunto de botões, motores e chaves elétricas (relés). Outros são caracterizados por apresentarem componentes de hardware e software. Neste caso, o hardware (Controlador Lógico Programável-CLP, computador) fornece uma interface para o sistema físico (sensores, atuadores) e um ambiente para a execução do software. O software usado para o controle pode ser representado por vários componentes distintos. O software de controle que influencia o fluxo de peças e o fluxo de informações, o software de sequenciamento, o software de planejamento, o software de gerenciamento e a aquisição de dados.

Muitas vezes o termo *controle* confunde os elementos citados acima em um único elemento de software. Neste trabalho o controle (software de controle) é o componente responsável por gerenciar/causar : as ações físicas, o fluxo de sinais e mensagens que iniciam, ou requisitam, as ações físicas e as interações com o sequenciador. O software de controle também fornece os *conectores* para a integração com o sequenciador e outros softwares. Desta forma, o controle é visto como sendo um componente situado entre o escalonamento e a interface de comunicação. Um dos aspectos importantes desta visão é o fato de separar o sequenciador das atividades diretamente envolvidas com a execução.

Em termos de funcionalidade, o controlador de chão-de-fábrica apresenta os seguintes requisitos :

Comunicação : A comunicação é necessária para que o controle de chão-de-fábrica seja capaz de executar suas tarefas de controle e troca de informações com o ambiente externo e os dispositivos (entidades) por ele controlados. As comunicações podem ser suportadas por diferentes métodos, tais como: redes de computadores e ligações ponto-a-ponto. Visto que a comunicação é um requisito importante, o método utilizado deveria ser padronizado. O suporte desta função inclui o meio de comunicação (hardware) e o protocolo desta comunicação (software). De acordo com DiCesare *et alli* [DIC86], a comunicação representa o *elo* que mantém conectados os sistemas de automação.

Coordenação e supervisão : Estas funções dizem respeito ao controle que um controlador de chão-de-fábrica tem sobre os dispositivos de produção. Estas funções também envolvem o escalonamento (sequenciamento) de tarefas, inicialização e desativação de dispositivos. O controle de chão-de-fábrica apresenta três componentes básicos: planejamento, escalonamento e execução, como mostra a Figura 1.3 [JOS90]. A função de *planejamento* recebe um conjunto de ordens de serviço e seus respectivos planos de processo e seleciona um conjunto de tarefas para serem escalonadas com base no impacto que cada ordem e plano tem em termos de performance do sistema. O planejamento é ativado pela função de execução, quando uma nova peça chega no chão-de-fábrica ou, pela função de escalonamento, quando é necessário replanejar. A função de *escalonamento* é responsável pelo sequenciamento dos lotes/planos de processo em função dos recursos existentes. Em tempo real, esta função é ativada quando uma atividade de planejamento em uma nova peça é terminada, quando uma operação é terminada em uma máquina-ferramenta ou quando um dispositivo

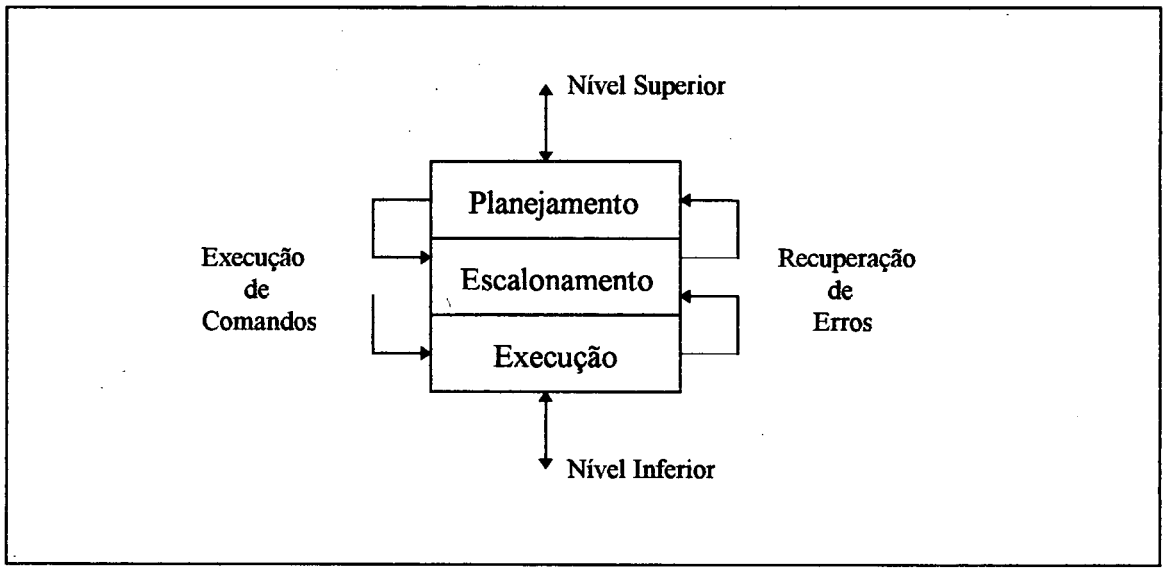


Figura 1.3 : Funções de supervisão e controle de um controlador.
Fonte : Adaptação de [JOS90]

torna-se livre. A saída da função de escalonamento para a função de execução é um conjunto de mensagens dizendo que ações devem ser feitas para cada evento. A função de execução/controle fornece os mecanismos para a execução das ações e gerencia o fluxo de informações/peças entre os computadores/máquinas. Esta função trabalha como coordenadora das outras funções e ponto de comunicação com os outros controladores [CHO93] [JOS90].

Recuperação e armazenamento de informações : Dados e informações são necessários para o funcionamento apropriado do sistema que está sendo controlado. Estas informações ou dados podem ser de diferentes tipos, por exemplo: programas necessários para os diversos controladores de dispositivo, parâmetros das peças, informação do estado dos dispositivos, etc., e o gerenciamento destas informações é responsabilidade do controlador.

O controlador deve estar habilitado a manter as informações necessárias à realização das tarefas. O envio/retirada de programas para/de dispositivos, considerado uma função de controle, é uma função fortemente associada ao gerenciamento de informações. A aquisição e análise de dados também pode ser considerada como função do controlador; estes dados podem incluir informações sobre os produtos, qualidade, produção e processos, e deveriam estar disponíveis para visualização.

1.3 Situação Atual do Controle de Chão-de-Fábrica

Um sistema de controle de chão-de-fábrica, como visto anteriormente, é composto por componentes de hardware e software. O hardware pode estar na forma de robôs, máquinas ferramentas, dispositivos de manuseio de material, cotroladores lógicos programáveis, e computadores. O software inclui programas de controle em tempo real, programas para escalonamento, programas de controle numérico, e

programas para robôs. Estes dois componentes, software e hardware, são importantes na busca de uma nova geração de controladores e na melhoria do seu desempenho no chão-de-fábrica.

Controladores Lógico Programáveis (CLP's) têm sido usados como elemento controlador de chão-de-fábrica na maioria das aplicações automatizadas [LAR89]. Entretanto, devido à grande quantidade e diversidade de informações necessárias nos sistemas de manufatura modernos e à complexidade e limitação da forma de programação dos CLP's, surgiu a necessidade de busca por novas alternativas de controle. A utilização de computadores de propósito geral para controlar sistemas de manufatura tem crescido à medida que o preço dos computadores tem decrescido. Cada vez mais o controle de chão-de-fábrica está utilizando estações de controle computadorizadas. Entre as soluções utilizadas encontram-se computadores pessoais (PC), supermicros (VAX), superminis e computadores especiais (Vista 2000) [JAS87]. Acredita-se que a disponibilidade de uma nova geração de controladores de chão-de-fábrica poderia ter um efeito significativo na melhoria deste tipo de controle [LAR89]. Esta nova geração de controladores deveria apresentar características de : interconectividade e funcionalidade dos computadores de propósito geral. O componente software tem uma participação muito importante na efetivação dessa melhoria.

O desenvolvimento de sistemas de controle de chão-de-fábrica representa hoje um dos maiores problemas na criação de indústrias de manufatura flexível automatizadas ou semi-automatizadas. Uma boa parcela desta dificuldade pode ser atribuída aos altos custos envolvidos no desenvolvimento e manutenção do software de controle e à dificuldade de se conseguir definir a forma de integração dos sistemas de chão-de-fábrica [SMI92]. Os dispositivos de produção, como robôs, máquinas ferramentas e outros equipamentos de produção, assim como os

computadores e redes de comunicação, geralmente são encontrados com facilidade. Entretanto, o *software de controle* necessário para a implementação de um sistema de controle flexível e integrado destes equipamentos não está prontamente disponível. Estes sistemas normalmente são multidisciplinares envolvendo conhecimento não só em manufatura mas também em programação de computadores, análise e especificação de sistemas, e redes de computadores. Por exemplo, estima-se que o custo de desenvolvimento do software no caso de sistemas flexíveis de manufatura gira em torno de 40% a 60% do seu custo total [AYR89].

O software de controle, considerado como o *kernel* das estações de controle (controladores), tem sido desenvolvido de forma altamente específica e necessita mudar sua forma de desenvolvimento para atender as rápidas mudanças que ocorrem em termos de requisitos e ambiente nestes sistemas. Atualmente existem três opções para a obtenção do software de controle: comprar do fornecedor do sistema de controle, utilizar consultores ou desenvolver o seu próprio software de controle [LIN94]. A compra de pacotes prontos implica normalmente na necessidade de adaptação dos mesmos à aplicação desejada, sendo que esta adaptação é dificultada pela ausência de padrões de interface e funções nos pacotes. A utilização de consultores demanda a seleção de consultores qualificados, a preparação de um contrato e a monitoração do projeto. A alternativa de desenvolver o software na própria empresa esbarra na falta de especialistas em engenharia de software capazes de tratar um projeto complexo. Tanto a compra de pacotes quanto a utilização de consultores representam custos altos sendo que somente as grandes empresas têm condições de investir. Neste caso, a definição de uma estrutura que auxilie no desenvolvimento do software para sistemas de controle de chão-de-fábrica pode representar uma boa possibilidade para as pequenas e médias empresas, as quais passariam a poder desenvolver seus próprios sistemas de controle de chão-de-fábrica.

Grande parte dos problemas encontrados no desenvolvimento de software para sistemas de controle decorrem da falta de um modelo (arquitetura) adequado para tratar as necessidades envolvidas na geração de software de controle genérico para sistema de chão-de-fábrica. Por exemplo, os esquemas de controle são predominantemente proprietários e inflexíveis, arquiteturas são baseadas principalmente em monoprocessadores dedicados causando grandes problemas de integração e re-projeto quando da necessidade de adicionar novos elementos. Diante desse quadro, arquiteturas de controle têm sido desenvolvidas na busca de estabelecer uma estrutura padrão para sistemas de manufatura controlados por computador e reduzir os custos de desenvolvimento do software [McL86] [MEY86] [BOU91].

Os modelos adotados incluem uma abordagem hierárquica sequencial e centralizada para o processo de desenvolvimento do sistema [DUA93]. Considerando a natureza hierárquica dos grandes sistemas, o modelo convencional de construção do controle envolve o uso de uma abordagem *top-down* centralizada onde, primeiro se define a estrutura global do sistema e, em seguida, os componentes são refinados. Embora esta abordagem forneça uma ferramenta de representação útil para muitos sistemas, ela apresenta alguns problemas no tratamento das características dinâmicas de um sistema. Primeiro, o modelo centralizado não é o mais adequado para representar atividades paralelas e simultâneas dos sistemas de chão-de-fábrica devido às restrições que o modelo impõe no que diz respeito à autonomia das entidades *controladas*. Segundo, visto que a centralização do controle exige a participação da entidade controladora em cada decisão, o sistema torna-se menos flexível. Terceiro, a centralização do controle faz com que o sistema de controle seja mais dependente de aplicação e ajustado para configurações específicas, não fornecendo meios para uma rápida

reconfiguração do sistema. Na necessidade de mudanças envolvendo as máquinas ou os requisitos de produção, o modelo de controle perde a sua validade e consequentemente o software tem de ser reconstruído. Esta limitação implica em custos adicionais na modelagem do sistema assim como no desenvolvimento do software.

1.4 Motivação e Objetivos da Pesquisa

Existem boas chances que a utilização de uma abordagem *bottom-up* distribuída possa significar um avanço na tentativa de estabelecer um modelo de controle genérico e reconfigurável. A diferença fundamental entre os modelos centralizado e distribuído é que o primeiro enfatiza a responsabilidade do controlador (central) na inicialização e controle de todas as tarefas dos *controlados* enquanto o segundo se preocupa mais com a coordenação e cooperação entre os *controlados*. No modelo distribuído, a representação de um sistema pode ser feita através da identificação de dois tipos de entidades : *componentes* e *conexões*. Os componentes caracterizam os elementos básicos a partir dos quais os sistemas são compostos e apresentam a característica de serem independentes de aplicação, ou seja, um componente pode ser usado em diferentes sistemas. As conexões estabelecem a parte dependente de aplicação. Quando estabelecemos as conexões de um sistema é necessária a definição dos componentes. Desta forma, a abordagem distribuída requer primeiro a identificação dos componentes e depois a definição dos relacionamentos entre os componentes (abordagem *bottom-up*). Esta abordagem apresenta algumas vantagens:

- Considerando que cada componente funciona de forma autônoma, atividades paralelas e simultâneas podem ser descritas no modelo.

- Um sistema pode ser modelado de forma modular considerando que os componentes são definidos de forma independente da estrutura global do sistema. Assim, um subsistema pode ser construído a partir do agrupamento de um conjunto de componentes em uma entidade.

- Considerando que a estrutura global de um sistema depende apenas da especificação das *conexões* , a reconfiguração de um sistema pode ser feita a partir da redefinição das *conexões*.

- Como as atividades de controle preocupam-se mais com a coordenação e cooperação entre os componentes, os limites do controlador podem ser reduzidos.

A adoção de uma estrutura distribuída para sistemas de controle de chão-de-fábrica permite a descrição das atividades de controle de uma forma genérica, fornecendo a necessária abstração para os diferentes níveis e a possibilidade de reconfiguração indispensável para representar as modificações no sistema. Entretanto, mesmo com a utilização de um modelo distribuído na especificação do sistema, a implementação do software pode assumir uma forma diferente daquela do modelo desenvolvido. Isto é frequentemente devido à utilização de ambientes que não suportam a implementação de software distribuído (concorrente).

O objetivo principal da abordagem proposta neste trabalho é :

Oferecer uma estrutura para o projeto do software de controle de chão-de-fábrica em sistemas de manufatura celular (modelo distribuído), e um ambiente de suporte para a sua implementação.

Desta forma a geração do controle para sistemas específicos será facilitada com base em estruturas e procedimentos genéricos que caracterizam os componentes do sistema. Para alcançar este objetivo de forma satisfatória, os seguintes aspectos são especificamente tratados:

- Visto que as arquiteturas de controle correntes não tratam diretamente aspectos do desenvolvimento de software, ou tratam com insuficiência de detalhes, a determinação de um modelo para o controle de chão-de-fábrica é considerada importante. Este modelo prevê uma estrutura distribuída que permite a integração do projeto do sistema e o desenvolvimento de software assim como a separação de aspectos referentes à execução do controle daqueles que se referem ao escalonamento e planejamento.

- O desenvolvimento de interfaces genéricas para os equipamentos de chão-de-fábrica a partir da utilização de uma estrutura para o software de controle, incluindo um esquema de representação (método de decomposição e modelo de comportamento), pode facilitar a geração dos sistemas de controle de chão-de-fábrica. As interfaces dizem respeito principalmente à função de execução/controle do controlador.

- Considerando que o desenvolvimento de controladores genéricos envolve o desenvolvimento de software com base em componentes independentes que interagem para a realização das tarefas requisitadas, alguns serviços básicos de suporte para a execução dos componentes devem ser especificados. Estes serviços permitirão ao software de controle características como : multi-tarefa,

comunicação entre tarefas, sincronização entre tarefas e tratamento de eventos de E/S [1] de forma independente do ambiente de execução específico.

- A necessidade de controladores com grande flexibilidade torna essencial a existência de ambientes de desenvolvimento e controle apropriados que possibilitem a especificação da lógica de controle necessária para a execução de diferentes peças. Em função das necessidades definidas, o software de controle pode ser composto de uma forma ou de outra, permitindo assim a reconfiguração do sistema. Neste caso a interface de usuário deve permitir o desenvolvimento dos programas de controle, por parte do engenheiro de manufatura.

1.5 Organização do Trabalho

Neste capítulo, foi apresentada uma introdução aos sistemas de manufatura e seus conceitos básicos, em termos de estratégias de manufatura, tipos de controle e software de controle, com o propósito de caracterizar a importância dos sistemas de controle de chão-de-fábrica na manufatura celular. O objetivo da pesquisa foi identificado como a proposta de uma estrutura para o desenvolvimento do software de controle para sistemas de chão-de-fábrica, a partir da utilização de um modelo distribuído de controle. As contribuições esperadas deste trabalho apontam para a obtenção de uma estrutura e ambiente de desenvolvimento de software de controle utilizando a noção de componentes genéricos que possibilitem a geração de sistemas com poder de reconfiguração e portabilidade.

Os próximos capítulos estão assim organizados: o Capítulo 2 apresenta uma revisão bibliográfica enfatizando as arquiteturas de controle de sistemas de

[1] E/S = Entrada / Saída

manufatura existentes e as metodologias de desenvolvimento de software adequadas para o tratamento do controle destes sistemas. No Capítulo 3 é apresentada a abordagem proposta para a definição da estrutura de desenvolvimento do software de controle para sistemas de chão-de-fábrica e o modelo de implementação. O Capítulo 4 apresenta os modelos genéricos desenvolvidos para representar os componentes dos sistemas de chão-de-fábrica, com respeito à função execução dos controladores. No Capítulo 5 é apresentado um protótipo que inclui o ambiente de suporte para a implementação do software de controle e uma interface de usuário que permite a configuração do programa de controle. O Capítulo 6 apresenta as conclusões e algumas propostas para futuras extensões.

CAPÍTULO 2
CONCEITOS E ABORDAGENS: UMA REVISÃO

Com o objetivo principal de situar o trabalho no contexto do que existe atualmente para controle de chão-de-fábrica, uma revisão bibliográfica é apresentada a seguir. Alguns aspectos são considerados importantes na busca do desenvolvimento de Sistemas de Controle de Chão-de-Fábrica genéricos e flexíveis : as arquiteturas de controle de manufatura, os requisitos de comunicacão para o controle de chão-de-fábrica e as metodologias de desenvolvimento de software de controle.

As arquiteturas de controle de manufatura são revisadas com o propósito de estabelecer uma estrutura apropriada para o chão-de-fábrica. Três principais estruturas de controle são apresentadas : centralizada, hierárquica e heterárquica. Com o objetivo de estabelecer de forma mais precisa os requisitos de comunicação para os sistemas de chão-de-fábrica, algumas propostas de padrões de comunicação são apresentadas. Considerando que o foco desta pesquisa é o desenvolvimento de software para controle de chão-de-fábrica, são apresentadas metodologias existentes que poderiam ser usadas para o projeto e desenvolvimento desse software. Finalmente, é apresentado um resumo apontando algumas necessidades correntes que deveriam ser focalizadas no que diz respeito ao desenvolvimento de sistemas de controle para chão de fábrica.

2.1 Arquiteturas de Controle de Sistemas de Manufatura

No contexto do controle de chão-de-fábrica, uma arquitetura de controle deveria fornecer uma boa representação do sistema descrevendo de forma completa e não-ambígua a sua estrutura e os relacionamentos existentes entre as entradas (inputs) e as saídas (outputs) do sistema. Esta representação é importante para o projeto e construção de um Sistema de Controle de Chão-de-Fábrica (SCCF). A partir da arquitetura, que descreve os componentes de controle em termos de tarefas e interações, é possível estabelecer a funcionalidade do sistema antes da sua implementação [BIE89].

Uma arquitetura de controle, com capacidades de reconfiguração e modificação dinâmicas, necessita de algumas condições para que o desenvolvimento do SCCF seja viável, técnica e economicamente. As seguintes condições são apontadas em [DIL91]:

- facilidade de modificação / extensibilidade
- facilidade de reconfiguração / adaptabilidade e
- confiabilidade / tolerância a falhas

Na busca do controle apropriado para sistemas de manufatura, várias arquiteturas de controle têm sido propostas. Em geral três abordagens principais têm sido estudadas[DUF91].A Figura 2.1 mostra estas abordagens; os nós retangulares representam entidades de controle (controladores) e os nós circulares representam as entidades físicas (máquinas).

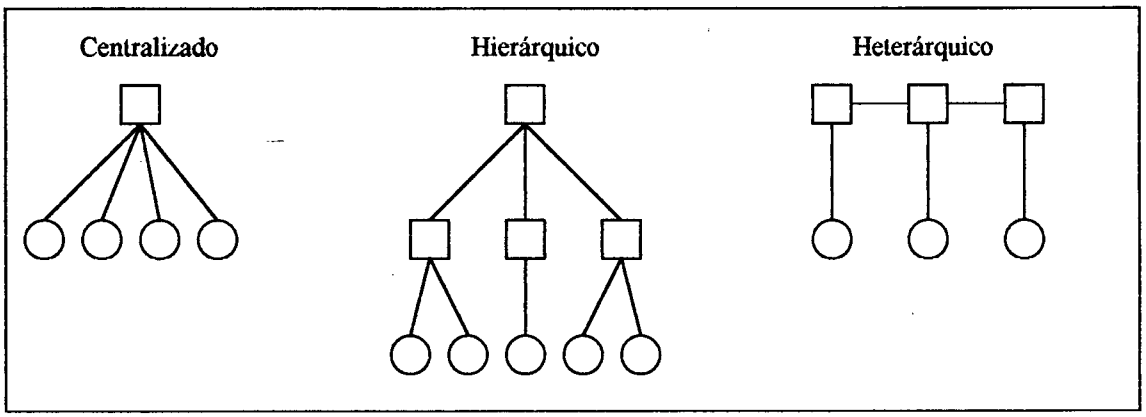


Figura 2.1 :Arquiteturas de Controle de Chão-de-fábrica.
Fonte : Adaptação de [DIL91]

2.1.1 Estrutura de Controle Centralizado

Na arquitetura de controle centralizado, um controlador central é responsável pela realização de todas as tarefas referentes ao planejamento e processamento de informações da fábrica. As tarefas e atividades referentes ao *chão-de-fábrica* são concentradas em um único equipamento com grande capacidade de processamento e decisão, enquanto que pequenos equipamentos sem capacidade de decisão são espalhados pelo ambiente de manufatura[RAN88]. Em outras palavras, um único controlador de chão-de-fábrica é responsável pelo escalonamento das peças nos equipamentos, verificação do estado de todos os recursos do sistema, transferência de programas de controle, e monitoração do sistema de manufatura.

Alguns trabalhos apontam que o fato de existirem diferentes protocolos de communicação para diferentes dispositivos faz com que a presença de um meio central de controle seja importante, pelo menos enquanto não existirem padrões no que diz respeito à comunicação [QUA86]. Entretanto, esta colocação deixa de ser aceitável com a existência de propostas para padronização das comunicações em ambientes fabris, como por exemplo: *Manufacturing Automation Protocol*(MAP) e *Manufacturing Message Specification* (MMS) [VAL92]. Outras razões apontadas para a utilização de controle centralizado em centros de trabalho são: a vantagem do armazenamento central, a disponibilidade de dados globais e o uso de controladores de máquina sem necessidade de capacidade de decisão [DIL91]. Entretanto, a utilização da estrutura centralizada resulta em sistemas com baixa tolerância a falhas e com falta de características como facilidade de modificação e extensão do software de controle. Considerando que alguns padrões de comunicação estão sendo desenvolvidos e aceitos pelos fabricantes de dispositivos de manufatura, o controle centralizado dentro dos centros de trabalho pode dar lugar a outras formas de conectividade entre os dispositivos, embora o papel central do controlador ainda

permaneça. Correntemente, o controle central é um dos métodos mais utilizados para o controle de células/centros de trabalho [BRA89].

2.1.2 Estrutura de Controle Hierárquico

Sistemas com controle hierárquico são baseados e construídos utilizando o conceito de níveis de controle, sendo arranjados em uma estrutura piramidal. Este tipo de controle estabelece relacionamentos do tipo *mestre-escravo*. Cada nível de controle é reponsável pela decomposição do comando recebido, do seu nível superior, em sub-comandos mais detalhados e sua posterior passagem para o nível inferior. Neste tipo de controle é usada a filosofia de distribuição de tomada de decisão, onde cada nível toma suas decisões de acordo com comandos do nível superior e resultados do nível inferior.

Existem pelo menos três grandes projetos envolvendo a arquitetura hierárquica de controle: o modelo hierárquico de controle *Advanced Manufacturing Research Facility* (AMRF) proposto pelo National Institute for Standards and Technology (NIST) , nos EUA; o modelo *Advanced Factory Management and Control Systems* (AFMCS) desenvolvido pelo Computer-Integrated Manufacturing-International (CAM-I), envolvendo EUA, Europa e Ásia; o *European Strategic Program for Research and Development in Information Technology* (ESPRIT), em seu projeto 932 na área de controle de sistemas de manufatura, que envolve a área industrial e acadêmica na Europa.

O NIST, através da sua proposta AMRF [McL86], tem como objetivo principal suportar as necessidades de automação em indústrias de manufatura de peças discretas com pequenos lotes (batch), como por exemplo, aquelas que

fornecem peças para a indústria aeronáutica, automobilística, e de máquinas, as quais são responsáveis por 75% da produção nos Estados Unidos[CHO93].

Algumas características importantes da hierarquia AMRF são: 1) a decomposição da hierarquia de controle de manufatura em níveis bem definidos, 2) a decomposição uniforme das ordens de serviço através da hierarquia, 3) o conceito de um módulo genérico de controle da produção gerenciando as atividades de cada nível, 4) a definição de elementos de trabalho para todos os sistemas de controle, 5) a representação consistente do plano de processo em cada nível, 6) serviços de comunicação e banco de dados independentes, e 7) um modelo de transição de estados genérico para os principais sistemas de controle.

A arquitetura AMRF é apresentada em uma hierarquia de cinco níveis: corporação (facility), fábrica (shop), célula (cell), centro de trabalho (workstation), e equipamento (equipment), como mostra a Figura 2.2. O nível mais alto da hierarquia, nível de corporação (facility level), é responsável pelas seguintes tarefas : planejamento do processo, gerenciamento da produção, e gerenciamento de informações. As funções envolvidas no planejamento de processo usualmente incluem CAD (Computer-Aided Design) e CAPP (Computer-Aided Process Planning) e tem como objetivo preparar a especificação das operações necessárias para as ordens de serviço. As funções de gerenciamento de informações permitem a realização das atividades necessárias nas áreas financeira e administrativa, por exemplo, previsão de custos, cobranças, estoques, pedidos. As funções de gerenciamento de produção estão relacionadas a lançamentos de ordens de serviço, identificação de necessidades de recursos de produção, geração dos planos de produção.

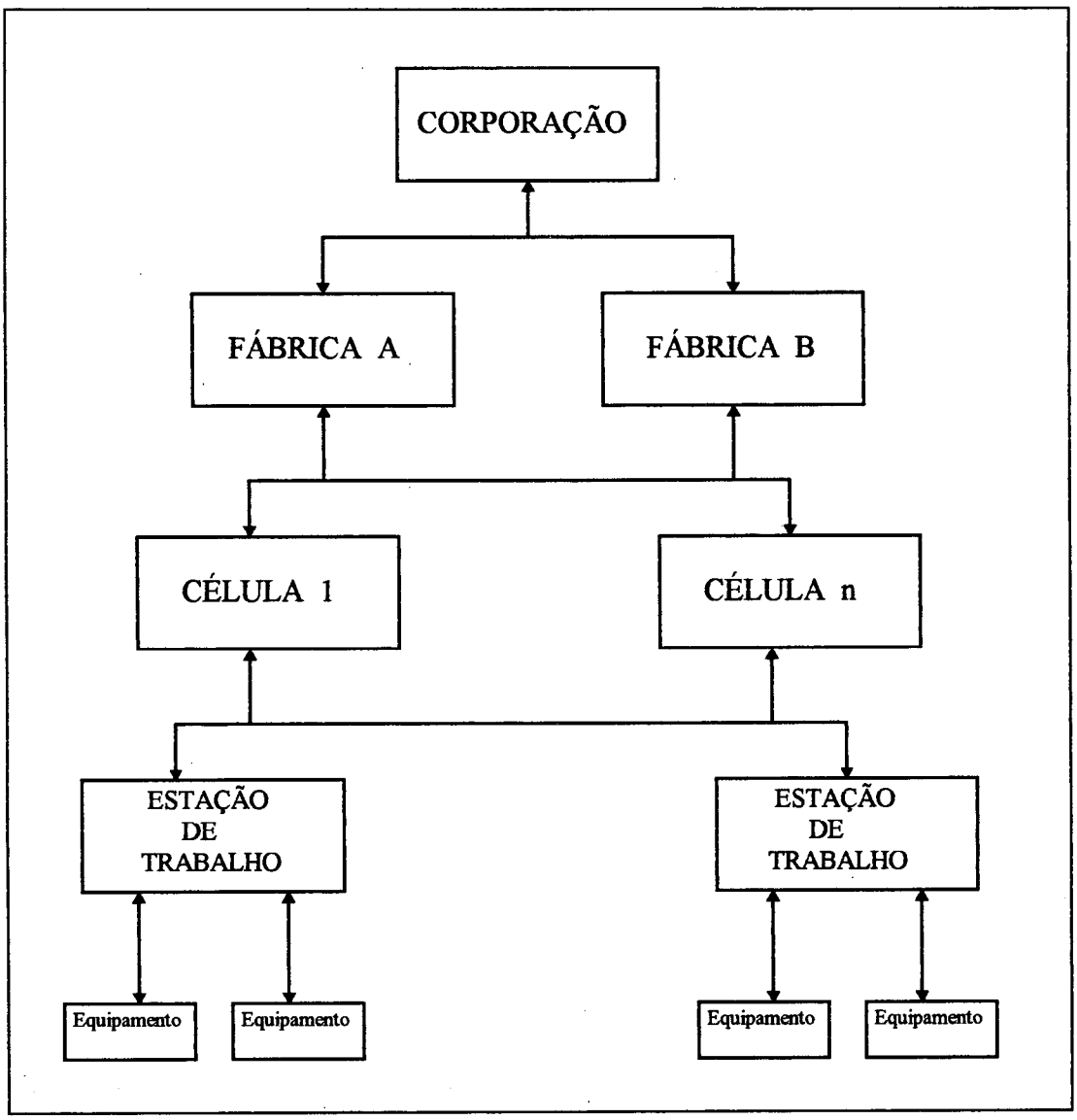


Figura 2.2 : Hierarquia de Controle AMRF - NIST.
Fonte: Adaptação de [SMI92]

O nível de fábrica (shop level) é responsável pelo gerenciamento e coordenação das atividades de produção, escalonamento de *jobs* [1], manutenção de equipamentos, e serviços de suporte. Neste nível o gerenciamento do fluxo de trabalho e grupamento de *jobs* em lotes de peças, é feito usando o conceito de tecnologia de grupo, ou seja, as peças são agrupadas segundo um conjunto de características comuns. Além disto, atividades como alocação de ferramentas e materiais também fazem parte deste nível. As atividades realizadas neste nível são re-avaliadas com base nas informações recebidas do nível inferior e das mudanças que ocorrem no nível superior.

O nível de célula (cell level) é responsável pelo escalonamento dos dispositivos de manuseio de material e ferramental dentro da célula. Este nível é definido como uma coleção de máquinas que formam uma célula virtual. A célula virtual é uma estrutura dinâmica de controle da produção que é definida por um número fixo de equipamentos do chão-de-fábrica.

O nível de centro de trabalho (workstation level) coordena as atividades de um conjunto de equipamentos físicos do chão-de-fábrica, tipicamente constituído de um robô, uma máquina-ferramenta, e um "buffer" para armazenar material. O controlador do centro de trabalho sequencia as operações para completar os *jobs* alocados pelo controle da célula.

O nível de equipamento (equipment level), é composto pelos controladores individuais dos equipamentos, por exemplo, máquina-ferramenta, robôs ou manipuladores de material, que são responsáveis pela leitura de dados e descarga de programas específicas de cada dispositivo.

---

[1] tarefas a serem executadas.

Os modelos de sistemas de banco de dados e redes de comunicação de dados também são importantes no desenvolvimento da estrutura hierárquica de controle. O desenvolvimento na área de sistemas de banco de dados tem como principais objetivos: 1) definição e projeto das estruturas de dados necessárias para armazenar informações de controle e planejamento da empresa, e 2) utilização de sistemas de gerenciamento de banco de dados para manter as informações necessárias para a operação da arquitetura AMRF. Existem duas bases de dados hierárquicas e distribuídas: a base de dados de planejamento e a base de dados de controle. A base de dados de planejamento contém os planos de processo, informações sobre as peças, informações de escalonamento, e requisitos de ferramentas e material. A base de dados de controle contém informação do estado da fábrica, incluindo estado das ferramentas, máquinas, ordens de serviço. O objetivo da rede de comunicação de dados é fornecer a integração física entre os computadores e os processos controlados.

A hierarquia proposta em CAM-I [BOU91], é similar à AMRF e consiste de quatro (4) níveis, nível de fábrica, nível de célula, nível de centro de trabalho e nível de recurso. O nível de controle da fábrica (factory control), considera aspectos tais como: determinação dos requisitos do produto final, e planejamento do processo. O nível de célula (job shop), é responsável pela determinação de comandos para as operações de processamento. O nível de centros de trabalho (work center) gera as tarefas necessárias para a execução das operações de processamento requisitadas e as passa para o nível inferior. O nível de recurso (unit/resource) subdivide as tarefas do nível superior em sub-tarefas e as executa.

O projeto ESPRIT apresenta uma hierarquia de controle que é derivada da hierarquia AMRF onde, em cada nível as decisões são responsabilidade de um

controlador[MEY86]. Cada controlador é composto por sub-controladores ou, unidades de tomada de decisão que têm a mesma estrutura interna e inclui um sistema especialista para o planejamento, interpretação e diagnóstico do nível controlado. O sistema especialista de cada nível resolve tarefas específicas de controle. Por exemplo, o controlador de centro de trabalho gera um plano diário e modifica este plano de acordo com o estado corrente da fábrica.

A arquitetura hierárquica oferece vários benefícios em relação aos outros tipos de arquitetura. A estrutura rígida do relacionamento *mestre-escravo* apresentada pela arquitetura hierárquica resulta em tempos de resposta mais rápidos, tanto na célula quanto nos outros níveis [DIL91]. O mapeamento da hierarquia de controle lógico para a arquitetura física é facilmente conseguido na arquitetura de controle hierárquico [RAN88]. Entretanto, algumas desvantagens também são apontadas.Uma delas se refere à rigidez que a estrutura hierárquica tende a impor desde a fase de projeto, fazendo com que qualquer mudança posterior seja de difícil implementação[DUF88]. Outra desvantagem está no fato de que a falha de um nível leva à paralização dos controladores dos níveis inferiores.

2.1.3 Estrutura de Controle Heterárquico

Este tipo de controle pressupõe a não existência de relacionamentos do tipo *mestre-escravo*, ou seja, o sistema é composto por entidades inteligentes que cooperam para realizar seus objetivos. Esta arquitetura é defendida a partir de uma implementação deste tipo de controle em um sistema de manufatura[DUF88]. O autor aponta que um sistema com estas características faz com que as informações sejam localizadas, desta forma isolando um módulo do outro. Os módulos comunicam-se através de mensagens e o desenvolvimento do sistema é menos complexo devido à independência que os módulos possuem. Na arquitetura

heterárquica de controle, todos os subsistemas participantes têm: 1) direitos iguais no acesso a recursos, 2) acesso a todos os componentes, 3) modo de operação independente, e 4) funcionamento de acordo com as regras de protocolo do sistema.

Uma característica importante deste tipo de arquitetura é a capacidade de tolerância a falhas, visto que o sistema é composto por entidades autônomas. A falha de um componente não afeta de maneira significativa a performance do sistema. Outra característica desta arquitetura é a eliminação ou minimização de informações globais com o objetivo de melhorar características, como: modularidade, facilidade de modificação, extensibilidade. Isto implica na existência de uma quantidade significativa de informações detalhadas em cada controlador de dispositivo (equipamento). Como resultado, o sistema operacional utilizado no controlador de uma arquitetura heterárquica é mais complexo que o usado no caso do controle hierárquico.

A organização das atividades entre controladores, na arquitetura heterárquica, é realizada a partir de um procedimento de negociação. Neste procedimento, cada controlador se comunica e negocia por troca de mensagens, em tempo real , com os outros controladores e um sistema arbitrador com o objetivo de escalonar e rotear as peças. Entretanto, as atuais limitações e deficiências tecnológicas tendem a limitar a adoção do controle heterárquico. Por exemplo, diferenças e incompatibilidades em termos de formatação interna, sistemas operacionais, protocolos de comunicação, e sistemas de banco de dados, fazem com que o controle cooperativo se torne de difícil implementação e operação. Por outro lado, a falta de ambientes de software suportando o conceito de processamento distribuído também se configura em desvantagem na utilização deste tipo de controle.

2.1.4 Resumo

A evolução das estruturas de controle tem sido acompanhada pelos seguintes fatores : incremento na autonomia dos componentes de controle, redução no uso de informações globais, e tendência de diminuição da relação mestre/escravo. A Tabela 2.1 mostra um resumo das características, vantagens e desvantagens das arquiteturas de controle.

A estrutura de controle hierárquico, onde as células ou centros são usados como meio de distribuição de controle dos níveis mais altos para o nível de equipamentos, tem sido estabelecida cada vez mais como um método de controle para a automação fabril. Desta forma ela é geralmente a escolha preferida no projeto da arquitetura de controle.

Por um lado, a estrutura de controle hierárquico proporciona a possibilidade de tratamento de dados globais (ou semi-globais) e respostas mais rápidas, mas estabelece uma rigidez ao sistema que dificulta os aspectos referentes a modificação e independência dos componentes do sistema e também apresenta uma baixa tolerância a falhas. Por outro lado, a estrutura heterárquica proporciona uma grande flexibilidade no que diz respeito a modificação do sistema e independência dos componentes, mas requer um grande volume de comunicação entre os componentes e também exige que cada componente tenha seus próprios dados localmente, ou seja, não existem dados globais.

Características	Vantagens	Desvantagens
Centralizado		
- um único computador de controle - decisões de controle centralizadas - atividades do sistema registradas em um banco de dados global	- acesso a informações globais - possibilidade de otimização global - informações de estado do sistema têm uma única fonte	- tempo de resposta vagaroso e inconsistente - confiabilidade em uma única unidade de controle - software de contole difícil de modificar
Hierárquico		
- múltiplos computadores (heterogeneos) - relacionamento mestre/escravo entre os níveis - supervisor coordena atividades dos subordinados - banco de dados agregado a cada nível	- implementação gradual, redução dos problemas de desenvolvimento de software, redundância - tempo de resposta rápido - possibilidade de comportamento adaptativo	- limitação computacional dos controladores locais - dificuldades no tratamento de controle adaptativo dinâmico - dificuldades de realizar futuras modificações não previstas
Heterárquico		
- múltiplos computadores - sem relacionamento mestre/escravo - tomada de decisão distribuída na coordenação de atividades - banco de dados local	- total autonomia local - redução da complexidade do software - tolerância a falhas implícita - facilidade de reconfiguração e adaptação - difusão rápida das informações	- limitações técnicas dos controladores - falta de padrões para protocolo, e sistemas operacionais - necessidade de capacidades de rede muito altas - não disponibilidade de software

Tabela 2.1 :Resumo das características, vantagens e desvantagens das arquiteturas de controle. Fonte : Adaptação de [DIL91]

Com o objetivo de melhorar aspectos relativos ao desenvolvimento de software, a arquitetura adotada para o chão-de-fábrica deveria apresentar características que pudessem auxiliar na decomposição dos requisitos funcionais do sistema particionando-os em um conjunto quasi-independente de entidades comunicantes, onde :

- A configuração física do sistema deveria ser transparente para as entidades do sistema, sendo que entidades não teriam a necessidade de conhecer onde outras entidades residem.
- As comunicações entre as entidades do sistema se dão na forma de mensagens,e o fluxo de mensagens de comunicação entre as entidades possa ser modelado, independente do nível de controle.
- As entradas, saídas e ações de cada um dos componentes envolvidos no sistema possam ser modeladas.
- O relacionamento entre as entidades é estabelecido a partir de uma política do tipo *cliente/servidor* (uma entidade cliente requisita serviços e uma entidade servidora atende o pedido). Uma entidade qualquer pode representar o papel de cliente e de servidor.

2.2 Padrões de Comunicação no Chão-de-Fábrica

Considerando que o chão-de-fábrica é formado por diferentes dispositivos com diferentes protocolos de comunicação, a tarefa de fazer com que os mesmos se conectem tem representado uma barreira para a integração flexível desses componentes. Na definição de padrões de comunicação, dois caminhos são conhecidos: o desenvolvimento de padrões por parte dos grandes fabricantes de computadores (cada um com o seu próprio padrão) e o desenvolvimento de padrões

por parte de organismos internacionais de padronização com a ajuda dos fabricantes e dos usuários. No passado, a falta de padrões internacionais de comunicação exigiu que os grandes fabricantes de computadores desenvolvessem seus próprios padrões. Com isto, os pequenos e médios fabricantes de dispositivos de automação, tais como: robôs, controladores lógicos programáveis, sensores e atuadores, têm que decidir entre permitir que os seus produtos possam aceitar todos os padrões proprietários de comunicação (estabelecidos pelos fabricantes) ou escolher um deles e restringir o seu mercado. Os usuários, por sua vez, encontram problemas similares. Os diferentes padrões existentes fazem com que o mercado seja fragmentado em produtos incompatíveis entre si, cada um com a sua solução proprietária específica. Desta forma, a partir da escolha de um padrão de comunicação, o usuário é forçado a adquirir computadores e dispositivos de automação apenas daqueles fabricantes que oferecem o tipo de padrão escolhido.

No nível de chão-de-fábrica alguns dos padrões de comunicação que mais se popularizaram incluem Ethernet, Token Bus/Ring, RS-232, RS-422, RS-485, Allen-Bradley's Data HighWay e Siemens SINEC [ANG87][JOH89]. Mesmo que estes padrões de comunicação sejam tecnicamente viáveis sob o ponto de vista do chão-de-fábrica, eles falham na capacidade de comunicar-se uns com os outros, formando assim uma barreira nos esforços de integração, visto que nenhum fornecedor tem todas as soluções e dispositivos de diferentes fornecedores não se comunicam, a não ser com a adoção de soluções específicas que normalmente representam um custo alto [JON88].

A solução para o problema dos padrões de comunicação começou a ser desenhada a partir da especificação do padrão OSI (Open Systems Interconnect) estabelecido pela ISO (International Standards Organization). Na mesma época a General Motors começou a trabalhar em uma implementação de um padrão baseado

em OSI, denominado MAP (Manufacturing Automation Protocol), cujo objetivo era reduzir o problema dos padrões de comunicação no ambiente industrial para alguma coisa do tipo *conectar um aparelho doméstico na tomada.* Em outras palavras, a necessidade era de poder conectar qualquer computador, máquina CN (Comando Numérico), controlador de robô, ou outro dispositivo em uma rede e os mesmos se comunicarem com sucesso.

2.2.1 O padrão MAP

A definição do padrão MAP foi baseada nos seguintes critérios :

- adoção do modelo de referência ISO-OSI selecionando para cada nível os protocolos definidos por organismos internacionais de padronização que atendam os requisitos da automação industrial;
- promoção de novos protocolos, submetendo-os a organismos internacionais de padronização, em áreas onde não existem padrões satisfatórios.

O padrão MAP na sua proposta *FullMAP* I (Figura 2.4) inclui todos os sete níveis definidos pelo modelo de referência OSI e utiliza o padrão de rede Token Bus. Este tipo fornece uma grande flexibilidade para as estações comunicantes. Entretanto, não é recomendado para aplicações de tempo real devido à falta de capacidade de fornecer velocidade de comunicação suficiente, em função do número e da complexidade dos protocolos usados. Com isto, este tipo de implementação é mais utilizado nos níveis mais altos da hierarquia de controle dos sistemas de manufatura. Na sua proposta *MiniMAP* não estão incluídos os níveis 3 a 6 do modelo de referência OSI permitindo assim a garantia de melhores tempos de resposta. Neste caso os únicos níveis OSI implementados incluem o nível 1 (Físico), nível 2 (Ligação) e o nível 7 (Aplicação).

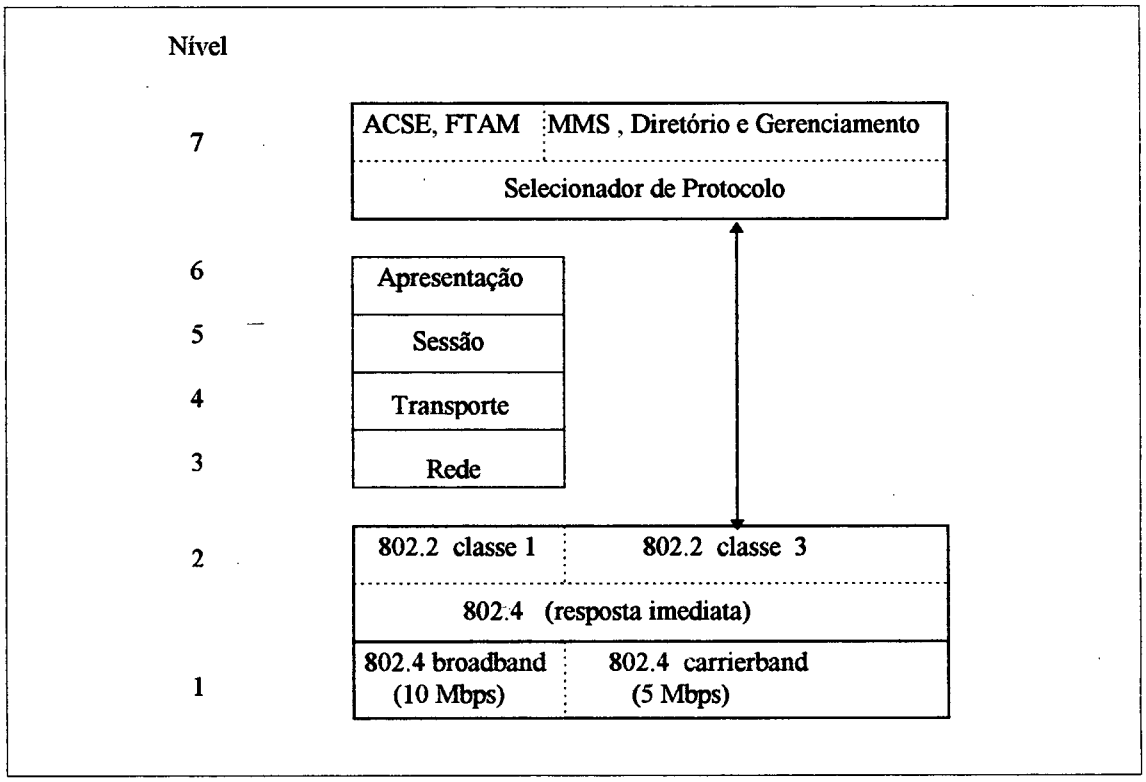


Figura 2.4 : Níveis do padrão MAP.
Fonte : Adaptação de [VAL92]

Os protocolos no nível de aplicação incluem apenas o MMS (Seção 2.2.2) para as mensagens do ambiente de manufatura e os serviços de gerenciamento da rede e diretório. A Figura 2.4 mostra o padrão MAP com as versões *FullMAP* e *MiniMAP*.

2.2.2 O MMS

O MMS é um serviço do nível de aplicação que padroniza as mensagens de comunicação de e para dispositivos programáveis em um ambiente CIM. Ele é designado como padrão ISO-9506 pela ISO e como RS-511 pela EIA (Electronics Industries Association). É o primeiro protocolo do nível de aplicação para o chão-de-fábrica [LED90].

A proposta do MMS é permitir a interconexão de sistemas, sem que para isto seja necessário o completo conhecimento do dispositivo remoto. MMS suporta comunicações entre dispositivos programáveis tais como controladores de robôs, CN's, CNC's, CLP's e outros dispositivos inteligentes como controladores de célula. O MMS é basedo no conceito de *Cliente-Servidor* e no chamado Dispositivo Virtual de Manufatura (VMD-Virtual Manufacturing Device).

No conceito Cliente-Servidor, o cliente requisita serviços e o servidor fornece os serviços. Por exemplo, um controlador de centro de trabalho como cliente poderia requisitar para um robô um serviço do tipo *mover peça de A para B*, o controlador do robô por sua vez como servidor trataria de executar o serviço requisitado. O dispositivo virtual de manufatura, definido no servidor, é uma representação abstrata da estrutura externa e do comportamento de um dispositivo de manufatura. Ele faz com que os recursos e funcionalidades associadas ao dispositivo de manufatura estejam disponíveis para o controle e monitoração por parte de uma aplicação cliente[MAC92].

Os serviços em MMS representam os serviços genéricos que podem ser aplicados a uma grande variedade de dispositivos. Seu desenvolvimento inclui todos os tipos de indústrias e por isto ele se configura num documento complexo. Entretanto, tipicamente uma aplicação usa apenas um pequeno sub-conjunto dos serviços fornecidos pelo MMS.

Apesar de todos os esforços em busca de uma padronização no que diz respeito a comunicação em ambientes fabris, a maioria dos dispositivos programáveis de manufatura instalados ainda implementam um dos padrões incompatíveis. Algumas razões são apontadas como determinantes da situação atual. Primeiro, a implementação de algumas características básicas da especificação OSI tem sido muito lenta e até esquecida. Isto gera uma certa hesitação por parte do pessoal envolvido neste processo em termos de investir recursos financeiros em uma especificação incompleta. Segundo, a tecnologia tem um custo elevado. Apenas as grandes organizações têm condições de adquiri-la. Outra consideração que deve ser feita diz respeito à falta de interesse que as empresas que trabalham nesta área (vendendo soluções prontas ou proprietárias) têm em abrir os seus sistemas temendo a perda de mercado.

2.3 Desenvolvimento de Software de Controle

A evolução do software para sistemas de manufatura tem estado fortemente relacionada com os avanços tecnológicos nas áreas de manufatura e de ciências da computação. A integração dos dispositivos de manufatura tem sido realizada através da automação destes dispositivos, a partir da utilização de unidades de controle numérico e do desenvolvimento de interfaces de software que facilitem esta integração. A tarefa de integração dos vários dispositivos de um sistema de

manufatura apresenta requisitos como: suporte de hardware (conecção física dos componentes) e suporte de software (conecção lógica dos componentes) adequados. Além disto, é necessário um software adicional para coordenar as operações dos diversos dispositivos e para implementar as estratégias de controle destes sistemas integrados. Este software, usualmente, recebe o nome de *software de controle*.

Na maioria dos casos, o software de controle é desenvolvido utilizando linguagens gráficas, como diagramas lógicos de relés (Relay Ladder logic). Estes diagramas especificam os procedimentos de entrada e saída de um Controlador Lógico Programável(CLP), que dirige as operações dos dispositivos de manufatura. Todas as combinações das entradas de um CLP, que são significativas ao processo controlado, devem ser capturadas pelo diagrama. Estes diagramas são considerados de baixo nível de abstração e difícil interpretação. Em alguns casos outras linguagens gráficas, com nível de abstração mais alto são utilizadas. É o caso dos diagramas de funções sequenciais (Sequential Function Chart),ou Grafcet,onde o fluxo de controle é descrito através da utilização de fluxogramas e as ações e decisões são descritas utilizando diagramas lógicos ou linguagem *C* [GIU93].

O desenvolvimento de software de controle para sistemas de manufatura é considerado como uma tarefa difícil, principalmente devido à complexidade que estes sistemas apresentam tanto a nível conceitual como a nível de implementação. Além disto, o desenvolvimento do software de controle representa um custo alto no que diz respeito ao desenvolvimento de sistemas de controle de chão-de-fábrica.

A utilização de esquemas de representação convenientes e ambientes de desenvolvimento que possibilitem a criação de sistemas de controle flexíveis (genéricos) e de fácil utilização, são importantes na busca da diminuição dos custos de desenvolvimento e da complexidade de implementção desses sistemass. Yourdon

[YOU88] define metodologia de desenvolvimento de software como "Uma especificação formal de um sistema para construir sistemas. Ela define as partes componentes de um sistema para a construção de sistemas de informação computadorizados, ou seja, as fases ou atividades que existem em um projeto de desenvolvimento de software típico". No que diz respeito ao desenvolvimento de software de controle para sistemas de chão-de-fábrica, esquemas de representação são usados com o objetivo de modelar tanto o sistema físico quanto a interação entre sistemas de aplicação. Neste caso, dois aspectos são considerados importantes : a estrutura do sistema de controle e a formalização ou descrição dos aspectos referentes ao comportamento do mesmo.

Com o objetivo de representar os componentes de um sistema de software, duas abordagens (ou paradigmas) têm se destacado: Análise Estruturada e Orientação a Objetos. Uma das abordagens mais conhecidas é a decomposição funcional ou análise e projeto estruturado [DEM78]. Nesta abordagem o sistema é hierarquicamente decomposto em subsistemas com base nas funções que o mesmo realiza. As entidades do mundo real são representadas em domínios separados: dados e procedimentos. Diagramas ER (Entidade-Relacionamento) e DFD (Diagramas de Fluxo de Dados) são usados para identificar os relacionamentos entre as entidades e representar as funções e dados significativos ao sistema. Seguindo esta abordagem podemos citar métodos como SREM [ALF85], SADT [BAN88], SSAD [GAN82], IDEF [ICA81].

Na abordagem Orientada a Objetos (OO), as entidades são caracterizadas por objetos que são a representação lógica das entidades do mundo real [MEY88]. Enquanto na abordagem funcional cada módulo representa uma transformação, na orientação a objetos cada módulo é responsável pelo gerenciamento de um objeto (físico,conceitual) do sistema real, ou seja , pode representar várias tranformações.

Uma vez que que o sistema foi decomposto (módulos, objetos), as atividades que devem ser realizadas pelos subsistemas e as interfaces entre os subsistemas devem ser definidas, ou seja, os aspectos referentes ao comportamento do sistema e/ou dos subsistemas são especificados. Por exemplo, tarefas concorrentes precisam se comunicar para trocar informações e controle. Na descrição do comportamento dos componentes do sistema é comum o uso de linguagens gráficas com alguma base formal. Alguns exemplos de formalismos utilizados na descrição do comportamento de sistemas de controle de manufatura são: máquinas de estado, redes de Petri, e linguagens formais.

2.3.1 O Desenvolvimento Orientado a Objetos

A computação orientada a objetos representa um meio poderoso para controle de acesso a dados compartilhados, abstração de dados, modularidade, e representação de conhecimento estrutural. A orientação a objetos é considerada um método de projeto e desenvolvimento de software no qual o sistema é organizado como uma coleção de *objetos* discretos. Diferente da forma tradicional de projeto de software, onde os dados e os procedimentos são considerados como entidades independentes e fracamente conectadas, em OO um módulo (objeto) é definido como um pacote contendo dados e procedimentos, ou seja, uma abstração de dados privados e operações que estão naturalmente associados. Através da utilização desta facilidade de abstração é possível a representação de entidades do mundo real, como por exemplo máquina e peças, com uma correspondência um para um.

O termo objeto apareceu de forma independente em vários campos da Ciência da Computação, no começo dos anos 70 [YON88], para referir notações que eram diferentes na sua aparência. Todas estas notações foram criadas para

gerenciar a complexidade dos sistemas de software, de forma que objetos representem componentes de um sistema decomposto modularmente, ou unidades modulares de representação do conhecimento. Computação ou processamento de informação em uma abordagem orientada a objetos é representada como uma sequência de mensagens passando entre os objetos. Uma vez que um objeto é uma entidade auto-contida provida de um protocolo de comunicação unificado, a decomposição de um sistema em uma coleção de objetos é muito flexível e a estrutura do sistema resultante tende a ser uma visão natural do problema.

Independente de linguagem, os ***objetos*** encapsulam um conjunto de dados privados que somente podem ser acessados ou modificados com a ativação de seus métodos. Os ***métodos*** são a descrição de um comportamento associado a um ou mais objetos, definindo um conjunto de operações a serem efetuadas sobre os seus dados, no momento que ele receber (de outro objeto) uma mensagem explícita solicitando sua execução. A ***mensagem*** é o mecanismo de comunicação entre objetos, através do qual se desencadeia a execução de um método específico. A mensagem indica o objeto destinatário, seletor do método e, opcionalmente, um conjunto de argumentos. O objeto receptor determina o método a ser executado que, por sua vez, devolve informações ao objeto solicitante, podendo também enviar mensagens a outros objetos. A Figura 2.5 mostra a idéia de objeto e a interação entre objetos [ADI89].

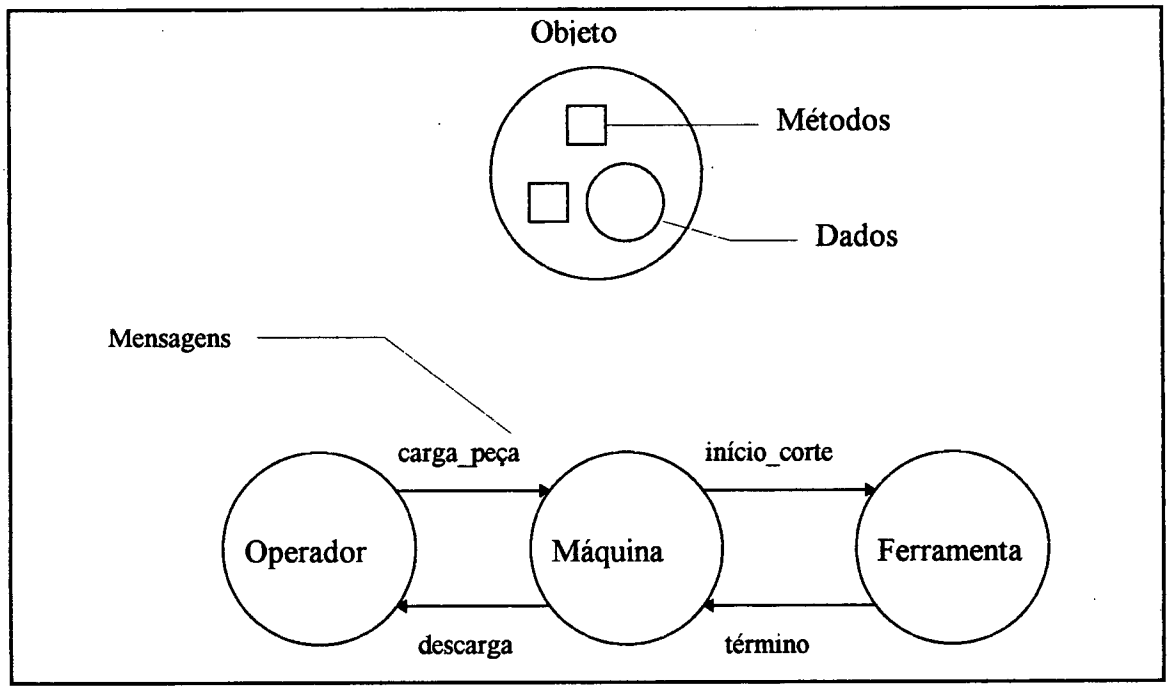


Figura 2.5 : Caracterização de um objeto e a interação entre objetos.

A ***abstração*** é definida como: "Uma especificação ou descrição simplificada de um sistema que enfatiza algumas propriedades ou detalhes do sistema enquanto suprime outras" [SHA84]. Como processo, a abstração é entendida como a extração de detalhes essenciais sobre um determinado item ou grupo de itens, enquanto outros são ignorados. Como entidade, a abstração é entendida como um modelo, uma visão, ou alguma outra representação de um item real. Como processo, a abstração envolve algumas operações importantes, *Classificação, Generalização* e *Agregação* [TAK89]. Por exemplo, na ***Classificação*** (o mecanismo de abstração mais básico) os objetos similares são identificados como instâncias de uma classe, que descreve propriedades comuns a todas as instâncias. A Figura 2.6 mostra a classe *Máquinas-Ferramentas* que é composta por três sub-classes , *Torno, Furadeira e Fresadora* . Neste caso os detalhes essenciais dizem respeito ao tipo da máquina, operações, etc.,enquanto que outros são ignorados. A operação de classificação tem uma operação inversa, que é denominada ***instanciação***.

Denomina-se classe de objetos um conjunto de objetos similares sendo que alguns objetos são especialização de outros objetos. Uma forma particular de hierarquia e relacionamento entre classes, ***herança***, permite que a definição de uma classe seja baseada em uma (simples) ou mais (múltipla) classes já existentes.

2.3.1.1 Conceitos Básicos do Modelo de Objetos

Características como *abstração de dados, herança, encapsulamento e polimorfismo*, são consideradas como importantes [PAS86] no que diz respeito à capacidade de uma linguagem suportar completamente o estilo de programação orientada a objetos e o desenvolvimento de software orientado a objetos.

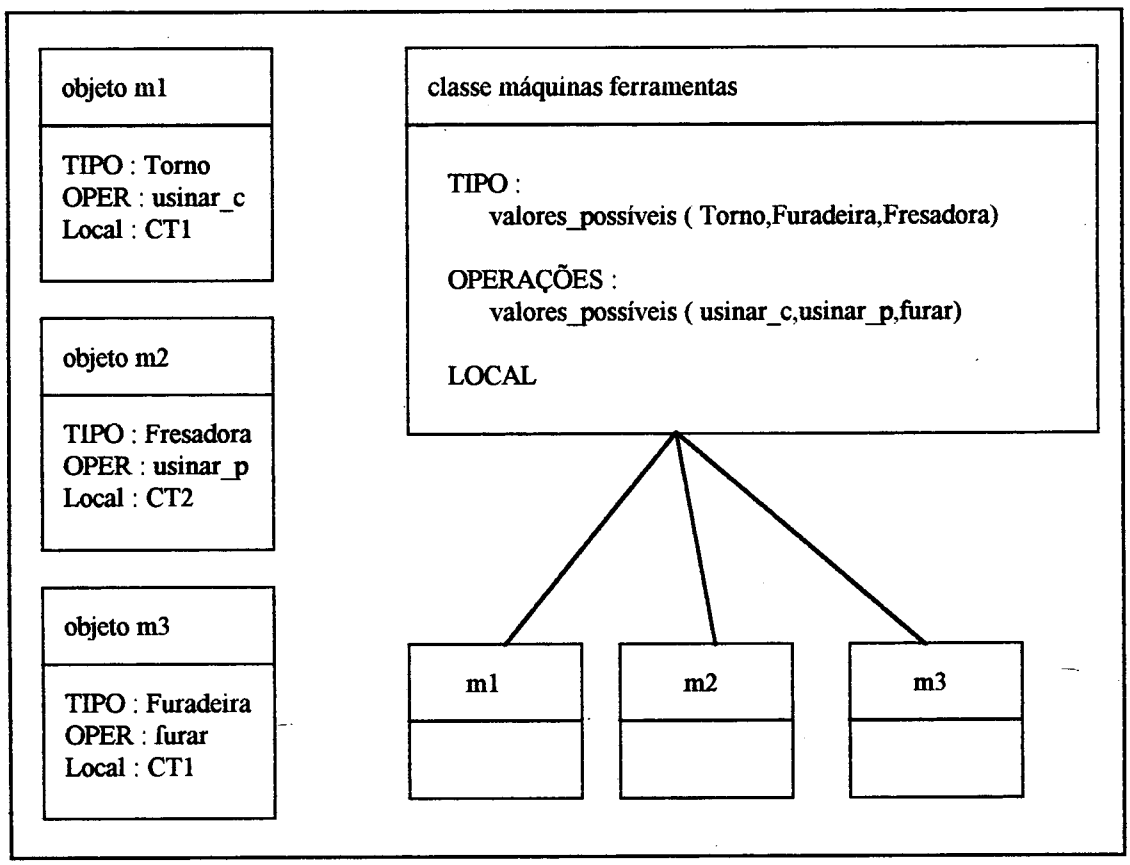


Figura 2.6 : Classificação / Instanciação

Na ***herança simples***, uma subclasse pode herdar instâncias de variáveis e métodos a partir de uma classe pai (superclasse), possivelmente adicionando algum método e instâncias de variáveis. Para entender o conceito de herança, o modelo será ilustrado através de um exemplo de herança simples, mostrado na Figura 2.7. Quando dizemos que Torno é uma Máquina CN, estamos dizendo, *o objeto Torno é uma instância da sub-classe máquina CN.* O objeto Torno herda todas as variáveis de instância e métodos da sub-classe Máquina CN. O objeto Torno pode adicionar seus próprios métodos e variáveis que são apropriados para a especialização do objeto. O objeto Torno pode também redefinir métodos já existentes.

A ***herança múltipla***, representa uma extensão natural para a herança simples, isto é, permite o compartilhamento ou combinação de descrições de várias classes. Neste caso poder-se-ia ver o objeto Torno herdando a partir de múltiplas classes, inclusive a sub-classe Máquina CN.

Um objeto consiste de uma representação encapsulada (estado) e um conjunto de mensagens (operações) que podem ser aplicadas no objeto. Como processo, ***encapsulamento*** é entendido como a colocação de um ou mais itens em um *pacote* (físico ou lógico). A idéia é de, ao invés de organizar programas em procedimentos que compartilham dados globais, os dados são encapsulados (empacotados) com procedimentos que acessam estes dados. A meta é separar o usuário do objeto do seu implementador. O usuário não se preocupa em *como* o objeto é implementado. O usuário só pode utilizar um objeto através das mensagens que o implementador fornece (interface do objeto). O benefício mais visível do encapsulamento é que é possível mudar a implementação de um objeto sem afetar as aplicações que usam o mesmo[THO89].

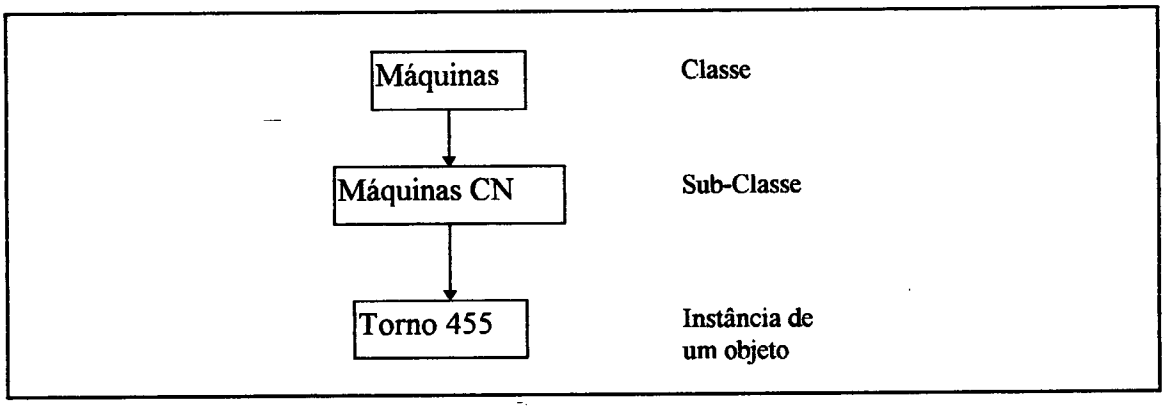


Figura 2.7 : Conceito de Herança.

O conceito de tipo, derivado da teoria de tipos abstratos de dados, representa a caracterização precisa das propriedades estruturais e comportamentais, que entidades de uma coleção compartilham. As linguagens podem ter tipagem forte (*strong typing*), no que diz respeito à consistência dos tipos de dados, e ligação estática (*static binding*), com relação ao momento em que os nomes são associados aos tipos de dados. Relacionado com a interação entre herança e *dynamic binding*, está o conceito de ***polimorfismo***, que por sua vez, é uma representação dentro da teoria de tipos de dados, em que um único nome (tal como uma variável) pode denotar objetos de diferentes classes relacionadas pela mesma superclasse. O polimorfismo permite que uma mesma mensagem elicite respostas diferentes, em função da classe que a receber. A vinculação de mensagem à ação (*binding*) pode ser parcial ou totalmente dinâmica. Isto quer dizer que a decisão de qual procedimento será invocado por uma determinada mensagem é tomada enquanto o programa está em execução. Por exemplo, quando uma mensagem *fim_de_operação_de_produção* é enviada para um objeto representando um centro de trabalho, este objeto incrementa seu contador de máquinas disponíveis e decrementa seu contador de máquinas ocupadas. A mesma mensagem pode ser enviada para um objeto do tipo lote (conjunto de peças) que irá mudar seu estado atual*ocupado* para *esperando*. O uso do polimorfismo enfatiza o fato de que o mesmo evento é experimentado por vários objetos de diferentes classes que reagem de forma apropriada a suas naturezas individuais.

2.3.1.2 A Orientação a Objetos em Sistemas de Manufatura

Como descrito na seção anterior, um objeto, num programa orientado a objetos, é composto por dados privados e um conjunto de procedimentos que podem acessar estes dados. Alguns procedimentos têm acesso público, permitindo que

outros objetos possam requisitar serviços através do envio de mensagens. Similarmente, um sistema de manufatura é visto como uma coleção de entidades que comunicam-se entre si para requisitar serviços com o objetivo de executar uma tarefa do sistema [ROG91]. Esta correspondência natural entre o modelo de objetos e sistemas de manufatura faz com que a orientação a objetos se adapte bem à modelagem de sistemas de manufatura. Várias características de OO são apontadas como vantajosas no tratamento de sistemas de manufatura [ADI89][GLA89][McF89].

Um problema que surge quando se quer *representar o mundo real* em um computador, são as transformações que as informações do mundo real devem sofrer até serem compreendidas como uma representação computacional (abordagem funcional). A idéia de que pessoas possam construir uma representação computacional do mundo real, sem se preocupar com estas transformações, permite que as entidades que compõem os sistemas de manufatura (peças,máquinas,etc.), sejam representadas diretamente como elas são vistas pelas pessoas ligadas ao sistema de manufatura, ou às situações de trabalho.

Estudos sobre problemas de implementação de sistemas de manufatura recomendam uma *abordagem incremental* para sua implementação. Duas características do paradigma orientado a objetos, *modularidade dos objetos* e *conceito de herança*, tornam esta abordagem incremental possível. Os sistemas de manufatura são caracterizados pela complexidade e necessidade de detalhes. Experiências em engenharia de software sugerem que uma maneira apropriada para tratar com muitos detalhes é através de sucessivas fases de refinamentos. Abordagens orientadas a objetos permitem este estilo de desenvolvimento de forma natural. As classes na orientação a objetos podem ser usadas para generalizar ou para especializar. A utilização de uma abordagem orientada a objetos na

representação de entidades que compõem os sistemas de manufatura pode representar uma redução no custo do desenvolvimento de software para gerenciamento e controle de sistemas de manufatura em função de características como modularidade e reusabilidade.

Herança, abstração de dados e reusabilidade, são características da orientação a objetos que aparecem como vantajosas na representação do conhecimento estrutural dos componentes de sistemas de manufatura. Além destas características, outras considerações, relativas ao modelo de objetos, devem ser apontadas no que diz respeito a software de controle para sistemas de manufatura [LIN94] :

- *Especificação e representação da lógica de controle*

Na abordagem orientada a objetos, as ações de controle são modeladas como sequências específicas de mensagens para outros objetos. Este tipo de representação torna o sistema flexível, visto que a implementação das respostas é resolvida pelo objeto receptor. Entretanto, enquanto a especifição do sistema se torna mais fácil com OO, a especificação do controle é afetada pela tendência que o analista tem de tratar controle de forma procedural [NOF94]. Assim sendo, uma forma de representação, intuitiva e de fácil compreensão, que trate da dinâmica do fluxo de mensagens deve ser oferecida.

- *Flexibilidade da representação*

Uma vez que o software projetado a partir da utilização do modelo OO representa quase que exatamente o sistema do mundo real, ele possui propriedades como: facilidade de compreensão e facilidade de modificação. As modificações necessárias em um dos objetos requerem quase nenhuma modificação nas outras

partes do software. Desta forma, comparado com a abordagem convencional de tabela de estados, não existe uma lógica de controle fixa e central. Os objetos possuem seus próprios estados e a mudança de estados ocorre conforme os eventos que são capturados pelos objetos.

- *Tolerância a falhas*

Uma característica importante no projeto de sistemas de controle diz respeito a tolerância a falhas. Por exemplo, num sistema de controle tipicamente hierárquico, uma entidade em um determinado nível necessita de conhecimento suficiente das entidades do nível acima, assim como das entidades do nível abaixo, para tratar de falhas. Isto aumenta a dificuldade e o custo do projeto no que diz respeito a manutenção e modificação dos sistemas hierárquicos[DUF91]. Na abordagem OO, os objetos são projetados de forma a terem o mínimo de conhecimento sobre o comportamento dos outros objetos. OO utiliza menos dados globais e mais controle local, de forma que os objetos podem armazenar seus próprios dados e acessar dados remotos conforme suas necessidades. A partir da utilização destas características é possível projetar sistemas onde a falha de um objeto não resultará na parada de outros.

- *Arquitetura do sistema de controle*

A partir da utilização da OO é possível tratar os vários níveis de controle, máquinas, centro de trabalho, e célula de trabalho, como entidades independentes que não têm qualquer conhecimento sobre o comportamento uns dos outros e que comunicam-se através de mensagens. Este ligação fracamente acoplada, entre as entidades de controle , permite que estas entidades possam ser alteradas ou substituídas de forma individual sem afetar o funcionamento geral do sistema. Isto

pode permitir uma flexibilidade em relação à definição da estrutura de controle, hierárquica ou heterárquica, a ser utilizada.

A aplicação de técnicas orientadas a objetos em sistemas de manufatura tem crescido rapidamente e várias áreas de manufatura têm aproveitado as vantagens que a OO apresenta em termos de modelagem e desenvolvimento de software[ICO92]. Alguns trabalhos, particularmente com respeito a controle de chão-de-fábrica, são citados a seguir.

Fabian & Lennartson apresentam uma abordagem orientada a objetos para o controle de uma célula de trabalho. O controle da célula é caracterizado como sendo um sistema distribuído composto por objetos físicos que são autocontidos e trabalham de forma concorrente a partir da troca de mensagens. Cada objeto externo é modelado por um objeto interno que contém um conjunto de mensagens comum a todos os dispositivos de manufatura [FAB92].

Uma metodologia de modelagem para sistemas de informação de manufatura baseada em objetos (OOMIS-Object-Oriented modeling methodology for Manufacturing Information Systems) é apresentada [KIM93]. A metodologia consiste de uma fase de análise e uma fase de projeto. Na fase de análise, as funções de manufatura são decompostas, descritas com diagramas funcionais, e tranformadas em tabelas: função, dados e operações. Na fase de projeto, as tabelas são traduzidas em um modelo de informação orientado a objetos. O modelo de informações é composto pelas classes dicionário (descreve as funções e dados) e relacionamento (descreve a semântica das operações) .

O desenvolvimento de software de controle para equipamentos do chão-de-fábrica em ambientes de manufatura discreta é discutido em [SMI92]. É apresentado

o desenvolvimento de uma classe de objetos, *equipamentos*, que forma a base para o desenvolvimento de controladores para o nível de equipamentos e que se encaixa em uma arquitetura hierárquica de controle com três níveis. Os objetos de software têm uma correspondência um para um com os objetos físicos, e apresentam uma parte genérica e uma parte específica.

Em [LIN94], é apresentado o desenvolvimento de um sistema de controle de uma célula flexível orientado a objetos. Os objetos e suas relações são modelados utilizando diagramas Entidade-relacionamento (ER), a interação entre os objetos é modelada utilizando diagramas de fluxo de mensagens (message flow diagrams), e o comportamento dos objetos é modelado com diagramas de transição de estados. Implementado em Turbo C++, em um computador IBM PS/2, o software aceita modificações na configuração da célula, sem reprogramação.

Glassey & Adiga [GLA89] propõem uma biblioteca de objetos de software para controle e simulação de sistemas de manufatura. O objetivo principal do projeto era o desenvolvimento de um conjunto de modelos de simulação para simplificar a tarefa de montagem de modelos de simulação apropriados para pesquisas específicas. Neste caso, o paradigma de orientação a objetos foi escolhido com o objetivo de prover características como: reusabilidade, extensibilidade, e facilidade de manutenção do software de simulação

Naylor & Volz [NAY87] apresentam os componentes *software/hardware* como parte integrante de um modelo formal para o controle de manufatura. Estes componentes são o encapsulamento de um dispositivo físico (máquina, robô, etc) e o software necessário para o controle de baixo nível deste dispositivo. Os componentes apresentam uma interface pública bem definida e uma implementação

interna que é acessível pelo usuário; a parte visível e a implementação dos componentes deveriam ser compiladas separadas dos componentes que as usam.

Apesar da grande adaptabilidade do modelo de OO no tratamento de sistemas de manufatura, alguns problemas são ainda considerados como não resolvidos[ROG91]. Embora o modelo OO permita modelar sistemas de manufatura como sendo objetos de software comunicando-se (solução distribuída), a implementação de um programa OO não é uma solução distribuída visto que existe um fluxo de controle único através do programa. OO quebra o sistema em componentes semi-independentes mas eles não são executados em paralelo. O autor apresenta problemas como : a modelagem de um sistema que envolve módulos de tomada de decisão que funcionam em paralelo em um único processador, a distribuição destes módulos numa implementação real, o tratamento de bases de dados distribuídas que representam o estado do sistema e a possibilidade de tratar requisitos de comunicação de um sistema real na fase de modelagem. Uma solução para o problema da representação do paralelismo dos sistemas de manufatura seria modelar os objetos, que representam os componentes do sistema de manufatura, como processos que podem executar em processadores independentes conectados de alguma forma.

2.3.2 Modelos Formais no Desenvolvimento de Software de Controle

Um modelo formal de um sistema é uma descrição precisa, sem ambiguidades, do sistema apresentado, em uma linguagem independente de implementação e com sintaxe e semântica bem definidas. Um modelo formal é uma visão abstrata de um sistema especificando a sua funcionalidade e o seu comportamento de forma independente e sem restrições de implementação. Através de modelos formais é possível a validação das especificações de um sistema e a

verificação da sua corretude. Alguns benefícios da utilização da especificação formal são [SCH84]:

- Crescimento do entendimento em relação ao comportamento do sistema.
- Redução dos erros na fase de desenvolvimento já que a correção do sistema pode ser verificada antes da implementação.
- Possibilidade de construção de ferramentas computacionais para o desenvolvimento, simulação, verificação, implementação e teste das especificações.

A seguir são apresentadas duas técnicas de modelagem que têm sido muito utilizadas em sistemas de controle de manufatura, o formalismo das redes de Petri e o formalismo dos Autômatos (Linguagens Formais).

2.3.2.1 Redes de Petri

As redes de Petri são uma ferramenta de modelagem, gráfica e matemática, aplicável a muitos sistemas. Como ferramenta gráfica, as redes de Petri podem ser usadas como auxílio na comunicação visual, similar a diagramas de bloco e fluxogramas. Além disto, fichas são usadas para simular as atividades dinâmicas e concorrentes dos sistemas. Como ferramenta matemática, é possível determinar equações de estado, equações algébricas e outros modelos matemáticos dirigindo o comportamento dos sistemas.

O conceito de rede de Petri teve sua origem na tese apresentada por Carl Adam Petri em 1962 [PET62]. Seu desenvolvimento tem sido estudado desde 1970, principalmente como modelo para representação de sistemas. O interesse na aplicação de redes de Petri é crescente e abrange diversas áreas, onde se sobressaem

aquelas relacionadas com sistemas computacionais, tais como : sistemas de manufatura, protocolos de comunicação, bancos de dados, desenvolvimento de software, sistemas distribuídos de computadores, sistemas administrativos [PET81][ROZ86][ROZ87][VAL87] [HEU88][DES94].

Uma rede de Petri (***RdP***) é um tipo particular de grafo orientado junto com um estado inicial chamado marcação inicial. O grafo de uma ***RdP*** é constituído de dois tipos de nós : ***lugares*** e ***transições***, onde os arcos vão de um *lugar* para uma *transição* ou de uma *transição* para um *lugar*. Na representação gráfica, *lugares* são círculos e *transições* são barras ou retângulos. Os arcos são rotulados com seus pesos (inteiros positivos), onde um arco com peso k pode ser interpretado como um conjunto de k arcos paralelos.

Uma ***marcação*** (estado) atribui para cada *lugar* um inteiro não negativo. Se uma marcação atribui a um *lugar* p um k não negativo, diz-se que p está marcado com k fichas. Gráficamente, colocam-se k pontos (fichas) no lugar p. Uma marcação é denotada por M, um vetor com $\mathbf{m}$ elementos, onde $\mathbf{m}$ é o número total de *lugares*. O p-ésimo componente de M, $M(p)$, é o número de fichas no *lugar* p.

Na modelagem, usando o conceito de ***condições*** e ***eventos***, *lugares* representam condições (estado de um recurso, execução de uma operação), e *transições* representam eventos (início ou término de uma operação). Uma *transição* (evento) tem um certo número de *lugares* de entrada e de saída que representam as ***pré-condições*** e as ***pós-condições*** de um evento, respectivamente. A presença de uma ficha em um *lugar* é interpretada como a existência da condição associada com o *lugar*. Numa outra interpretação, k fichas são colocadas em um *lugar* para indicar que k itens de dados ou recursos estão disponíveis.

Uma ***RdP*** de *lugares* e *transições* (place/transition net) pode ser definida como uma quíntupla :

RP = { L, T, E, S, M } onde :

L = { $\ell 1, \ell 2,..., \ell m$} é um conjunto de *lugares*,

T = {t1,t2,...,tn} é um conjunto de *transições*, sendo **L** $\cap$ **T** = $\varnothing$

E : L $\rightarrow$ **T** é a função de entrada,

S : T $\rightarrow$ **L** é a função de saída,

M : L $\rightarrow$ **N** é a marcação da rede.

A marcação pode ser representada por um vetor **M** = { **m**($\ell 1$), **m**($\ell 2$),...,**m**(ℓn) } onde, **m**(ℓi) é o número de fichas existentes no *lugar* ℓi. **M0** é denominado marcação inicial. A Figura 2.8 mostra a representação gráfica de uma ***RdP*** cuja descrição é a seguinte :

RP = { L,T,E,S,M }

L = { $\ell 1, \ell 2, \ell 3, \ell 4, \ell 5, \ell 6, \ell 7$ } **T = { t1,t2,t3,t4,t5 }**

E(t1) = { $\ell 1$ } **S(t1)** = { $\ell 2, \ell 2, \ell 3$ }

E(t2) = { $\ell 2, \ell 2$ } **S(t2)** = { $\ell 4$ }

E(t3) = { $\ell 2, \ell 3, \ell 3$ } **S(t3)** = { $\ell 5, \ell 5, \ell 5$ }

E(t4) = { $\ell 4$ } **S(t4)** = { $\ell 4, \ell 6$ }

E(t5) = { $\ell 5, \ell 6$ } **S(t5)** = { $\ell 1, \ell 7$ }

M0 = { 0,2,1,0,0,3,1 } marcação inicial

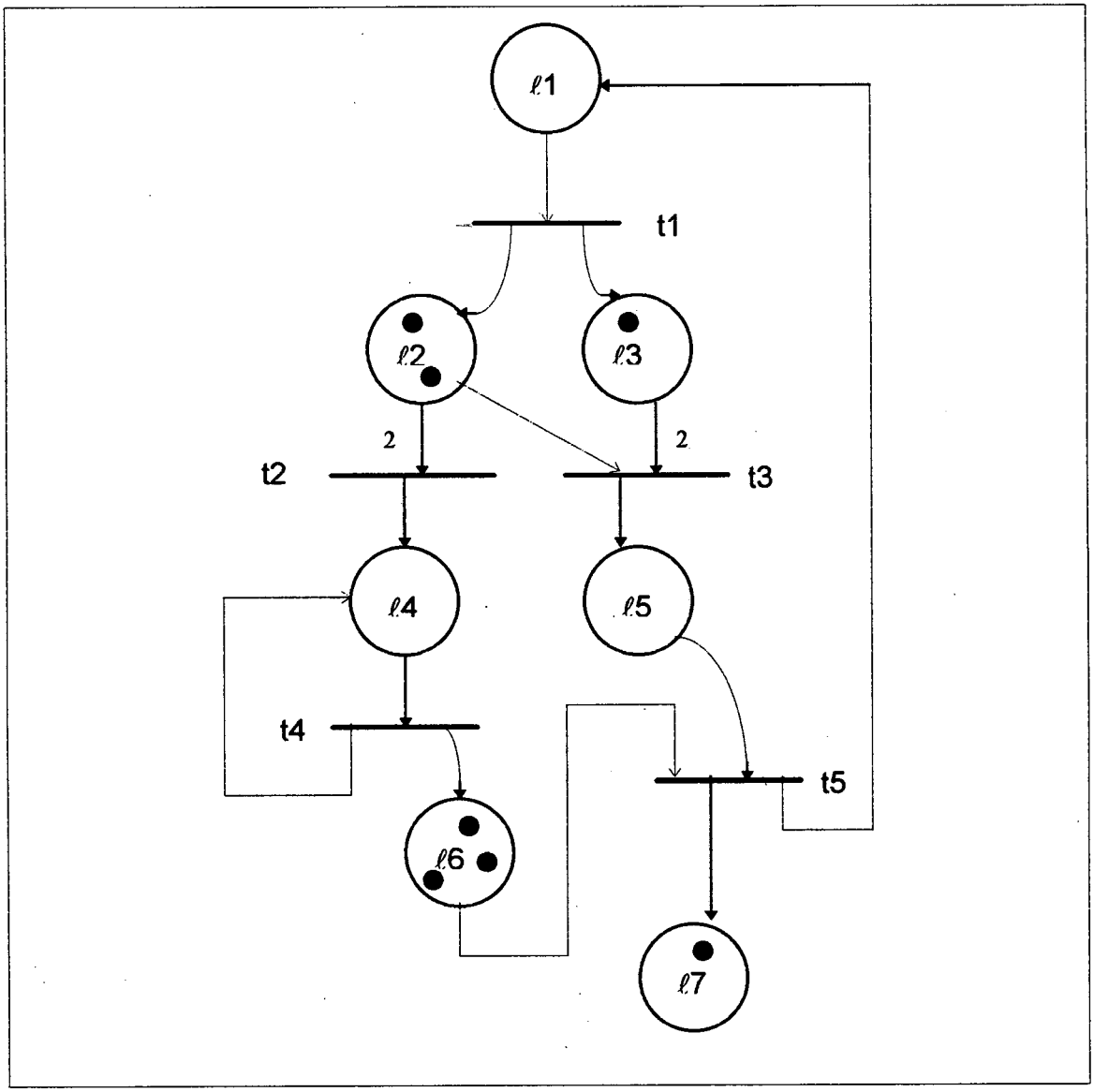


Figura 2.8 : Exemplo de uma rede lugar / transição.

O comportamento de muitos sistemas pode ser descrito considerando os estados da rede e suas mudanças. Para simular o comportamento dinâmico de um sistema, um estado ou marcação nas redes de Petri é modificado de acordo com a seguinte regra de transição (de disparo) :

1. Uma transição t é dita habilitada se cada lugar de entrada ℓ de t está marcado com pelo menos P(ℓ,t) fichas, onde P(ℓ,t) é o peso do arco de ℓ para t.

2. Uma transição habilitada pode ou não disparar (depende de quando o evento realmente acontece).

3. O disparo de uma transição habilitada t remove P(ℓ,t) fichas de cada lugar de entrada ℓ de t, e coloca P(t, ℓ) fichas para cada lugar de saída ℓ de t, onde P(t, ℓ) é o peso do arco de t para ℓ.

A Figura 2.9 mostra o disparo de uma transição em uma rede que modela uma operação de montagem de uma bicicleta.

Uma transição sem qualquer lugar de entrada é chamada transição fonte e uma transição sem qualquer lugar de saída é chamada transição consumidora. A transição fonte está habilitada incondicionalmente e o disparo de uma transição consumidora consome marcas mas não produz nenhuma. Um par com um lugar ℓ e uma transição t é chamado ciclo se ℓ é lugar de entrada e saída de t. Uma ***RdP*** é pura se ela não tem ciclos, e é ordinária se todos os seus arcos têm peso 1.

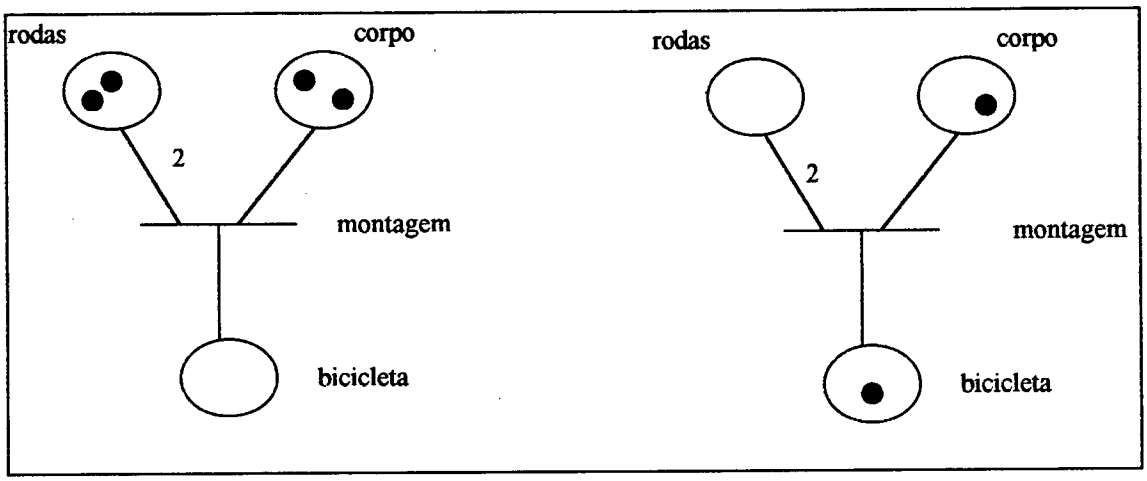


Figura 2.9 : Disparo de uma transição.

A utilização das redes de Petri nos sistemas de manufatura pode se dar desde a fase de especificação funcional até a fase de implementação. As vantagens da utilização das redes de Petri incluem a sua relativa facilidade para representar e modificar a lógica de controle e a sua potencialidade para análise matemática e simulação gráfica para validação do projeto. A partir de uma representação matemática as redes podem ser usadas para a prototipação rápida de sistemas de controle. As redes de Petri têm sido combinadas com outras abordagens na busca de vários propósitos em planejamento de processos e escalonamento, controle inteligente, construção de sistemas especialistas, representação de conhecimento e tratamento de incertezas [ZHO93].

Uma metodologia para especificação, modelagem e simulação de sistemas de eventos discretos é apresentada em [CAN92]. A metodologia apresenta uma linguagem de especificação baseada em regras de produção, sendo que a especificação é traduzida em modelos de redes de Petri que podem ser simulados.

Um método para especificação de sistemas flexíveis de manufatura usando redes de Petri é apresentado em [EZP92]. A metodologia proposta para a especificação é composta de duas fases: inicialmente, o sistema é decomposto em subsistemas menores os quais são especificados; em seguida, as especificações dos subsistemas são integradas formando assim a especificação do sistema como um todo. As redes utilizadas na especificação são do tipo lugar/transição, definido anteriormente, resultando em redes onde o número de lugares e transições pode ser muito alto.

A existência de um grande número de técnicas de análise formal que podem ser utilizadas sobre um modelo baseado em redes de Petri, é um ponto importante na utilização das redes como metodologia de especificação para sistemas

de manufatura[MUR89]. Além disto, é possível, a partir de um modelo de ***RdP***, realizar a simulação do mesmo com o objetivo de verificar propriedades e comportamento.

Quando se trata da implementação do sistema de controle de tempo real do sistema de manufatura, é muito importante dispor-se de um conjunto de ferramentas de engenharia de software para transformar esta descrição em código executável.

O sistema K-NET [NAG92] se configura num sistema de programação para desenvolvimento de software de controle para sistemas flexíveis de manufatura, baseado em redes de Petri de alto nível . O sistema suporta as fases de desenvolvimento, projeto, programação e teste. K-NET é composto por um editor responsável pela edição das especificações de controle (redes de Petri), um simulador das especificações e um gerador de código que pode converter a rede em programas C .

A linguagem orientada a objetos PROTOB [BAL91] é uma metodologia baseada nas redes Prot e um ambiente CASE (Computer Aided Software Engineering) . As redes Prot integram diagramas de fluxo de dados estendidos e redes de Petri em um formalismo orientado a objetos. O ambiente CASE é composto de várias ferramentas que suportam as atividades de especificação, modelagem e prototipação através da utilização da linguagem PROTOB. O ambiente CASE gera código em linguagem ADA ou C de forma automática. Como maiores aplicações a metodologia inclui : sistemas de tempo-real, protocolos de comunicação e sistemas de manufatura [BRU85][BRU86a][BRU86b].

KRON- Knowledge Representation Oriented Nets é um esquema de representação que possibilita a criação de modelos para sistemas de eventos

discretos. Ele é baseado na integração de redes de Petri de alto nível com uma representação baseada em *frames* (tipo de objeto) e uma metodologia orientada a objetos. Os objetivos principais considerados na sua definição são: 1) obter um modelo de representação completo e poderoso para dados e controle; 2) incorporar uma metodologia de modelagem poderosa [VIL93] [VIL94].

2.3.2.2 Autômatos Finitos

Os fundamentos matemáticos que sustentam os modelos baseados em autômatos se encontram na teoria dos autômatos em conjunto com a teoria de linguagens formais. Particularmente, as linguagens formais dizem respeito a definição de linguagens e máquinas capazes de reconhecer se um conjunto particular de caracteres pertencem a uma certa linguagem. Algumas classes de linguagens são : gramática regular (autômato de estados finitos), gramática livre de contexto (autômato pushdown), gramática sem restrições (máquina de Turing). Um autômato finito A é definido pela quíntupla $A = (\Sigma, Q, q0, \delta, F)$, onde :

Σ é um conjunto finito de símbolos de entrada

Q é um conjunto finito de estados

$q0 \in Q$ é o estado inicial

δ é a função de próximo estado definida por $Q \times \Sigma \rightarrow Q$

$F \subseteq Q$ é o conjunto de estados finais

A Figura 2.10 representa de forma conveniente um autômato finito. O controle de estados finito, que pode estar em um dos estados de **Q**, lê os símbolos de entrada em ordem sequencial da esquerda para a direita. Inicialmente, o autômato está no estado inicial **q0** e buscando o próximo símbolo de entrada. Cada símbolo causa uma transição do estado atual para um outro estado do autômato.

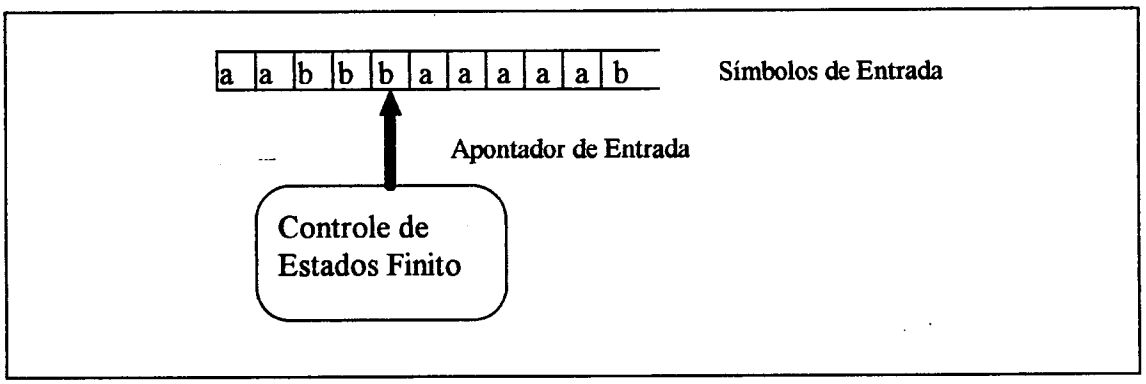


Figura 2.10 : Representação de um Autômato Finito.

A interpretação da função de próximo estado $\delta(q, a) = q'$ onde $q, q' \in Q$, $a \in \Sigma$, é o que segue : o autômato **A**, no estado **q** e recebendo o símbolo de entrada **a**, vai para o estado **q'** e o apontador de entrada move-se para a próxima entrada à direita.

Um autômato finito A é determinístico se, para cada estado **q** e cada símbolo **a**, não existe mais do que uma transição a partir de **q** com a chegada de **a**. Na Figura 2.11 , a função de transição no autômato (a) é determinística e reconhece todos os conjuntos de caracteres (string) com um ou mais caracteres "a" seguidos de um "b". Em (b) a função de transição é não-determinística porque do estado 1 existem duas transições para o símbolo de entrada "b". Este autômato reconhece todos os *strings* compostos por um "a" seguidos por um ou mais "b". Para qualquer autômato finito não-determinístico existe uma representação determinística equivalente, como mostrado em (c) [FU82].

Uma característica importante dos autômatos finitos determinísticos (DFA) é que existem métodos para automaticamente construir programas reconhecedores (analisadores léxicos) com base no autômato [AHO86]. Um analisador léxico reconhece o mesmo conjunto de *strings* que o autômato e permite a execução de módulos de software conforme o reconhecimento dos símbolos do *string*. Os analisadores léxicos são usados em compiladores para decompor o arquivo de entrada (código fonte) em um conjunto de *tokens* que são entrada para um outro componente do compilador (o analisador sintático ou *parser*). O procedimento que verifica o tamanho dos identificadores de variáveis de um programa e os trunca no tamanho especificado pela linguagem, é um exemplo de módulo de software. A partir da modificação da estrutura dos autômatos finitos determinísticos com o objetivo de descrever protocolos de processamento para um controlador, estas técnicas de

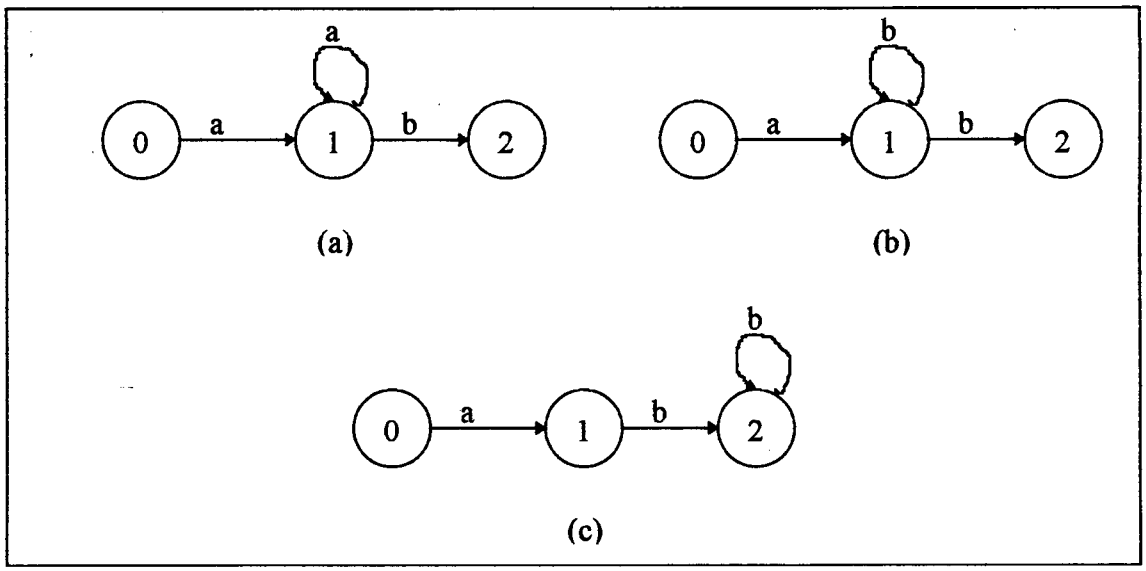


Figura 2.11 : Autômato finito determinístico (a,c) e não-determinístico (b).

geração automática podem ser adaptadas para serem usadas na geração de controladores para o chão-de-fábrica.

Em [SMI92] autômatos finitos determinísticos (DFA) são modificados para serem utilizados na descrição do comportamento de um controlador. O formalismo é chamado de MPSG (Message-based Part State Graph). Duas modificações básicas são realizadas na estrutura dos DFA para a modelagem de controladores de chão-de-fábrica. Primeiro, o mecanismo de entrada deve ser modificado para aceitar entradas de sequências de símbolos múltiplas. Isto é conseguido através da utilização do conjunto de eventos do controlador como símbolos de entrada e criando mecanismos de entrada múltiplos para a leitura dos eventos. Segundo, a função de próximo estado deve ser modificada para que as tarefas do controlador possam ser realizadas durante as transições de estado. Isto é conseguido pela introdução de um conjunto de ações do controlador e uma função de transição das ações do controlador. O modelo MPSG foi desenvolvido para prover uma representação formal do comportamento de controladores de chão-de-fábrica em ambientes de controle distribuídos. A diferença entre o modelo MPSG e outros modelos (baseados em autômatos) formais de controle é que o modelo MPSG é baseado nos estados das peças e não nos estados do sistema. Desta forma o modelo não apresenta problemas de explosão do espaço de estados.

Em [DUA93], é apresentado um esquema de modelagem distribuído chamado EMMN (Extended Moore Machine Networking model) e uma estrutura de controle, baseada em EMMN, para sistemas de eventos discretos. A máquina de Moore (Moore machine) é um tipo de autômato com transição de saída, ou seja, capaz de tranformar um string de entrada em um string de saída. Esta característica é importante quando se quer definir uma máquina abstrata que seja capaz de interagir com o seu ambiente. O conceito básico do modelo EMMN é representar a estrutura

e o comportamento de sistemas através da interconexão de máquinas abstratas. Um modelo EMMN é construído pela definição dos componentes do sistema e suas interconexões. Baseado na teoria dos autômatos, o problema de controle de sistemas flexíveis de manufatura é representado em termos de estrutura e comportamento, através dos conceitos de autômatos finitos e linguagens regulares. Um método formal é proposto para a transição sistemática de um modelo EMMN para o seu correspondente software em um ambiente orientado a objetos. A aplicabilidade do esquema EMMN é demonstrada pela construção de um modelo de software que simula as atividades do EMMN na modelagem de um sistema flexível de manufatura.

Em [VIT94] é apresentado um formalismo de modelagem para sistemas de eventos discretos, baseado em autômatos finitos, com o objetivo de representar o comportamento dinâmico do sistema e a forma como o comportamento do sistema pode ser modificado. A escolha de autômatos finitos é estabelecida pela sua facilidade de utilização para representação do controle e dinâmica dos sistemas de eventos discretos, e pela grande base teórica e resultados práticos já apresentados pelos autômatos finitos.

Uma abordagem sistemática para automatizar o desenvolvimento de software de controle para células flexíveis de manufatura nos níveis de célula, estação de trabalho e equipamento é apresentada em [JOS92]. Uma arquitetura funcional e gramáticas livres de contexto são propostas como base formal, sendo que o sistema é controlado a partir do reconhecimento das gramáticas geradas.

Os dois formalismos apresentados, redes de Petri e Autômatos, são considerados métodos baseados em modelos orientados a transição que seguem uma abordagem operacional. A abordagem operacional descreve o sistema a partir de um

modelo executável. O modelo pode ser matematicamente verificado através do uso de análise estática e validado a partir da sua execução. As duas abordagens são consideradas como aplicáveis à modelagem de aspectos comportamentais dos sistemas de manufatura. Entretanto, as redes de Petri apresentam uma expressividade maior que os autômatos ou máquinas de Estado. Os autômatos são considerados como uma sub-classe das redes de Petri. As redes se destacam especialmente no que se refere à possibilidade de modelagem de aspectos referentes à sincronização e ao não-determinismo da execução (paralelismo). Por outro lado, os autômatos são mais intuitivos com respeito à representação visual . Neste caso a utilização de redes de Petri estendidas, que possibilitem uma melhor representação de aspectos funcionais e estruturais da especificação, significa um melhoramento no aspecto representação.

2.4 Necessidades Correntes

A discussão de vários aspectos envolvidos no desenvolvimento de sistemas de controle de chão-de-fábrica e a revisão de alguns conceitos e metodologias utilizadas com este propósito nos permite projetar algumas necessidades correntes nesta linha de pesquisa. Acredita-se que uma das necessidades seja o desenvolvimento do software de controle genérico e flexível de forma a possibilitar a sua utilização em várias situações de controle e permitir a sua rápida configuração. No entanto, o que se observa hoje é que as soluções para o software de controle de chão-de-fábrica são raramente transportáveis e características como flexibilidade e reusabilidade estão contrapostas com simplicidade e facilidade de uso. As soluções de software que são simples de usar são muito rígidas e podem ser usadas por muito poucas aplicações, enquanto que o software que é moderadamente flexível é complexo e difícil de ser usado. Dois aspectos são importantes na busca do desenvolvimento do software de controle de chão-de-fábrica genérico e flexível: 1) uma arquitetura (modelo) de

controle de chão-de-fábrica que possibilite a definição dos tipos de componentes e seus relacionamentos, 2) uma abordagem para o desenvolvimento que inclua uma estrutura de software, suporte e ambiente de implementação.

Deficiências existentes nos níveis de controle de centro de trabalho e de equipamento trazem problemas na proliferação do conceito de manufatura por célula. O problema maior diz respeito à rigidez, ou seja, à falta de habilidade do usuário em realizar mudanças ou incorporar novas características. Esta rigidez dos controladores está associada a três considerações importantes: conectividade, portabilidade do software e configurabilidade.

Na busca de um modelo ideal para o controle, o fator conectividade define a forma como os controladores se comunicam. Acreditamos que uma arquitetura ideal deva apresentar a possibilidade de comunicação de controladores em diferentes níveis da hierarquia (estrutura hierárquica) assim como no mesmo nível. Isto melhora aspectos referentes à independência de desenvolvimento do software de controle e aumenta as capacidades de tolerância a falhas. Por outro lado, a existência de níveis de controle representa a possibilidade de tratamento de aspectos como dados compartilhados e sincronização de uma forma mais rápida.

Presentemente, o software de controle de chão-de-fábrica precisa ser mais *configurável* e *portável*, objetivando a redução do custo de desenvolvimento do software e possibilitando atingir o nível de flexibilidade desejado no controle de chão-de-fábrica. *Portabilidade* refere-se à habilidade do software de ser transportado para diferentes plataformas sem modificações significativas. *Configurabilidade* refere-se à habilidade do software de ser reconfigurado rápida e facilmente, para acomodar diferentes dispositivos no chão-de-fábrica e diferentes sequências de execução para diferentes peças. As características de configurabilidade e

portabilidade também são influenciadas pela forma com que os controladores de chão-de-fábrica são desenvolvidos. Duas abordagens são consideradas : 1) a construção de controladores específicos para uma aplicação em particular e, 2) o desenvolvimento do controlador de uma forma genérica, sendo que o mesmo pode ser usado em várias aplicações.

No controle específico, tanto o hardware como o software são projetados para atender uma aplicação de manufatura específica. Mesmo que o software seja desenvolvido seguindo uma arquitetura geral, promovendo flexibilidade em relação ao tipo de situações em que ele pode ser aplicado, o produto final é gerado especificamente para uma aplicação em particular. Por um lado, visto que o hardware e o software são projetados e instalados para uma situação específica, a integração dos dispositivos é feita de forma satisfatória. Entretanto, a capacidade de expansão e a flexibilidade são muito reduzidas, e a tarefa de manutenção fica bastante prejudicada.

Na abordagem genérica, o software é geralmente desenvolvido de forma que possa ser configurado para atender a maioria das aplicações, ou seja, para tratar um conjunto de funções que represente os principais requisitos da aplicação. O estabelecimento de uma interface de software para os equipamentos de chão-de-fábrica com o objetivo de superar a falta de um protocolo padrão é apontado como uma das soluções em busca do desenvolvimento genérico. Entretanto, não existe uma descrição sistemática de como construir este nível de interface de software, ou seja, não existe nenhuma especificação largamente aceita para a interface das máquinas, mesmo considerando que o número de máquinas distintas é relativamente pequeno.

Na especificação de interfaces bem definidas para os componentes de um sistema de chão-de-fábrica, requisitos como a definição da estrutura dos componentes e a utilização de abordagens para a descrição do comportamento dos componentes são importantes. A estrutura define a composição dos componentes individualmente e seus relacionamentos com os outros componentes. O comportamento formaliza como os componentes realizam suas tarefas.

Diferentes modelos têm sido utilizados na representação da estrutura, entre eles abordagens funcionais e abordagens orientadas a objetos. A vantagem maior apresentada pelas abordagens orientadas a objetos diz respeito à facilidade de reprodução que estas têm na representação de sistemas de manufatura. Entretanto, a falta de ambientes que suportem características como multiprocessamento, sincronização e comunicação tem restringido ou diminuído o potencial de utilização dos modelos baseados em objetos na representação de sistemas distribuídos.

Na representação do comportamento dos componentes, é importante a utilização de abordagens com alguma base formal, possibilitando assim a especificação de componentes genéricos com capacidade de geração automática de código. Aspectos como definição das atividades, eventos e ações que o componente realiza estabelecem o comportamento do componente. Desta forma, o modelo de representação deve fornecer um meio para a especificação do conjunto admissível de eventos ou trajetórias de eventos. As abordagens com base em redes de Petri e máquinas de estado finito são as mais utilizadas. Com estas abordagens é possível a definição do comportamento dos componentes através de estados e transições de estados (trajetória) e, devido à base formal destas abordagens, é possível também a utilização de métodos para geração automática de código.

A criação de ambientes de desenvolvimento do software de controle de chão-de-fábrica representa um importante requisito para facilitar o desenvolvimento de sistemas de controle de chão-de-fábrica genéricos e flexíveis. Estes ambientes deveriam permitir ao usuário do sistema de controle a possibilidade de : reconfiguração do sistema de controle para diferentes dispositivos de chão-de-fábrica, especificação da lógica de controle necessária no chão-de-fábrica para diferentes tipos de peças e execução da lógica de controle especificada.

Na maioria das propostas apresentadas, tanto no que diz respeito a metodologias de modelagem do software de controle quanto no que diz respeito à arquitetura de controle, os aspectos referentes a suporte de implementação, interface e ambiente de desenvolvimento não são tratados ou são tratados de forma superficial. Neste trabalho os seguintes aspectos são considerados importantes na busca do desenvolvimento de sistemas de controle de chão-de-fábrica :

1) Apresentar um modelo de representação conceitual capaz de capturar a realidade dos sistemas de chão-de-fábrica, que consiste de elementos auto-contidos que comunicam-se e representam objetos físicos. O modelo deve levar em consideração a necessidade de desenvolvimento de sistemas com características de reusabilidade, extensibilidade e adaptabilidade.

2) Fornecer um suporte operacional que atenda os serviços necessários à implementação do modelo. Isto inclui aspectos relacionados com gerenciamento de comunicações, gerenciamento e controle de dispositivos e sincronização. O suporte operacional deve levar em consideração a portabilidade.

3) Fornecer interfaces que permitam a sua utilização tanto por parte de engenheiros de manufatura quanto por especialistas em computação. A ênfase deveria ser na

possibilidade de rápida prototipação de novas aplicações e a habilidade de colocá-las em funcionamento.

No Capítulo 3 é apresentada uma abordagem para o desenvolvimento do software de controle de sistemas de manufatura , que tem como objetivo atender os requisitos aqui estabelecidos.

CAPÍTULO 3
ABORDAGEM PROPOSTA

Este capítulo apresenta uma abordagem para o desenvolvimento de sistemas de controle de chão-de-fábrica. O objetivo principal é permitir o desenvolvimento do software de controle, genérico e flexível, de forma que a incorporação de mudanças sejam facilmente gerenciadas. Os principais aspectos da abordagem dizem respeito ao modelo de controle adotado, que possibilita a descrição dos componentes do sistema, e a estrutura utilizada para o desenvolvimento do software que promove características como reusabilidade e extensibilidade. Este capítulo é apresentado em quatro seções. A primeira seção apresenta uma descrição dos componentes de um sistema de chão-de-fábrica com base no exemplo apresentado no Capítulo 1. Na segunda seção é descrita a arquitetura de controle adotada para o desenvolvimento do sistema de controle. A terceira seção apresenta a estrutura do software de controle, a abordagem utilizada para o desenvolvimento desse software e descreve aspectos referentes à especificação da interface de usuário, objetivando facilitar a utilização e o desenvolvimento do software de controle por parte do pessoal de chão-de-fábrica. A seção final apresenta o modelo de implementação que serve como suporte para o sistema de controle de chão-de-fábrica.

3.1 Caracterização dos componentes de chão-de-fábrica

Com base no exemplo da Figura 1.2 (Capítulo 1), as diferentes máquinas que fazem parte do sistema de chão-de-fábrica podem ser identificadas e classificadas da seguinte forma :

- *equipamentos de processamento,*
- *equipamentos de movimentação de material,*
- *equipamentos de transporte de material e*
- *equipamentos de armazenagem.*

A classe dos equipamentos de processamento é composta por máquinas-ferramenta, dispositivos de inspeção e dispositivos de montagem. A caracterização destes equipamentos diz respeito à capacidade de processar, de forma autônoma, uma peça como descrito pelo plano de processo. Mesmo que os processos destes equipamentos sejam diferentes (torneamento, furação, inspeção, montagem), do ponto de vista de controle eles processam peças de acordo com um conjunto de instruções de processamento. Normalmente, estes equipamentos são acompanhados por um controlador, *controlador de equipamento*, que é tipicamente um controlador industrial padrão, tal como um CLP ou CN, e usualmente fornecido pelo vendedor do equipamento.

A classe dos equipamentos de movimentação de material é basicamente composta por robôs e outros dispositivos capazes de mover peças em uma direção especificada de uma localização para outra, onde as localizações são de pequenas distâncias entre si, considerando o tamanho da fábrica. Estes equipamentos são responsáveis em servir um ou mais equipamentos de processamento, em um centro de trabalho, com operações do tipo carga/descarga de peças. Estes equipamentos também possuem um *controlador de equipamento* que pode ser um CLP ou um controlador de robô.

A classe dos equipamentos de transporte de material é formada por esteiras, AGV's (Automated Guided Vehicles) e outros equipamentos de transporte manual ou automatizado cujo propósito seja o transporte de peças para vários locais dentro

da fábrica. Uma distinção é feita entre equipamentos de transporte e equipamentos de movimentação em função do tipo de transporte que eles realizam. Os primeiros realizam transporte dentro de um centro de trabalho enquanto que os últimos realizam o transporte entre centros de trabalho. Entretanto, um equipamento de transporte (robô ou esteira) poderia ser classificado como qualquer das duas classes. De forma similar à classe dos equipamentos de movimentação de material, os equipamentos de transporte de material também possuem um *controlador de equipamento*.

A classe dos equipamentos de armazenagem inclui máquinas automáticas de armazenamento e recuperação (AS/RS) com capacidade de entregar/recuperar as peças em pontos de carga/descarga definidos, e unidades de armazenamento (buffers) que são locais de armazenamento passivo. As unidades de armazenamento têm uma capacidade limitada e normalmente não tem um *controlador de equipamento*.

Além disto, os pontos de entrada/saída definem para um determinado equipamento o vértice inicial de uma tarefa produtiva (ponto de entrada) e o vértice final da tarefa (ponto de saída).

É importante ressaltar que em um sistema real pode existir a necessidade da utilização de equipamentos de operação manual e equipamentos automáticos de forma permanente ou quando do crescimento ou incorporação de outros equipamentos no sistema. Isto é possível uma vez que, para um determinado controlador, as outras entidades (controlados e controladores) são vistas como caixas pretas. Desta forma, não existe diferença para um controlador de centro de trabalho se ele está mandando/recebendo mensagens para/de um equipamento controlado por software ou para/de um controlado por um operador humano.

Outro componente importante em um sistema de chão-de-fábrica são as peças. Uma peça pode ser considerada como sendo um item individual que é fabricado pelo sistema de manufatura. Com o objetivo de descrever uma peça são usados atributos administrativos e também atributos técnicos, que descrevem a peça e as operações necessárias. Como atributos administrativos uma peça pode incluir as seguintes características : 1) identificador da peça, que identifica de forma única uma peça dentro do sistema; 2) tipo da peça ou número da peça, normalmente usado para documentação; 3) ordem de produção e prazos de entrega; 4) descrição da peça e nome da peça.

A descrição técnica de uma peça usualmente utiliza uma representação CAD (Computer Aided Design) que posteriormente possibilita a criação do plano de processo da peça. Um plano de processo fornece as instruções de processamento necessárias para produzir uma determinada peça. Uma representação para um plano de processo deve mostrar as restrições de precedência e rotas alternativas que a peça tem em termos de operações e máquinas. A figura 3.1 [SMI92], mostra um grafo onde cada nó representa uma operação ou um conjunto de operações realizadas por uma máquina ou um grupo de máquinas. Cada arco representa o movimento de uma peça da máquina atual para a próxima máquina. Cada caminho no grafo representa uma possível rota para a peça.

Por último, os controladores são classificados como componentes de um sistema de chão-de-fábrica, usualmente na forma de sistemas de computação (hardware e software), com capacidade de comunicação e processamento de informação, responsáveis pela coordenação dos equipamentos.

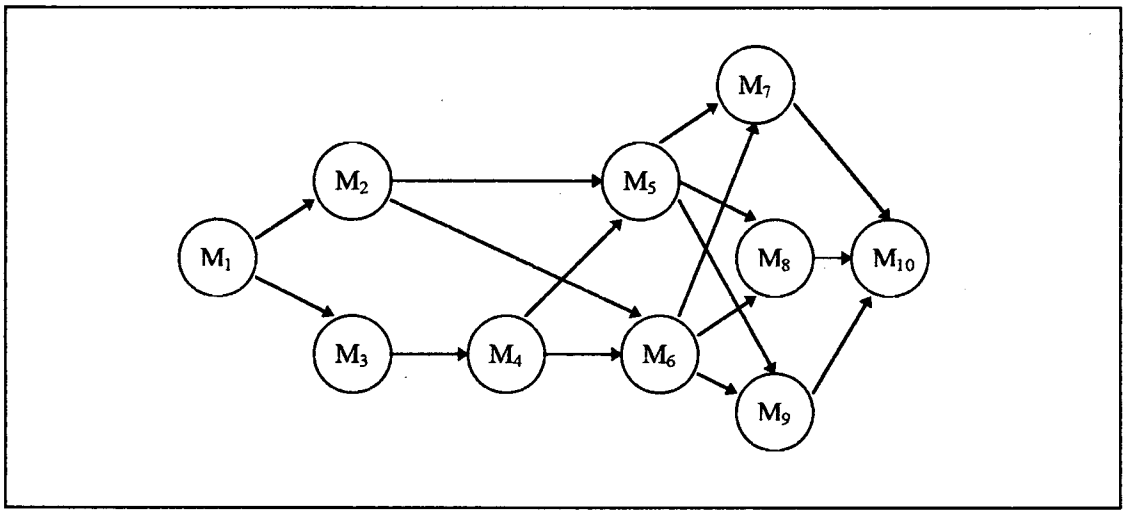


Figura 3.1 : Um exemplo de representação de plano de processo.

3.2 Modelo da Arquitetura de Controle

Uma arquitetura de controle deveser capaz de descrever tanto a estrutura do sistema como os relacionamentos entre as entradas e as saídas, de forma completa e não ambígua. Ela deve descrever os componentes do sistema e fornecer a funcionalidade ou serviços que cada componente oferece. A partir de uma arquitetura bem definida é possível o desenvolvimento de sistemas, onde os componentes são completamente independentes. Isto faz com que a mudança de um componente ou mesmo a sua substituição seja feita de forma transparente, sem afetar o resto do sistema. Dois modelos de arquitetura têm sido estudados para serem utilizados em controle de sistema de manufatura : o controle heterárquico e o controle hierárquico (Capítulo 2).

Analisando as arquiteturas no que diz respeito a flexibilidade, tolerância a falhas e independência de desenvolvimento, a heterárquica apresenta características mais atraentes visto que os módulos são completamente independentes, seus dados locais, e sem conhecimento global. Entretanto, esta falta de conhecimento global tende a contradizer o objetivo de otimização do desempenho do sistema como um todo. Por exemplo, é difícil forçar prioridades entre peças sem conhecimento global, assim como retardos de processamento, que frequentemente são necessários em um escalonamento ótimo, não são facilmente identificados utilizando conhecimento local.

A arquitetura de controle adotada neste trabalho leva em consideração a necessidade de informação global ao mesmo tempo em que tenta manter a independência de desenvolvimento das entidades. A arquitetura de controle de chão-de-fábrica adotada prevê três níveis: *equipamento*, *centros de trabalho* e *supervisão global*.

A Figura 3.2 da uma idéia da estrutura de controle do sistema de chão-de-fábrica mostrado na Figura 1.2, segundo o modelo adotado. O nível de *equipamento* corresponde aos dispositivos físicos de chão-de-fábrica, onde existe uma correspondência um-para-um entre os controladores de equipamentos e as máquinas do chão-de-fábrica (máquinas-ferramentas, robôs, máquinas de inspeção,etc.). O nível de *centros de trabalho* corresponde à integração de várias entidades do nível de equipamento que, juntas, executam alguma tarefa de manufatura. O nível de *supervisão global* é visto como a integração de vários centros de trabalho cuja atividade principal é fornecer o controle organizacional dos mesmos. Este nível funciona como controle central e ponto de interface para o sistema de chão-de-fábrica.

Ao contrário das demais propostas [SMI92] [JOS90], a arquitetura aqui apresentada prevê a ligação entre entidades do mesmo nível, por exemplo entre centros de trabalho e entre equipamentos de um centro de trabalho, promovendo assim um modelo distribuído de controle. Vantagens desta estrutura incluem : a definição dos componentes de forma independente da estrutura global do sistema, possibilidade de descrição das atividades paralelas e simultâneas e capacidade de reconfiguração a partir da definição das interações entre os componentes.

Uma vez que o controle do sistema envolve a participação de várias entidades de controle, cada entidade necessita de *serviços de comunicação* para poder se comunicar com as outras entidades do sistema. Na arquitetura proposta, cada controlador pode se comunicar com o seu supervisor, seus subordinados e com outras entidades do mesmo nível. Neste caso, cada controlador incorpora um *módulo de comunicação* que é responsável pelo provimento dos serviços de comunicação.

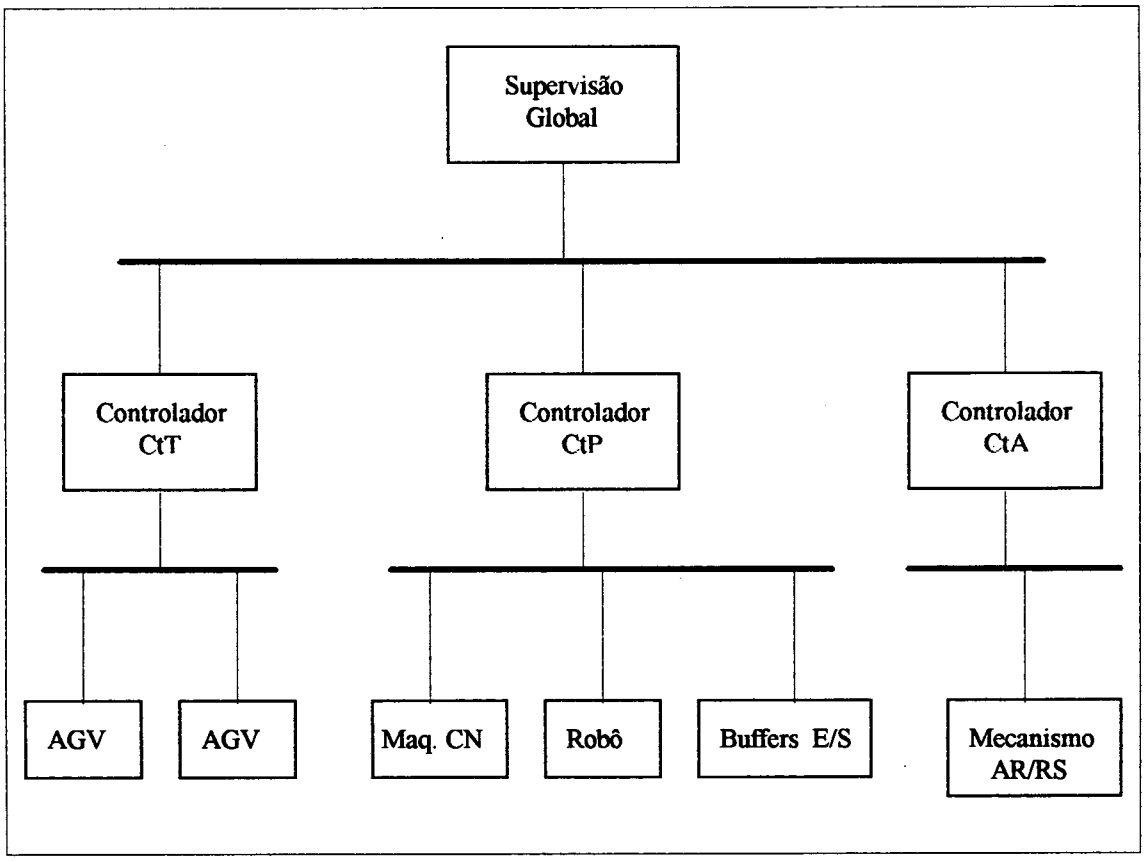


Figura 3.2 : Uma Arquitetura de Controle do Chão-de-Fábrica.

As entidades de controle, independentemente do nível, podem necessitar de acesso a diferentes tipos de informações, como por exemplo planos de processo das peças e arquivos de programas de controle numérico. Estas informações podem estar disponíveis localmente a uma entidade específica ou globalmente a todas as entidades. Cada entidade é responsável pelo provimento das suas informações locais para outras entidades que as requisitarem. Neste caso, um *servidor de informações* é usado para atender as requisições das entidades sejam elas locais ou remotas.

3.2.1 Nível de Equipamento

O nível de equipamento tem como característica principal a representação lógica de uma máquina física, ou seja, uma entidade deste nível é composta pelo controlador do equipamento, o controlador da máquina, e a máquina propriamente dita. O controlador de equipamento está dividido em duas partes : 1) a parte genérica que funciona como interface entre a máquina que ele representa e o restante do chão-de-fábrica, e 2) a parte específica, que é responsável pelo controle da máquina real, através do controlador da máquina. A figura 3.3 mostra a estrutura do nível de equipamento.

O nível de equipamento pode ser definido como o conjunto de equipamentos $\mathbf{E = \{ e_1, e_2, ..., e_n \}}$, onde para cada $\mathbf{e_j \in E}$, $\mathbf{e_j = < Ce_j, M_j >}$ onde: $\mathbf{Ce_j}$ é um controlador de equipamento e $\mathbf{M_j}$ é uma máquina (dispositivo) física e seu controlador.O conjunto **E** pode ser particionado em **{ EPM, EMM, ETM, EAM }** onde:

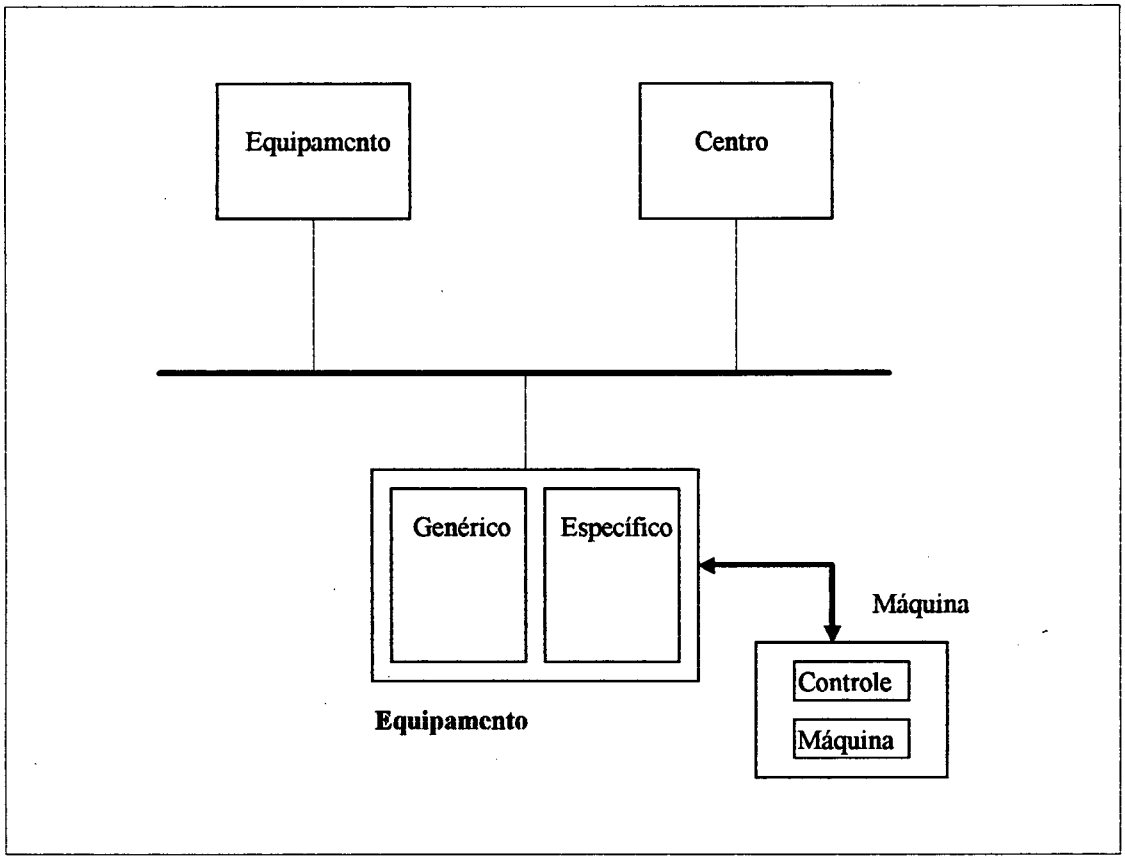


Figura 3.3 : Nível de Equipamento.

EPM = { e_j | M_j é um *equipamento de processamento de material* },

EMM = { e_j | M_j é um *equipamento de movimentação de material* },

ETM = { e_j | M_j é um *equipamento de transporte de material* }, e

EAM = { e_j | M_j é uma *equipamento de armazenamento de material* }.

Equipamentos mais sofisticados, com capacidade de armazenamento e carga/descarga próprios, também são considerados como entidades do nível de equipamentos e não como centros de trabalho. Neste caso, estes equipamentos podem realizar várias tarefas ao mesmo tempo. Por exemplo, um determinado equipamento pode estar realizando uma tarefa de carga de peças ao mesmo tempo em que está realizando uma tarefa de processamento em outras peças.

A comunicação entre o controlador de equipamento (Ce) e o controlador de máquina pode incluir transferência de mensagens, arquivos e sinais, através de canais de Entrada/Saída que são dependentes do tipo de dispositivo controlador (CN, CLP, controlador de robô). A comunicação entre os controladores de equipamentos e entre o controlador de equipamento e o controlador de centro de trabalho, se processa a partir do envio/recebimento de mensagens.

Para processar uma peça, o nível de equipamento segue um plano de processo onde são representadas as operações que o equipamento específico deve realizar. A estrutura de dados correspondente a cada operação contém as informações necessárias para a máquina processar a peça.

3.2.2 Nível de Centro de Trabalho

O nível de centro de trabalho é composto por um ou mais equipamentos sob o controle do controlador de centro de trabalho. Neste nível, a ênfase é na

sincronização dos controladores de equipamento. Usualmente, os centros de trabalho são definidos a partir do *layout* físico dos equipamentos.

Formalmente, $\mathbf{CT} = \{ \mathbf{Ct_1}, \mathbf{Ct_2}, ..., \mathbf{Ct_n} \}$ é um conjunto de centros de trabalho onde para cada $\mathbf{Ct_i} \in \mathbf{CT}$, $\mathbf{Ct_i} = < \mathbf{Cct_i}, \mathbf{E_i}, \mathbf{Pe/s_i} >$ onde:

$\mathbf{Cct_i}$ é o controlador do centro de trabalho,

$\mathbf{E_i} = \mathbf{EPM_i} \cup \mathbf{EMM_i} \cup \mathbf{ETM_i} \cup \mathbf{EAM_i}$,

$\mathbf{Pe/s_i}$ é um ponto de entrada/saída de material.

O controlador de centro de trabalho recebe requisições de serviço do seu supervisor (supervisão global) ou de outros centros de trabalho e é responsável pela movimentação das peças entre os equipamentos através da sincronização das ações necessárias para a coordenação dos equipamentos de processamento (máquina-ferramenta) e equipamentos de manuseio de material (robô) durante a transferência das peças. Os centros de trabalho podem ser classificados em 3 tipos: processamento (CtP), transporte (CtT) e armazenamento (CtA).

Um centro de trabalho de processamento é composto por elementos da união dos conjuntos de equipamentos de processamento de material (EPM), equipamentos de movimentação de material (EMM) e equipamentos de armazenamento de material (EAM), sendo que os equipamentos são agrupados com o objetivo de atender sequências de operações das peças (planos de processo), de forma a maximizar a utilização das máquinas e minimizar as necessidades de transporte de material. Uma vez no centro de trabalho, a peça segue um plano de processo (nível de centro) que especifica as máquinas pelas quais a peça deve passar e a sequência em que as operações devem acontecer.

Um centro de trabalho de transporte é composto por equipamentos de transporte de material (ETM) e fornece serviços de transporte de um centro para outro. Quando necessário equipamentos de movimentação de material podem ser usados para transferir peças entre EPM's. O objetivo principal destes centros é agrupar diferentes ETM's de forma a atender as necessidades de transporte dos outros centros.

Por sua vez, os centros de trabalho de armazenamento são compostos por equipamentos de armazenamento de material que não fazem parte de outros centros de trabalho em particular.

3.2.3 Estrutura do Controlador

O controlador, independentemente do nível de controle, tem habilidade para receber informações, tomar decisões e trocar informações com os outros controladores. A função de tomada de decisões corresponde as funções de *planejamento* e *escalonamento*, enquanto que o recebimento e geração de informações corresponde à função de *execução*. As saídas do planejamento servem como entrada do escalonamento, as saídas do escalonamento servem como entrada para a execução, e as saídas da execução gerenciam os equipamentos físicos do chão-de-fábrica.

Neste trabalho, a preocupação maior é com o detalhamento e a descrição da função de *execução* dos controladores de centro de trabalho e de equipamento. Este detalhamento é independente das funções de planejamento e escalonamento. A função de execução é desenvolvida com base no sistema físico ou no seu modelo. As funções de planejamento e escalonamento não dependem apenas da função de execução mas também dos requisitos de produção. Desta forma, as necessidades do

planejamento e do escalonamento mudam conforme os requisitos de produção. No caso da função de execução, as mudanças só acontecem quando da modificação da configuração da célula ou do centro de trabalho. Visto que no ambiente de manufatura os requisitos de produção são mudados mais frequentemente que a configuração dos centros, a separação explícita das funções de planejamento e escalonamento da função de execução e o desenvolvimento de interfaces bem definidas entre estas funções faz com seja possível a conexão de diferentes funções (estratégias) de planejamento e escalonamento com a função de execução.

A função de execução do controlador de centro de trabalho recebe as mensagens de controle, necessárias para processar as peças e gerenciar os recursos, e envia mensagens detalhadas para a função de execução dos controladores de equipamento onde, para cada mensagem recebida, é especificada uma sequência de ações a serem tomadas. A função de execução dos controladores (centro de trabalho e equipamento), comunica-se com outros controladores e também com as funções de escalonamento e planejamento. A figura 3.4 mostra o fluxo de controle na arquitetura adotada. A função de execução pode ser detalhada em três sub-funções: 1) monitoração de mensagens de entrada, 2) execução de ações, e 3) geração de mensagens de saída. A sub-função de monitoração recebe mensagens dos outros controladores e as identifica para posterior execução de ações. Por exemplo, uma ação para um controlador de máquina pode ser a carga/descarga de peças, processamento, etc. A sub-função de geração de mensagens de saída envia mensagens (controle e informação) resultantes da execução de ações para os outros controladores. Desta forma, o controlador é um dispositivo reativo que responde a mensagens recebidas de controladores externos, a partir da interpretação das informações e da tomada das ações determinadas.

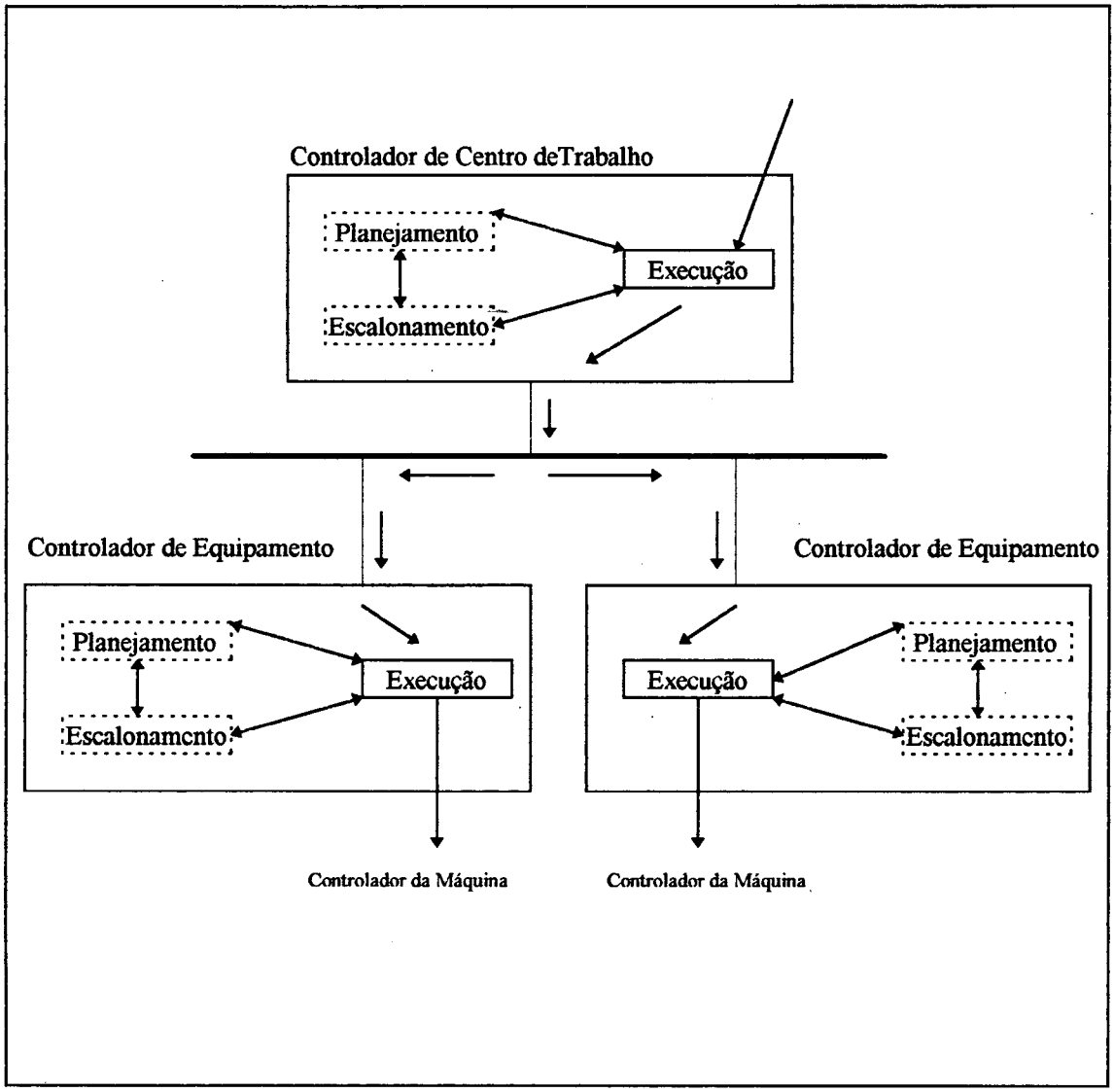


Figura 3.4 : Fluxo de controle na arquitetura adotada.

3.3 Estrutura de desenvolvimento do Software de Controle

Nesta seção é apresentada uma estrutura para o desenvolvimento de software de controle de chão-de-fábrica que tem como objetivo proporcionar a obtenção de características como reusabilidade e extensibilidade. Dois aspectos são importantes nesta abordagem: a arquitetura do software e a utilização de um formalismo para a modelagem. A arquitetura particiona o software de controle em componentes, enquanto que o formalismo permite representar de forma precisa as caraterísticas dos componentes do sistema.

3.3.1 Arquitetura do Software

Com o objetivo de proporcionar o desenvolvimento do software de controle de uma forma genérica, através da utilização de características como reusabilidade e extensibilidade, e também definir a arquitetura de software a ser utilizada, primeiro vamos analisar a natureza das modificações que envolvem um sistema de manufatura. Dois tipos de modificações podem ser identificados: 1) modificações dependentes do sistema, que incluem aquelas que dizem respeito a equipamentos ou capacidades/restrições do sistema, e 2) modificações dependentes da aplicação, que incluem aquelas que dizem respeito à forma como o sistema é usado. Assim sendo, podemos particionar o software de controle da forma como foram classificadas as modificações do sistema, ou seja, componentes dependentes do sistema que implementam as tarefas que estão associadas com as capacidades do sistema (independente da aplicação) e componentes dependentes da aplicação que implementam tarefas que estão associadas com aplicações em particular.

Por exemplo, no centro de trabalho de processamento mostrado na Figura 1.2 à direita, pode haver a seguinte restrição: o robô R2 primeiro necessita retirar a peça

que está na máquina CN 2 para, depois, retirar a peça que está na máquina CN 1 e colocá-la na máquina CN 2. Uma das tarefas do software de controle é assegurar que este procedimento será seguido. Esta tarefa diz respeito às capacidades/restrições do sistema, e é independente de qualquer aplicação na qual o sistema é usado. Por outro lado, para executar uma determinada operação de manufatura, os equipamentos e processos no sistema são coordenados por um plano de processo (figura 3.1). Considere o de sistema de manufatura da Figura 1.2; peças diferentes podem estar sendo processadas pelos centros, sendo que cada uma tem o seu plano (aplicação). Os planos podem ser completamente diferentes e serem implementados de forma concorrente. Entretanto, eles não precisam saber detalhes sobre os equipamentos e processos envolvidos.

A especificação do software de controle de um sistema, utilizando o conceito de componentes dependentes de sistema e componentes dependentes de aplicação, permite que, quando a interface dos componentes dependentes do sistema é especificada formalmente e é representativa, novos componentes dependentes da aplicação possam ser incluídos e/ou componentes existentes possam ser modificados. Neste caso, não será necessário modificar os componentes dependentes de sistema existentes. Por exemplo, se desejarmos usar o sistema definido na Figura 1.2 para produzir uma nova peça, podemos desenvolver um plano para a mesma e implementá-lo como um componente dependente da aplicação, utilizando os mesmos componentes dependentes do sistema. Da mesma forma, quando as capacidades do sistema não são reduzidas (de forma a afetar os componentes dependentes da aplicação), modificações nos componentes dependentes do sistema podem ser feitas sem necessitar modificações nos componentes dependentes da aplicação existentes.

Os componentes dependentes do sistema, em um sistema de chão-de-fábrica, podem ser vistos como sendo compostos por duas partes principais: a parte funcional e a parte de controle, como mostra do na Figura 3.5.

- A parte funcional engloba o conjunto de recursos funcionais (máquinas, equipamentos, mão de obra) que são utilizados para executar as atividades necessárias para armazenar ou transformar as matérias primas (peças).

- A parte de controle assegura o sequenciamento e condiciona a execução de operações definidas na parte funcional do sistema.

Uma forma de abstração das partes funcional e de controle de um sistema (ou componente) é a utilização de componentes de software ou, de uma forma mais geral, componentes de software/hardware [NAY87]. Cada entidade no sistema é encapsulada em um componente software/hardware, que é composto de uma interface de software, uma interface de hardware (componentes de software não têm uma interface de hardware), e ações ou operações internas (controle interno). A interface de software especifica a interação, a nível de software, que o componente implementa e representa a forma de acesso para os serviços que o componente pode realizar. A interface de hardware define os pontos onde as interações físicas do sistema acontecem. As operações internas representam a abstração das interações existentes entre software e hardware e são privativas a cada componente software/hardware. Desta forma, um sistema e seu software de controle podem ser construídos como sendo uma montagem dos componentes software/hardware de cada entidade, assim como dos componentes dependentes de aplicação, permitindo auto-documentação e desenvolvimento incremental.

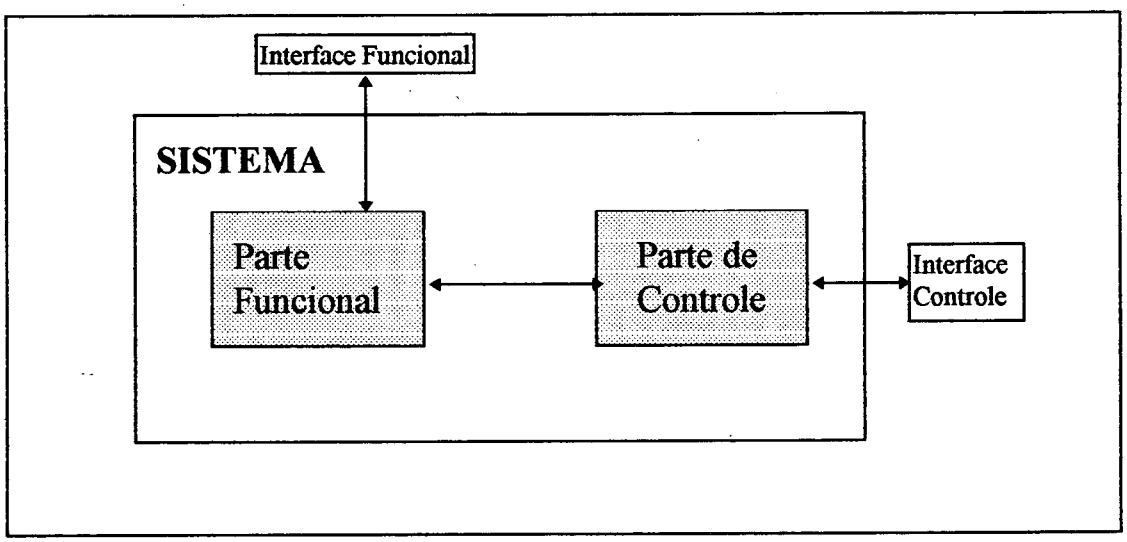


Figura 3.5 : Decomposição de um componente dependente de sistema.

A Figura 3.6 mostra a arquitetura de software adotada. Neste caso, o software de controle interage com os equipamentos e processos do sistema, através das interfaces de hardware dos componentes dependentes do sistema (componentes hardware/software). Os componentes dependentes da aplicação interagem com o sistema através da interface de software dos componentes dependentes do sistema. Considerando a interação existente entre os componentes da arquitetura de chão-de-fábrica adotada (Figura 3.4), o controlador de centro de trabalho relaciona-se com os controladores de equipamentos através da interface de software enquanto que os controladores de equipamento, através da interface de hardware, interagem com o sistema físico. Os componentes dependentes da aplicação interagem com os componentes dependentes do sistema (controlador de centro de trabalho). Desta forma, os componentes dependentes da aplicação podem se relacionar com vários controladores de centro e um controlador de centro pode estar executando mais de um componente dependente de aplicação.

Com o objetivo de melhorar a característica de reusabilidade dos componentes software/hardware, três aspectos são importantes. Em primeiro lugar, a interface de software deve ser construída da forma mais geral possível. Isto quer dizer que a interface de software de uma entidade deve fornecer uma abstração da forma com que a interação com outras entidades acontece, sendo que qualquer característica específica da entidade deve ser escondida do usuário. Segundo, o componente software/hardware deve ser auto-contido, ou seja, sem conexão com os usuários do componente, de forma que não seja necessário para o componente conhecer o nome dos usuários. Por último, os componentes software/hardware devem ser tão pequenos quanto possível, ou seja, os componentes devem conter o suficiente para serem auto-contidos mas não devem conter nada mais do que o necessário.

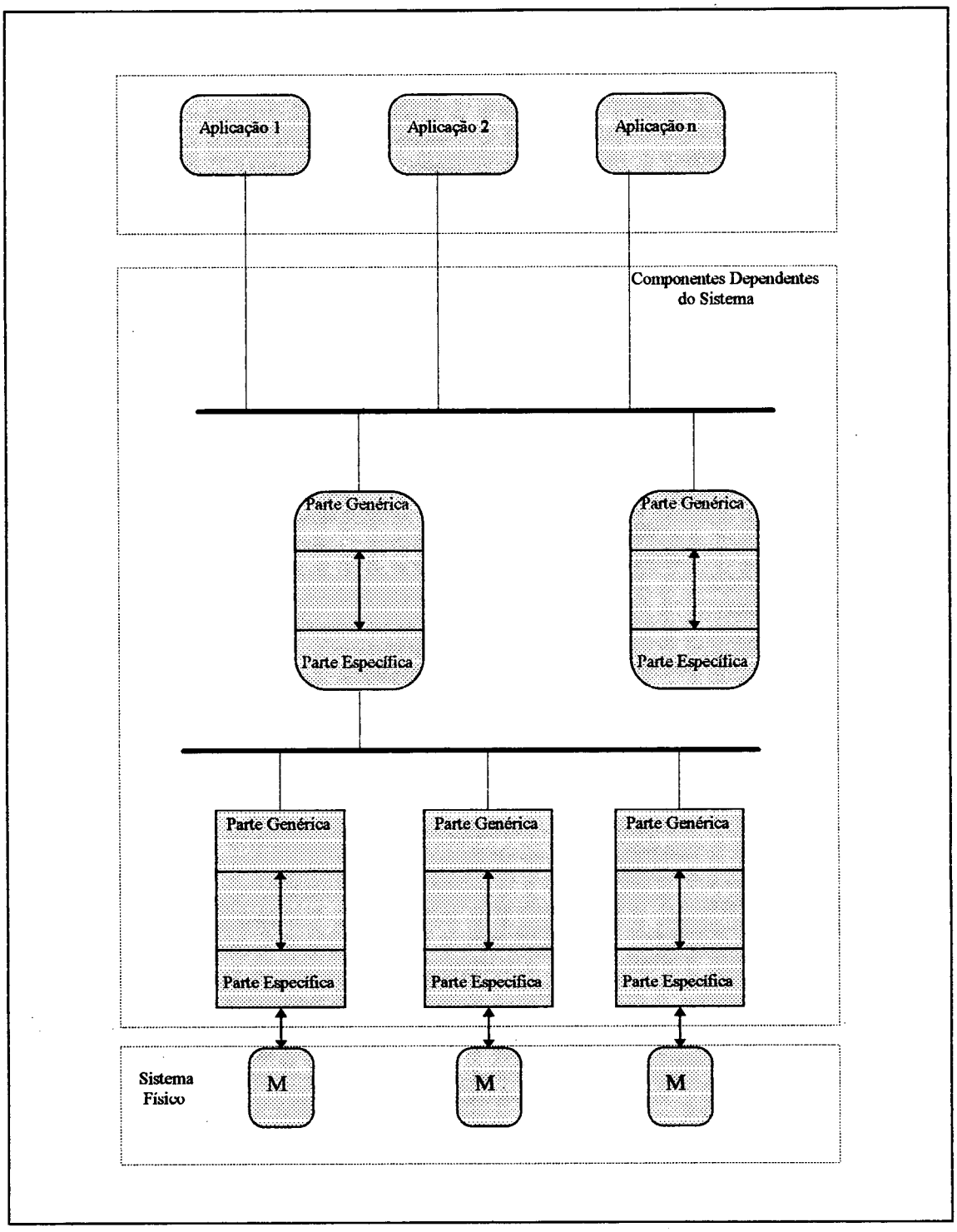


Figura 3.6 : Arquitetura do software de controle.

Para atingir os objetivos propostos no que diz respeito ao desenvolvimento do software de controle é proposta uma estrutura para o software de controle que está fundamentada na especificação das partes funcional e de controle dos componentes. Cada componente do sistema é representado pela sua estrutura funcional (interface de software) e pelo seu comportamento (operações internas).

3.3.2 Representação dos componentes do sistema

A representação dos componentes do sistema leva em consideração a estrutura e o comportamento dinâmico dos mesmos. A estrutura é caracterizada pelos atributos (estáticos e dinâmicos) e pelas operações individuais. O comportamento define como cada subsistema executa suas operações.

Inicialmente, com o objetivo principal de formalizar o conhecimento sobre o sistema observado, em termos de componentes e relacionamentos, utiliza-se o *modelo de informações*. Por exemplo, considere a arquitetura do controle de chão-de-fábrica mostrada na Figura 3.2. Os relacionamentos entre os componentes do centro de trabalho de processamento (CtP) são representados no modelo de informações, como mostra a Figura 3.7. Os relacionamentos mais significativos são : 1) o controlador de centro de processamento (CtP) controla um robô e três máquinas, 2) o robô serve as três máquinas e o buffer de saída, e é servido pelo buffer de entrada a partir da movimentação de peças que são processadas pelas máquinas, 3) o CtP comunica-se com o controlador de chão-de-fábrica (supervisão global) e com outros controladores de centro de trabalho, 4) as peças são processadas pelas máquinas e movimentadas pelo robô, sendo que as mesmas podem estar nos buffers de saída ou de entrada . Os relacionamentos mostrados no modelo de informações servem para apontar as interações que existem entre os componentes do sistema, caracterizando assim o fluxo de comunicação e controle

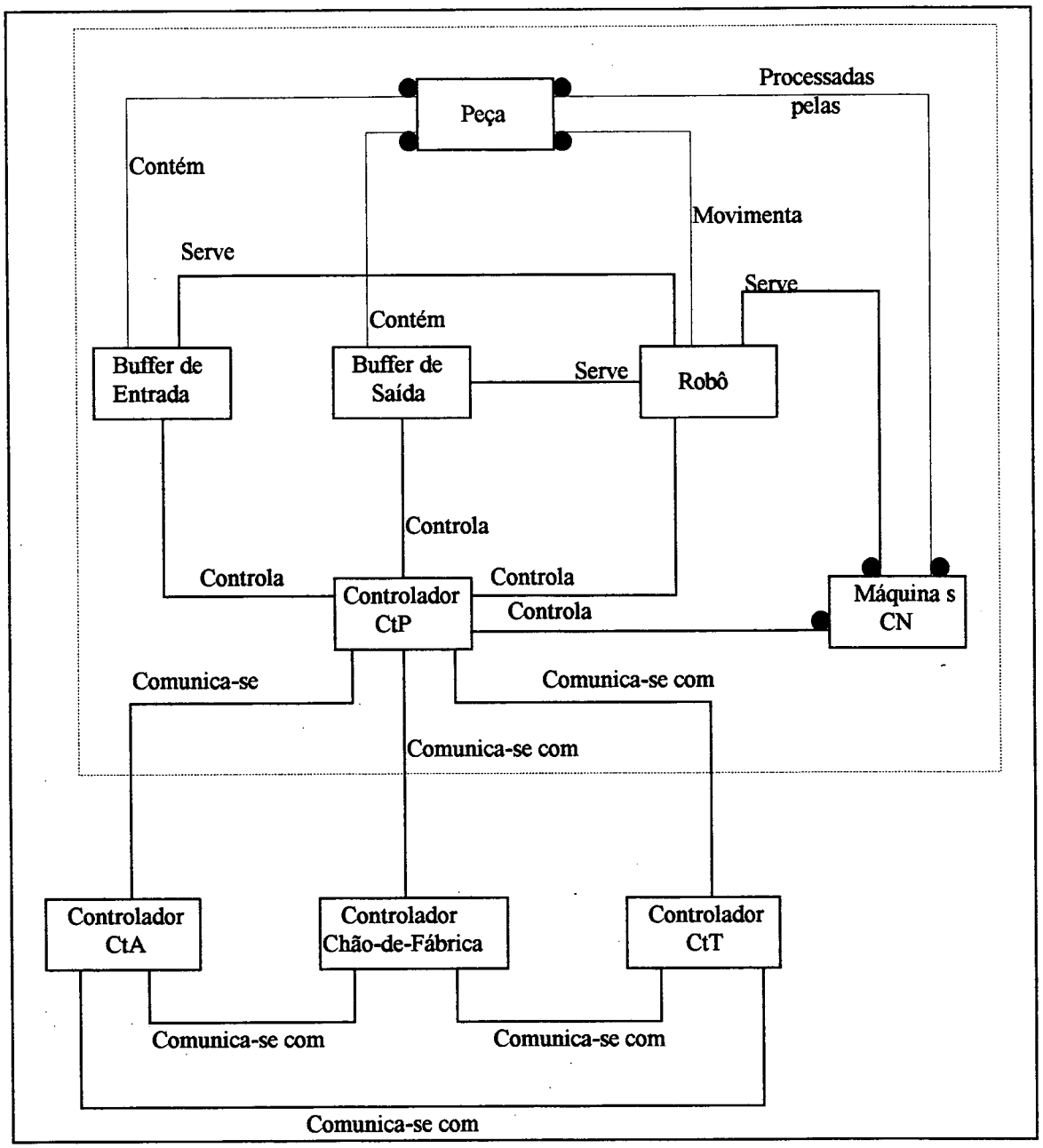


Figura 3.7 : Modelo de informações de um sistema.

existente entre os componentes. Desta forma, o modelo de informações nos permite identificar os componentes existentes no sistema e os seus respectivos relacionamentos.

Após a identificação dos componentes do sistema, a estrutura de cada um deles é então descrita através do modelo funcional, como mostra a Figura 3.8. O modelo funcional descreve os componentes individualmente, através da especificação dos seus atributos, operações e relacionamentos. Um componente é definido por:

C = { Cl, At, Op, Re } , onde :

Cl é a classe do componente,

At = $At_e \cup At_d$ é o conjunto de atributos do componente, onde :

At_e é o conjunto de atributos estáticos, características que não mudam seus valores durante a operação, e

At_d é o conjunto de atributos dinâmicos que mudam seus valores durante a operação (estado do componente).

Op = $Op_e \cup Op_p \cup Op_s$ é o conjunto de operações do componente, onde :

Op_e é o conjunto de operações que tratam eventos de entrada que recebem comandos ou requisições de outros componentes,

Op_p é o conjunto de operações de processamento que executam tarefas de controle, e

Op_s é o conjunto de operações que tratam eventos de saída através da comunicação com outros componentes.

Re é o conjunto de outros componentes (recursos) que interagem com o componente sendo descrito.

Classe do Componente : MÁQUINA_CN

Atributos:

Maq_Status	(estado da máquina)
Maq_Id	(identificador da máquina)
Peça_Info	(informação da peça na máquina)
Maq_Local	(localização da máquina)
Maq_N_peças	(número de peças na máquina)

Operações :

Entrada :

Maq_Carga ( ) ; carrega peça na máquina com o auxílio de um EMM
Maq_Descarga ( ); descarrega peça da máquina com o auxílio de um EMM
Maq_Executa ( ); inicia o processamento de uma peça
Pronto_colocar ( ); máquina pode colocar peça
Livre_trabalho ( ); espaço de trabalho da máquina está livre
Peça_segura ( ); máquina pode soltar peça
Oper_fim ( ) ; máquina pode terminar processamento

Processamento:

Prog_CN ( ); máquina envia programa CN

Saída:

Fim_proc ( ); informa término de processamento da máquina
Planeja ( ); requisita tarefa de planejamento na máquina
Fixação_aberto ( ); informa abertura dispositivo de fixação da máquina
Fixação_fechado ( ); informa fechamento dispositivo de fixação da máquina
Início_op ( ); inicia operação na máquina

Recursos:

Controlador Ct : de quem a máquina recebe requisições de serviço
Controlador EMM : de quem a máquina usa serviços de carga/descarga
Planejamento : quem executa serviços de planejamento no nível da máquina
Escalonamento : de quem a máquina recebe informações para execução das peças

Figura 3.8 : Modelo funcional do componente controlador de uma máquina CN.

A representação do comportamento do sistema especifica como os seus elementos componentes participam na realização das tarefas. Cada elemento é caracterizado por seu ciclo de vida, que define de forma individual a sua participação no sistema. O objetivo da modelagem do comportamento é entender e documentar de que forma cada componente participa ou atua no sistema. Na modelagem do comportamento são registrados : 1) os estados do componente, 2) as condições e os eventos que fazem com que um componente mude de um estado para outro, 3) as ações que ele realiza e, 4) as ações realizadas nele. O comportamento dos componentes é representado com o uso das *redes de comportamento*. A rede de comportamento de um componente é formada por um conjunto de estados e um conjunto de atividades. Cada estado representa uma condição particular do componente, e cada atividade representa a realização de alguma operação no ciclo de vida do componente. A idéia principal das redes de comportamento é inspirada nas redes de Petri [PET81] [BRU86].

Cada componente é representado por uma rede autônoma que troca mensagens com os outros componentes do sistema. Se o componente é completamente independente ele é chamado *componente fechado*. Quando o componente recebe informações de outros componentes e envia informações para outros componentes ele é chamado de *componente aberto*. Um componente aberto tem estados ou lugares especiais que são chamados *canais de Entrada/Saída*. Um canal de entrada recebe informações de fora do componente, enquanto que um canal de saída envia informações para fora do componente. O conjunto de canais de E/S do componente define a sua interface. A comunicação entre componentes é estabelecida pela ligação do canal de saída do enviante ao canal de entrada do receptor.

Uma rede de comportamento é um grafo dirigido **Rc = (E, C, A)**, onde:

E é um conjunto finito de condições (lugares internos)

C é um conjunto de canais externos, $C = C_e \cup C_s$ onde

C_e é o conjunto de canais de entrada, e

C_s é o conjunto de canais de saída

A é o conjunto de atividades.

A cada atividade em **A** corresponde um par (Ee, Es), onde Ee é o conjunto de estados de entrada e Es é um conjunto de estados de saída da atividade. O conjunto Ee é um subconjunto não vazio de $C_e \cup E$, com no máximo um elemento de C_e. Se $Ee \cap C_e \neq \varnothing$ a atividade é dita global ou externa, caso contrário é chamada local ou interna. O conjunto Es é um subconjunto de $C_s \cup E$. Para cada atividade em **A** está associada uma sequência de ações. A sequência de ações determina como são afetados os estados de saída correspondentes à atividade.

Uma atividade é executada (disparada) quando o seu conjunto de estados de entrada está marcado. Neste caso, o conjunto de estados de saída é afetado de acordo com a execução das ações, atuando nos canais de saída, através do envio de eventos (sinal, mensagem) para um outro componente, e modificando os estados internos. Os estados internos de um componente (subsistema), na rede de comportamento, são representados por retângulos arredondados. Uma atividade é representada por um retângulo, com o nome da atividade e a especificação das ações associadas, que é conectado por um conjunto de arcos aos pré-estados e pós-estados da atividade. O evento externo associado a uma atividade também é descrito na atividade; o sinal **?** indica um evento de entrada, enquanto que o sinal **!** indica um evento de saída. Os canais são representados por nós quadrados que se ligam às atividades. Um círculo em um canal de comunicação representa um canal de entrada enquanto que um losângulo representa um canal de saída. Os canais

podem ser conectados de forma dinâmica, o que representa uma maior flexibilidade e portabilidade do software de controle.

A figura 3.9 mostra a modelagem de uma das operações do componente controlador de equipamento máquina CN. Por exemplo, no estado *L.Pronto_Desc* o equipamento de processamento (EMP), através de seu controlador, está pronto para a descarga de uma peça e a ocorrência do evento *Maq_Descarga* faz com que a máquina inicie a descarga. Assim que o equipamento de movimentação de material (EMM) estiver pronto para pegar a peça, ele envia o sinal *Peça_Segura* que faz com que o equipamento de processamento execute a ação de abertura do dispositivo de fixação com o objetivo de liberar a peça. A realização desta ação implica no envio do sinal *Fixação_aberto* informando o EMM que a peça está solta.

A modelagem do comportamento dos componentes (subsistemas) a partir da utilização das redes de comportamento permite descrever aspectos importantes, nos sistemas de manufatura, como o paralelismo (duas ou mais tarefas independentes) e a sincronização entre atividades de diferentes redes. As redes permitem ainda uma boa visualização do fluxo de mensagens existente entre os componentes de um sistema de manufatura.

Através de suas atividades, uma rede de comportamento identifica as operações públicas do componente (serviços que o componente fornece para os outros componentes do sistema) e, devido à semântica das redes de Petri, determina as possíveis sequências em que elas podem ser aplicadas.

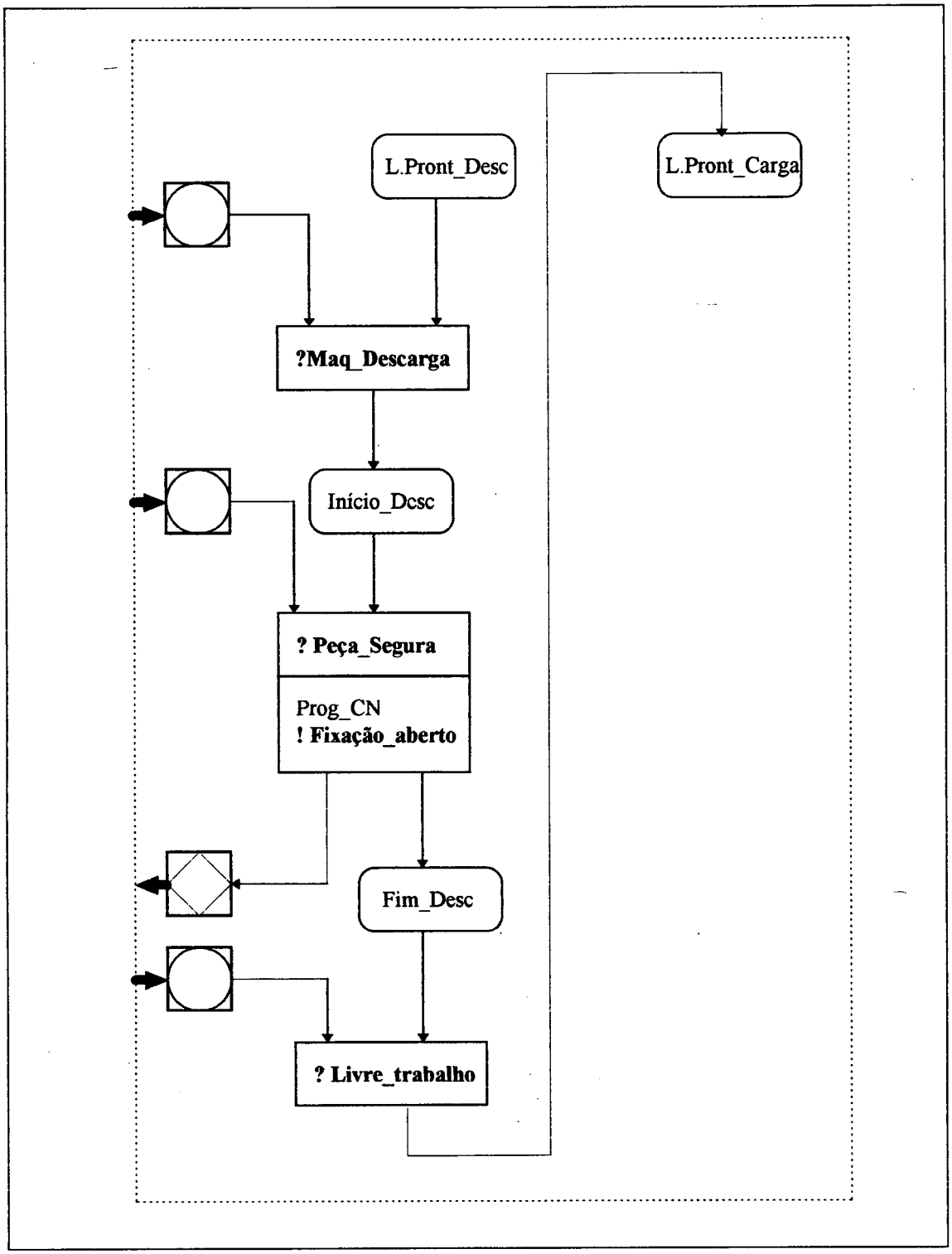


Figura 3.9 : Modelagem da operação de Descarga de uma máquina CN.

O comportamento dos componentes também pode ser especificado a partir da utilização de uma forma textual [TOS93]. Neste caso, os componentes são descritos a partir da representação textual da rede. A descrição textual de um componente poderia ter a seguinte forma:

componente C;
[entrada : E,
saída : S,
interno : I,
var : V,
inicialização : F,
A
];

onde:

E = { e_1, e_2, ..., e_i }, i≥0, é a lista de eventos de entrada;

S = { s_1, s_2, ..., s_j }, j≥0, é a lista de eventos de saída;

I = { i_1, i_2, ..., i_k }, k≥0, é a lista de eventos internos;

V = { v_1, v_2, ..., v_l }, l≥0, é a lista de variáveis;

F = { f_1, f_2, ..., f_m }, m≥0, é a lista de comandos de inicialização;

A = {a_1, a_2, ..., a_n }, n≥0, é a lista de atividades.

Alguns comandos podem ser definidos para tornar a descrição mais formal, por exemplo :

v:= *expressão*

interno(*evento-interno*)

externo(*evento-externo*)

O primeiro é utilizado para atribuir o valor de uma expressão a uma variável, o segundo é utilizado para ativar algum evento interno (condição) e o terceiro é utilizado para enviar eventos externos (mensagens). As atividades têm a seguinte forma geral:

$$e_e \ \& \ [e_1, e_2, ..., e_n] \Rightarrow \text{ação}$$

onde e_e é um evento de entrada e e_i ($1 \leq i \leq n$) é um evento interno. Ação especifica uma lista de ações que são realizadas em função da atividade disparada, incluindo o envio de eventos externos e a ligação de eventos internos.

A Figura 3.10 mostra um exemplo da descrição textual de parte do componente máquina CN.

A partir da especificação das redes que representam o comportamento dos componentes do sistema de chão-de-fábrica, é possível identificar como estes subsistemas interagem para a realização de suas atividades. As interações entre as redes de comportamento dos componentes podem ser representadas através do *modelo de comunicação*. Este diagrama mostra os eventos de entrada e de saída de cada rede de comportamento dos componentes do sistema, sem mostrar os eventos internos.

O modelo de comunicação caracteriza as *conexões* existentes entre os componentes. A figura 3.11 mostra um exemplo do modelo de comunicação envolvendo o componente controlador de máquina CN.

```
componente máquina_CN

[ entrada : [..., Maq_Descarga,
                  Peça_Segura,
                  Livre_trabalho, ... ],
  saída : [..., Fixação_aberto, ...],
  interno : [..., L.Pront_Desc, L.Pront_Carga,
                 Inicio_Desc, Fim_Desc, ...],

  inicialização : [ ..., interno(L.Pront_Desc), ...],

  ...
  Maq_Descarga & L.Pront_Desc  ==> [ interno( Inicio_Desc) ],
  Peça_Segura & Inicio_Desc  ==> [ Prog_CN, externo( Fixação_aberto),
                                              interno( Fim_Desc) ]
  Livre_trabalho & Fim_Desc  ==> [ interno( L.Pront_Carga) ]
  ...
].
```

Figura 3.10 : Descrição textual de parte do componente máquina CN.

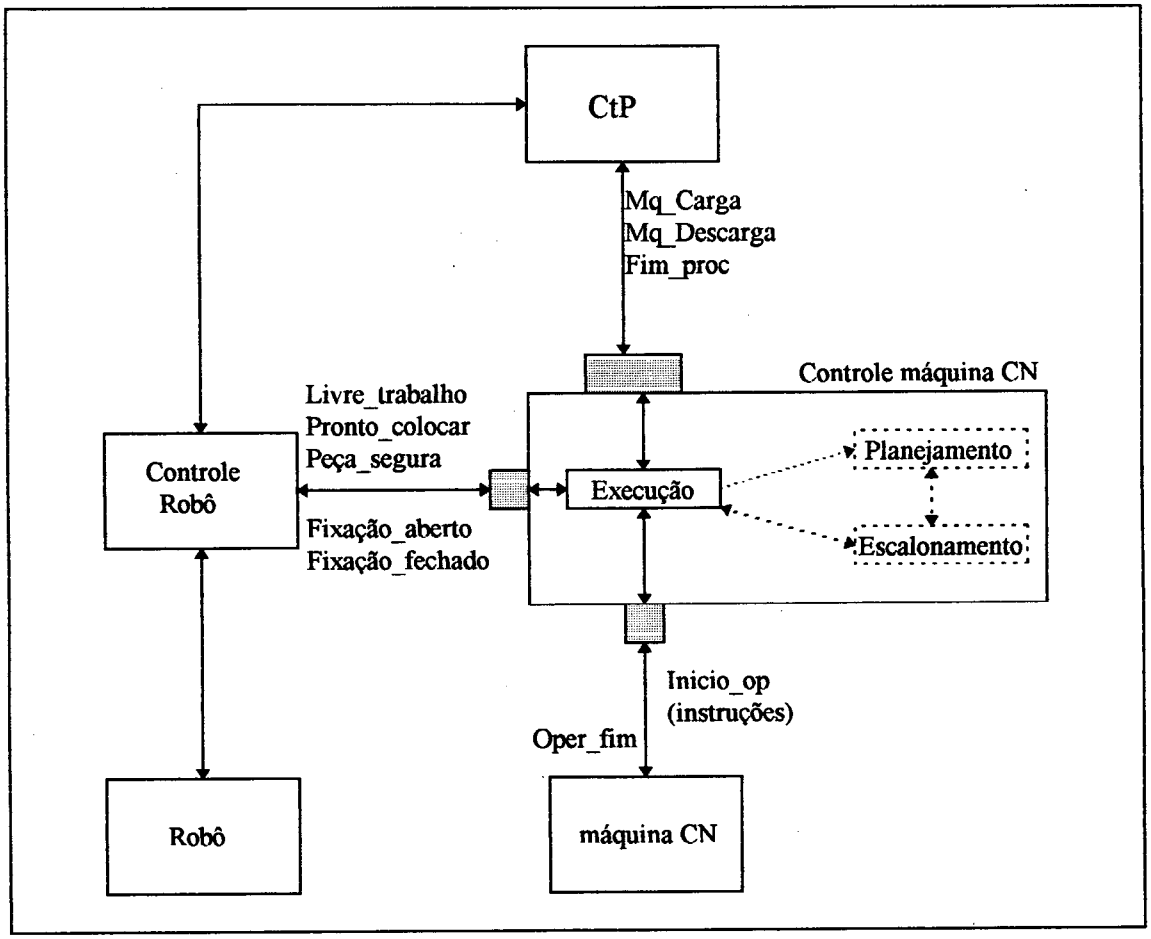


Figura 3.11 : Modelo de comunicação do controlador de máquina CN.

3.4 Modelo de Implementação

Considerando o modelo de comportamento e a arquitetura do software proposta, onde um sistema é visto como um conjunto de componentes (sub-sistemas) que interagem para a realização de uma determinada tarefa, o modelo de implementação tem como meta principal fornecer o suporte necessário para a implementação do software de controle. Este suporte deve permitir a visão do sistema como um conjunto de componentes (objetos), fornecendo os mecanismos necessários para a implementação do modelo.

A partir da utilização do paradigma da orientação a objetos a implementação é vista como um conjunto de objetos (controladores) que se comunicam (através da passagem de mensagens) para a realização de tarefas. Esta visão pode ser implementada de duas maneiras, como mostra a Figura 3.12.

A primeira considera que existe apenas uma linha de controle (processo) de execução, onde os objetos são considerados como entidades passivas e o controle passa de um objeto para outro. As comunicações entre os objetos são realizadas a partir da passagem de parâmetros entre funções. O modelo computacional neste nível é altamente abstrato e não existe uma execução paralela verdadeira dos objetos.

Na segunda abordagem existem mais de uma linha de controle de execução, sendo que os objetos que representam os componentes do sistema são processos ativos que estão habilitados a se comunicar a partir de facilidades criadas para permitir a comunicação entre processos. O modelo de implementação proposto utiliza esta segunda abordagem. Assim sendo deve suportar os conceitos estabelecidos no modelo de especificação, tais como: componentes e canais de

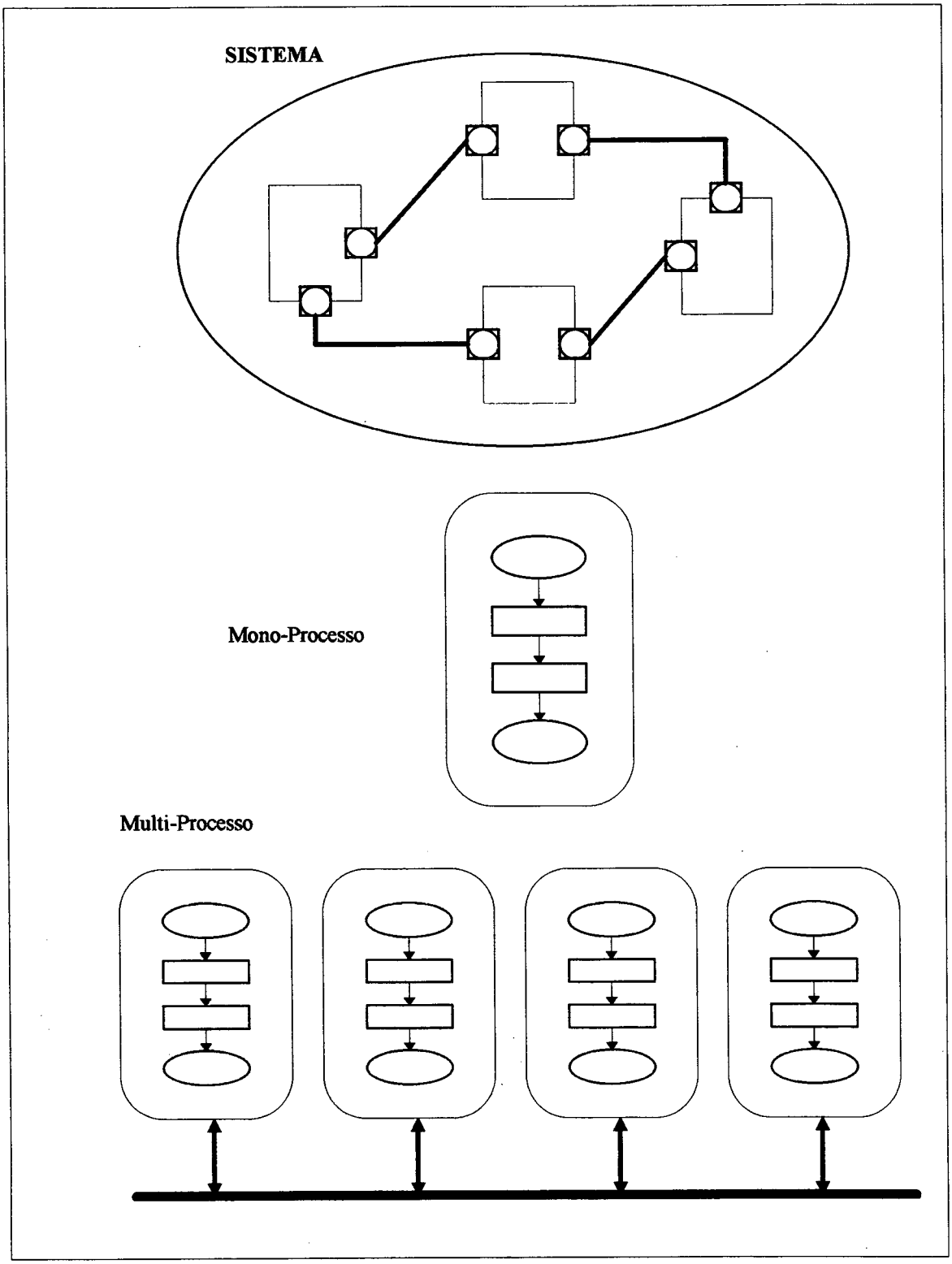


Figura 3.12 : Modelo de Implementação Multi-Processo.

comunicação. Com esta abordagem o modelo de controle especificado é mais facilmente representado computacionalmente.

3.4.1 Controle e Serviços do modelo de implementação

No modelo de implementação adotado o suporte de implementação é composto por um sistema de serviços e um sistema de controle. O sistema de serviços é responsável pelo provimento de serviços como, por exemplo, os serviços de acesso a dados. O sistema de controle é responsável pela criação e destruição dos componentes (gerência de processos), comunicação entre processos e a gerência de conexões entre os processadores.

As primitivas de comunicação e gerência de conexões são responsáveis pela construção e destruição dinâmica dos canais de comunicação, assim como o transporte e entrega das mensagens.

A entrega das mensagens é responsabilidade do serviço de comunicação e eventualmente a mensagem é colocada em uma fila do componente (processo) destino. Quando esta mensagem for atendida pelo componente destino ela pode resultar no disparo de uma atividade. Entretanto, se o estado corrente do componente destino não tem previsão da mensagem ela é descartada sem indicação de erro.

Cada processador da arquitetura do modelo de implementação contém um componente com a seu respectivo comportamento e os serviços necessários para a execução das funções pertinentes ao componente. A Figura 3.13 mostra a arquitetura do modelo de implementação adotada neste trabalho.

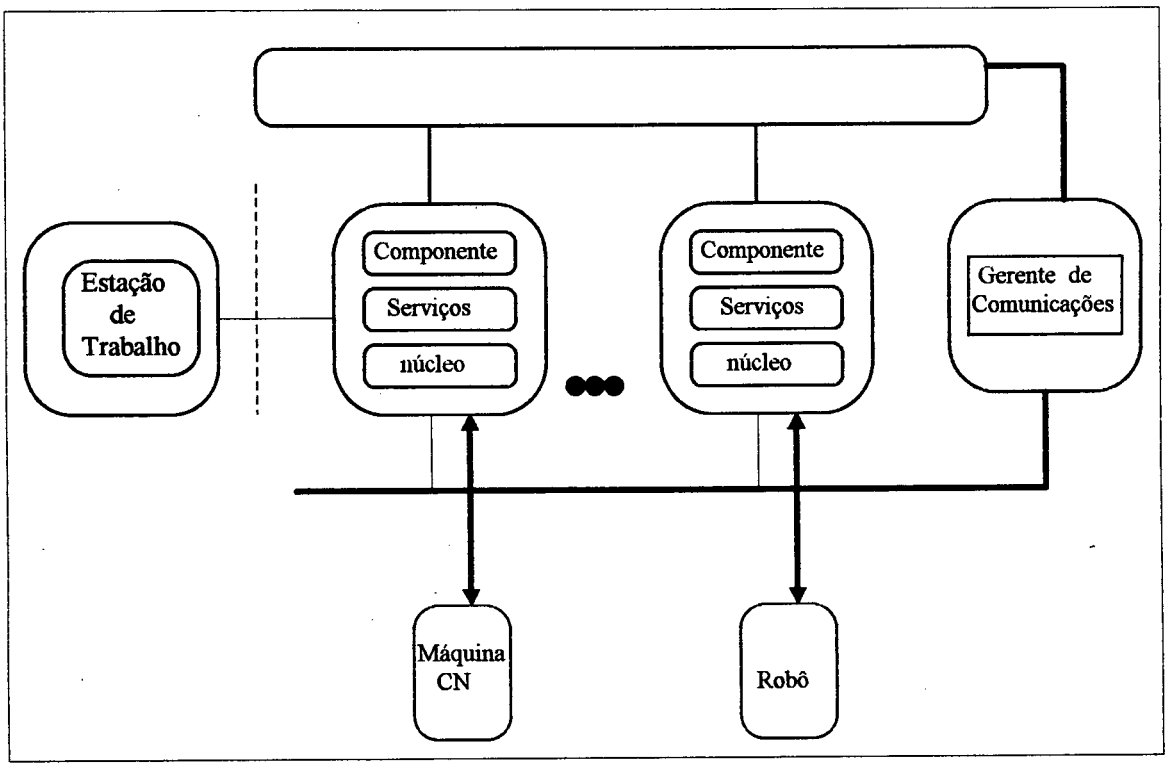


Figura 3.13 : Arquitetura do modelo de implementação.

A comunicação entre os componentes pode se dar de forma assíncrona ou síncrona. Na comunicacão assíncrona, o enviante passa uma mensagem para um componente e imediatamente continua sua execução. Uma mensagem assíncrona não retorna nenhum resultado para o seu enviante. A comunicação síncrona consiste de uma troca de mensagens controlada entre enviante e receptor. A mensagem é carregada pelo serviço de comunicação para o componente destino da mesma forma que no caso da comunicação assíncrona. Esta mensagem é então colocada na fila do destinatário onde, quando for tratada, poderá disparar uma atividade no comportamento do destinatário. Entre as ações associadas com esta atividade deve estar uma que realiza uma operação de resposta, que envia uma mensagem de volta para o enviante através do canal de comunicação no qual o enviante está esperando a resposta. A mensagem de resposta tem precedência sobre qualquer outra mensagem que possa estar na fila do enviante, ou seja, a mensagem de resposta tem prioridade maior que qualquer outra mensagem. Quando a mensagem de resposta é recebida, o remetente é desbloqueado e continua a sua execução.

Uma das funções principais de um controlador de centro de trabalho (célula) é a de supervisionar um conjunto de equipamentos de chão-de-fábrica que foram agrupados com o objetivo de otimizar a produção de uma *família*[1] de peças específica. Para que os benefícios da manufatura celular sejam alcançados, o controlador deve ser flexível. Esta flexibilidade apresenta alguns requisitos, entre eles um ambiente apropriado que permita a reconfiguração da lógica de controle de um controlador de centro de trabalho com o objetivo de acomodar modificações de uma *família* de peças para outra. Este ambiente constitui a interface do usuário.

[1] família no sentido da Tecnologia de Grupo

3.4.2 Interface do usuário

A definição da interface do usuário tem como objetivo principal permitir que o usuário possa configurar o centro de trabalho e construir a lógica de controle específica para cada aplicação do centro de trabalho. A partir da especificação da estrutura e comportamento genéricos dos componentes de chão-de-fábrica, a interface fornece ao usuário as seguintes capacidades:

a) **Configuração do centro de trabalho** : O usuário tem acesso a uma biblioteca de *componentes* que representam os dispositivos de chão-de-fábrica e o seu conjunto de operações. Esta biblioteca permite que o usuário possa selecionar os componentes apropriados para uma determinada configuração de centro de trabalho, permitindo assim uma rápida prototipação e configurabilidade. Adicionalmente o usuário pode definir o *layout* do centro de trabalho. No processo de configuração do centro de trabalho, o usuário define os nomes dos dispositivos e para cada um seleciona um dos componentes genéricos disponíveis que servirá de controlador do dispositivo definido. Um mecanismo de especificação das *ligações* existentes entre os dispositivos definidos também é fornecido. Estas ligações indicam a forma como os dispositivos estão conectados, a partir dos canais de entrada/saída.

b) **Especificação da lógica de controle** : Durante a execução, o controlador de centro de trabalho coordena os seus componentes de acordo com os planos de processo. Um conjunto de ferramentas permite que o usuário especifique a sequência da lógica de controle. A especificação da lógica de controle determina a sequência de operações executadas pelos dispositivos (mensagens enviadas pelo controlador de centro para os dispositivos). O plano de processo indica todos os passos a serem seguidos na elaboração de um produto no centro de trabalho. Em cada passo do plano de processo, uma operação é executada nos dispositivos associados. A representação de um plano de processo, sequência de dispositivos e

parâmetros correspondentes para os controladores, pode ser feita de forma gráfica (esquema de processo) . Desta forma, o plano de processo é visto como a representação interna de dados do esquema de processo. Cada passo do plano de processo pode conter informações como : identificação do passo, condições, nome do(s) dispositivo(s) envolvido(s) no passo, tipo de programa a ser executado pelos dispositivos, e próximo passo. Na especificação da lógica de controle, funções de teste, salto e gerenciamento de variáveis, são importantes no sentido de fornecer ao usuário a possibilidade de testar condições para possíveis saltos para diferentes partes da sequência de controle.

c) **Capacidades de depuração e execução** : A partir da especificação da lógica de controle o usuário deve estar habilitado para validação e execução da sequência de controle. Para isto, a interface de usuário deve fornecer capacidades de depuração e validação adequadas e de fácil utilização. Associado ao mecanismo de depuração é necessário um sistema interpretador que execute a lógica de controle a partir da ativação das funções necessárias nos dispositivos do centro de trabalho.

CAPÍTULO 4
MODELOS GENÉRICOS DOS CONTROLADORES

Como estabelecido no capítulo anterior, este trabalho está concentrado no desenvolvimento dos componentes de execução dos controladores de equipamento e de centro de trabalho. Este capítulo descreve os controladores no que diz respeito a sua estrutura básica e seus componentes de execução.

Considerando o modelo distribuído, o controle é estabelecido a partir da passagem de mensagens entre os controladores. Neste caso, as redes de comportamento (Rc) representam formalmente o protocolo existente entre os controladores na execução de tarefas. Este protocolo é caracterizado pela sequência de mensagens necessárias aos controladores para a realização das tarefas. Cada controlador (equipamento e centro de trabalho) tem uma rede de comportamento associada que descreve a sua funcionalidade.

4.1 O Controlador e suas partes componentes

O controlador, independentemente do nível de controle, engloba três funções básicas : planejamento, escalonamento (sequenciamento) e execução. Desta forma, o controlador apresenta três componentes principais : um planejador, um sequenciador e um executor. Visto que a separação do componente de execução dos componentes de planejamento/escalonamento é desejável, pois possibilita maior flexibilidade para o controle, a estrutura do controlador fornece :

- um modelo básico de referência, que descreve o sistema e as tarefas disponíveis
- uma interface entre os componentes de planejamento/escalonamento e o componente de execução.

O modelo de referência apresenta de uma forma geral os atributos, estáticos e dinâmicos, e as operações físicas genéricas do sistema. Os atributos estáticos descrevem as características das entidades específicas que são controladas pelo controlador. Por exemplo, atributos estáticos de um controlador de equipamento incluem : a capacidade do equipamento (número de peças que o equipamento suporta) e a localização do equipamento (para carga/descarga de peças) enquanto atributos dinâmicos informam : o estado do equipamento (operacional, quebrado, em manutenção) e as características da peça atualmente atribuída ao equipamento. A descrição do conjunto genérico de tarefas físicas que podem ser realizadas pelo controlador independentemente do estado corrente das máquinas, é apresentada pelo modelo das capacidades físicas.

A interface entre os componentes planejamento/escalonamento e o componente de execução permite que o escalonador especifique tarefas a serem realizadas. O componente de execução apenas realiza as tarefas sem saber de que forma a sequência das tarefas é determinada. Por outro lado, o escalonador não sabe de que forma as tarefas são realizadas. Esta interface é implementada a partir de uma política do tipo cliente/servidor, onde o escalonador é o cliente e o executor é o servidor. Esta interface permite uma separação explícita dos componentes escalonador e executor, possibilitando que qualquer tipo de escalonador (cliente) requisite serviços do componente de execução (servidor). A Figura 4.1 mostra o esquema de um controlador e a sua estrutura. As seções seguintes detalham o modelo de referência (capacidades físicas e atributos) e a interface.

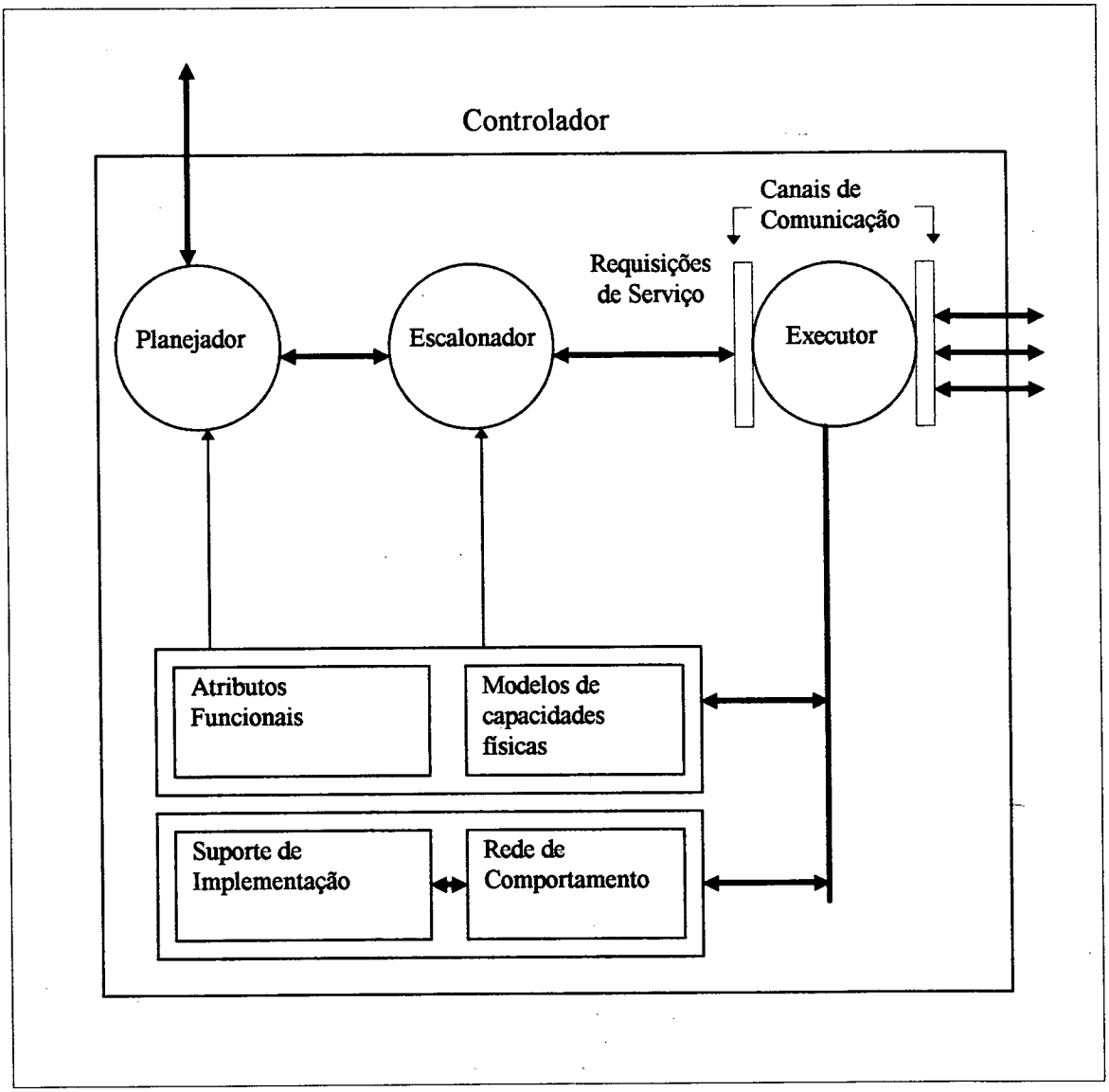


Figura 4.1 : Controlador e suas partes componentes.

4.1.1 O modelo de capacidades físicas

Um modelo genérico de capacidades físicas estabelece apenas o conjunto de tarefas possíveis de serem realizadas pelas entidades do sistema. Estes modelos não levam em consideração *pré-condições* de realização das tarefas, sejam elas físicas ou lógicas, nem consideram a necessidade de comunicação (troca de mensagens) que existe na realização de uma tarefa.

Por exemplo, considere um centro de trabalho composto por um robô e um torno. Uma das tarefas especificadas no modelo de capacidades físicas é o robô carregar uma peça no torno. O que não está sendo considerado neste modelo é que antes de realizar esta tarefa algumas *pré-condições* devem ser satisfeitas : 1) O torno deve estar pronto para receber uma peça e o robô deve estar livre (condições físicas); 2) Esta tarefa só faz sentido ser executada se o torno é uma das máquinas pela qual a peça deve passar, como estabelecido no seu plano de processo (condição lógica). Estas condições não são descritas no modelo de capacidades físicas. Entretanto, o modelo fornece uma visão genérica das capacidades do sistema. No exemplo acima, o modelo especifica que é possível para o robô carregar uma peça no torno. Estas informações são suficientes para os componentes planejamento/escalonamento trabalharem. Quanto às pré-condições, estas são especificadas pelos atributos e pelo próprio comportamento de cada entidade.

Um modelo de capacidades físicas é especificado por uma rede de Petri do tipo lugar/transição (place/transition net) onde, as transições representam as tarefas que a entidade realiza e os lugares representam os estados/localização de uma peça na entidade. Esta forma de representação está baseada em [MET89] [SMI92]. Cada controlador tem o seu próprio modelo de capacidades físicas.

4.1.1.1 Equipamentos

Considerando a arquitetura de controle adotada, os controladores de equipamentos são responsáveis pelo processamento das peças que são atribuídas aos centros de trabalho. A classe dos equipamentos de processamento (EPM) tem capacidade de processar peças de acordo com um conjunto de instruções de processamento. Neste caso, as peças são colocadas no equipamento, processadas pelo mesmo e retiradas do equipamento. Esta descrição configura o modelo de capacidades físicas no caso de um EPM, como mostra a Figura 4.2. Entretanto, não existe nenhuma informação sobre a capacidade de processamento da máquina, o estado atual da máquina, ou mesmo, o tipo de comunicação necessário para realizar as tarefas descritas. Estas informações dizem respeito aos atributos e ao comportamento da máquina.

Os equipamentos de movimentação (EMM) são responsáveis por mover peças dentro de um centro de trabalho. As peças são pegas em uma localização especificada e colocadas em uma outra posição, possivelmente diferente da primeira. Os locais especificados são considerados como pontos de entrada/saída de material que podem ser atribuídos aos equipamentos e centros de trabalho. A tarefa de pegar uma peça, no caso de um EMM, pode significar o deslocamento do equipamento até uma localização inicial, pegar a peça e deslocar-se até uma segunda localização chamada de intermediária. Enquanto que na tarefa de colocar a peça, o equipamento seria deslocado da posição intermediária até a posição final para largar a peça. Visto que um EMM realiza tarefas de carga/descarga de peças em outros equipamentos de um centro de trabalho, as tarefas de pegar e largar peça devem estar sincronizadas com tarefas semelhantes nos outros equipamentos. Por exemplo, a tarefa de *pegar* uma peça de um EMM está sincronizada com a tarefa retirar peça de um EPM. Este sincronismo será estabelecido na descrição do

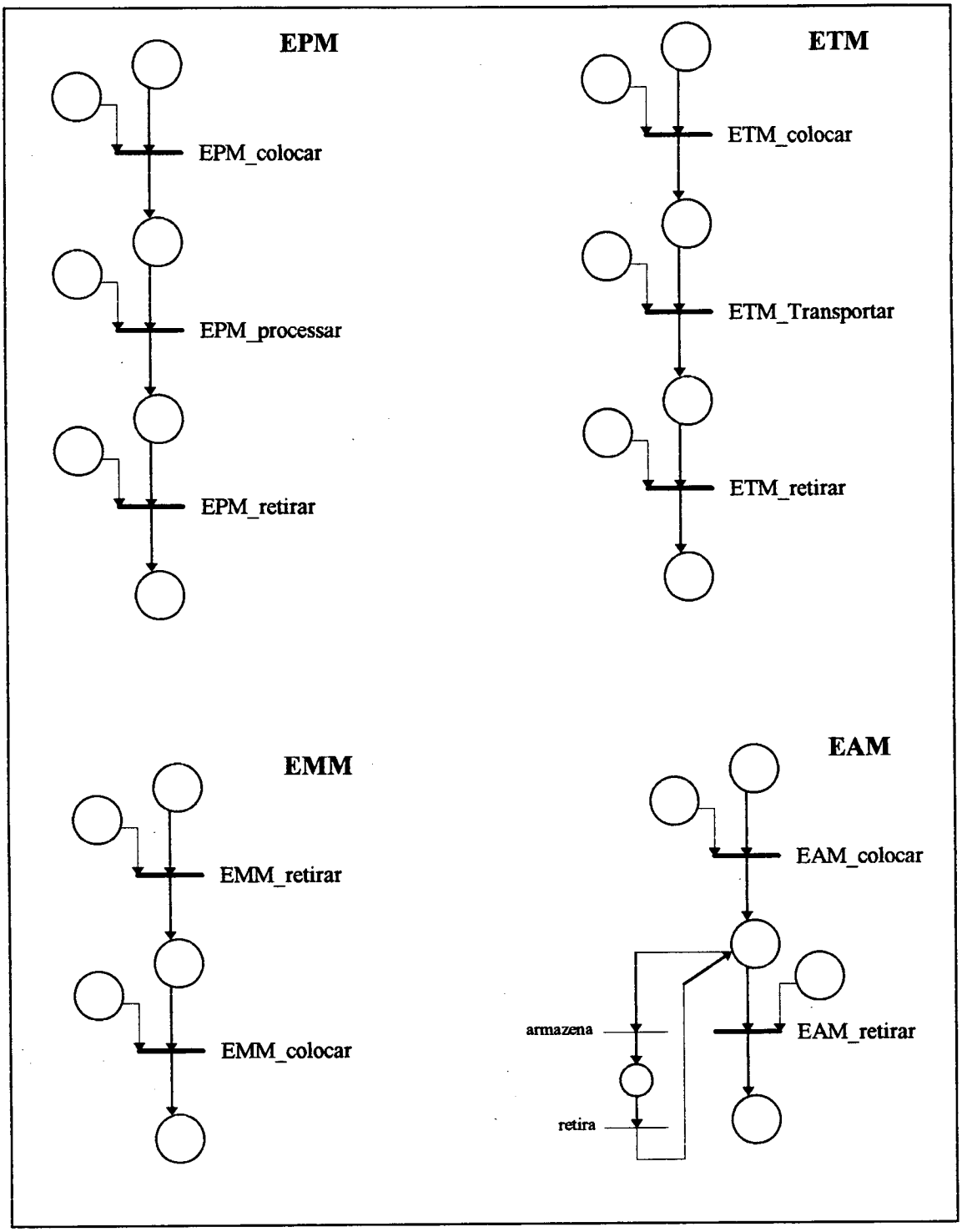


Figura 4.2 : Modelos de capacidades físicas dos equipamentos.

comportamento de cada um destes equipamentos. A Figura 4.2 mostra o modelo de capacidades físicas no caso dos EMM.

Os equipamentos de armazenagem (EAM) são responsáveis pelo armazenamento de peças. Neste caso, as peças são colocadas no equipamento e depois retiradas do mesmo. Internamente um EAM, de posse de uma peça, pode armazená-la em uma posição e depois recuperá-la. Finalmente, os equipamentos de transporte (ETM) são responsáveis pelo transporte de peças entre os centros de trabalho. Neste caso, as peças são colocadas no equipamento, transportadas para uma outra posição (ponto de E/S de material) e finalmente retiradas do equipamento. A Figura 4.2 mostra os modelos EAM e ETM.

4.1.1.2 Centros de Trabalho

Um centro de trabalho é composto por uma ou mais entidades do nível de equipamento. O modelo de capacidades físicas mostra as tarefas que um centro de trabalho pode realizar considerando a sua composição. Desta forma, cada tipo de centro de trabalho identificado anteriormente tem o seu próprio modelo de capacidades físicas. Basicamente, um modelo de capacidades físicas de um centro de trabalho pode apresentar lugares/locais que representam os diferentes tipos de equipamentos que compõem o centro. Por exemplo, num centro de trabalho de processamento (CtP) as peças chegam, usualmente, em um ponto de E/S de material, como mostra a Figura 4.3. Em seguida, estas peças podem ser retiradas pelo CtP, com a participação de um EMM. No centro, as peças podem ser colocadas e retiradas em/de um dos equipamentos do CtP (EPM, EAM). Normalmente, a colocação e retirada de peças de/para entidades de um CtP e do CtP para outros Ct's deve ser sincronizada. Esta sincronização pode ser feita pelo controlador de centro ou pelos controladores de equipamento diretamente.

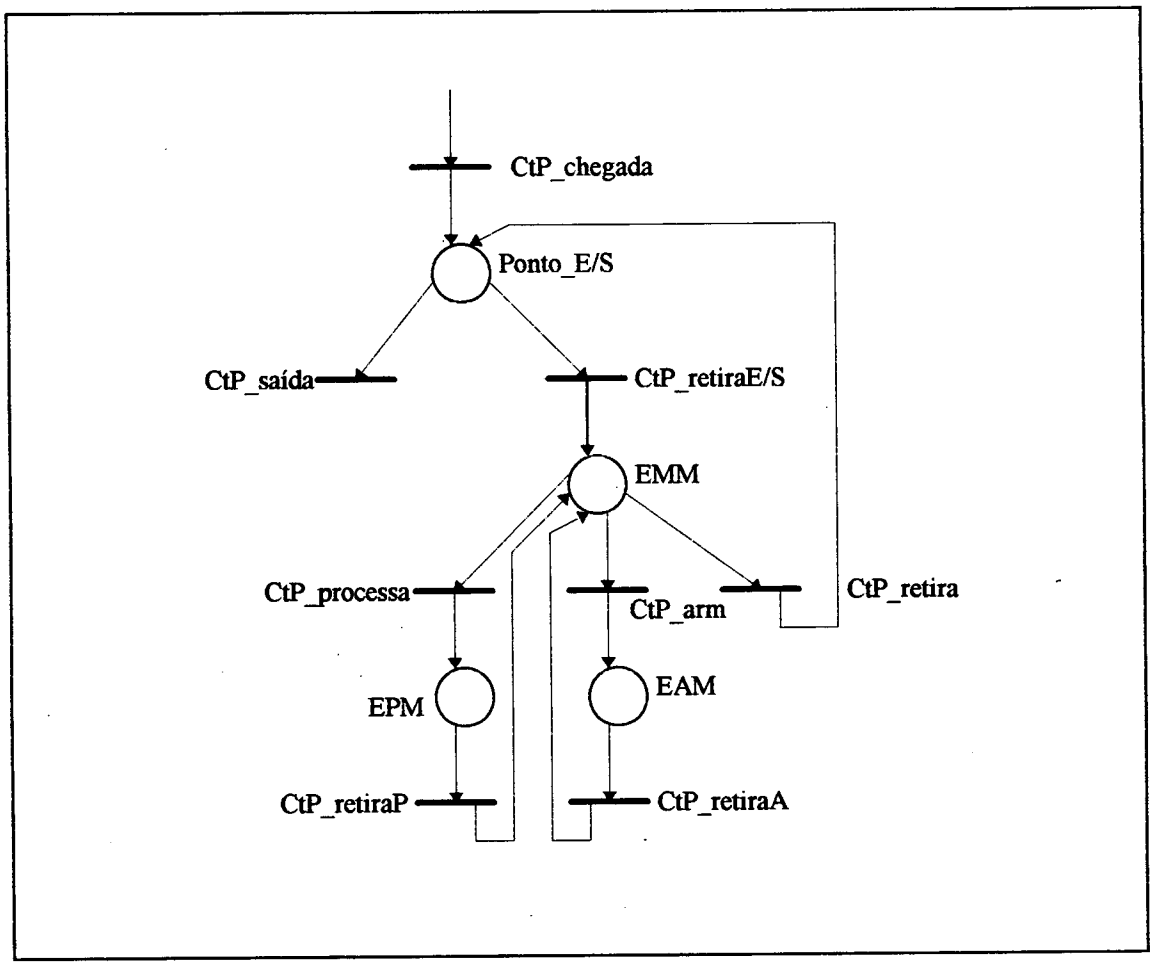


Figura 4.3 : Modelo de capacidades físicas de um CtP.

Quanto maior a independência dos controladores, no que diz respeito a conhecer o parceiro de sincronismo, mais genérico será o controlador. Neste caso, a política cliente/servidor aumenta as características de generalidade do controlador.

A Figura 4.4 mostra o modelo de capacidades físicas de um centro de trabalho de armazenamento. Neste caso, as peças chegam ao CtA através de um ponto de E/S de material, são retiradas com a participação de um EMM e posteriormente colocadas em um EAM. Para saírem do CtA, as peças são retiradas de um EAM e colocadas no ponto de E/S de material.

Um centro de trabalho de transporte (CtT) pode ter como componentes ETMs e EMMs. Neste caso, as peças entram em um ponto de E/S no CtT e podem ser removidas do ETM com a utilização de um EMM. A remoção das peças pode representar a transferência da mesma para outro ETM ou para outros centros de trabalho (CtP, CtA). A Figura 4.5 ilustra o modelo de capacidades físicas de um CtT.

4.1.2 Os Atributos funcionais

Os atributos funcionais fornecem uma descrição detalhada das entidades controladas pelo controlador. Os atributos incluem detalhes, sobre os equipamentos, que são necessários na implementação das tarefas genéricas especificadas nos modelos de capacidades físicas das entidades. Para exemplificar, será considerado um equipamento de processamento de material executando a tarefa de receber uma peça (Figura 4.2). Nesta tarefa uma peça é carregada em um EPM para ser processada. Se o equipamento tiver uma capacidade de armazenamento local, a peça em questão poderá ser carregada na posição de processamento ou em uma posição de armazenamento local.

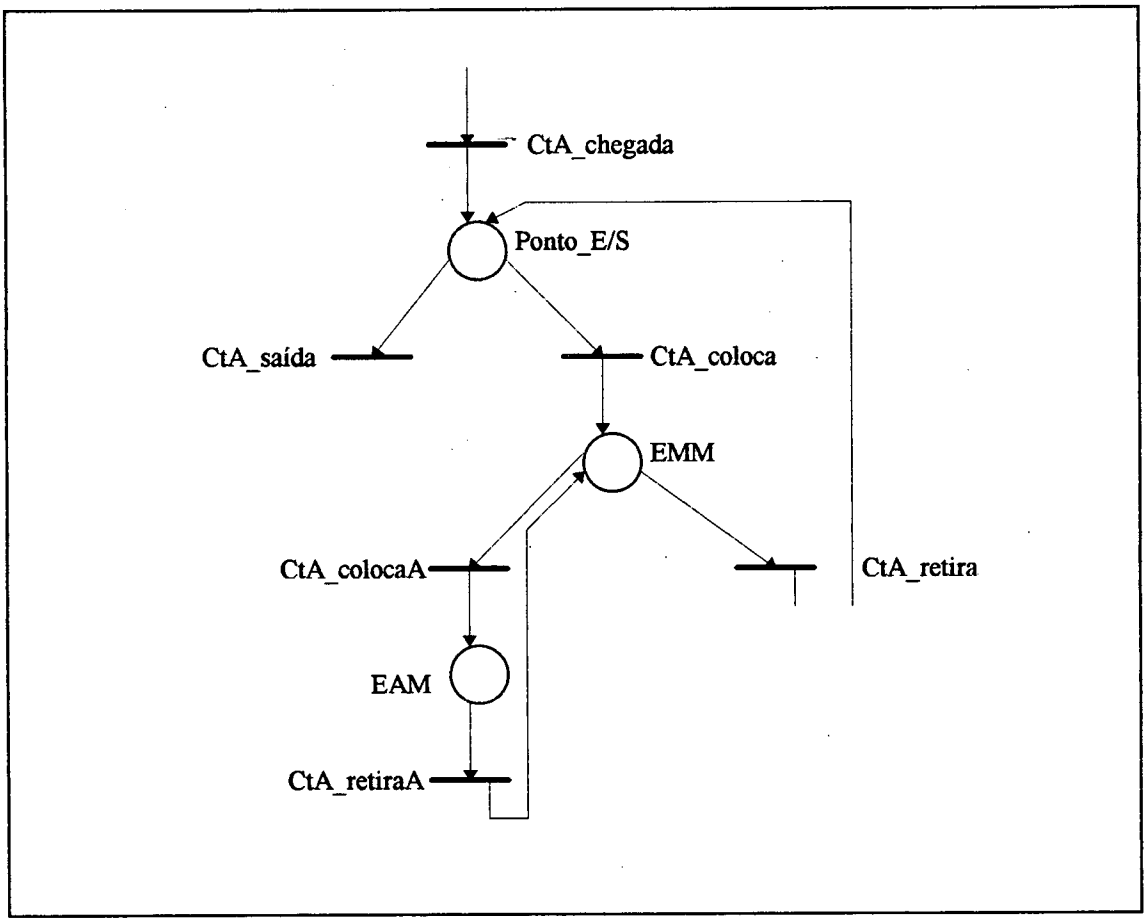


Figura 4.4 : Modelo de capacidades fisicas de um CtA.

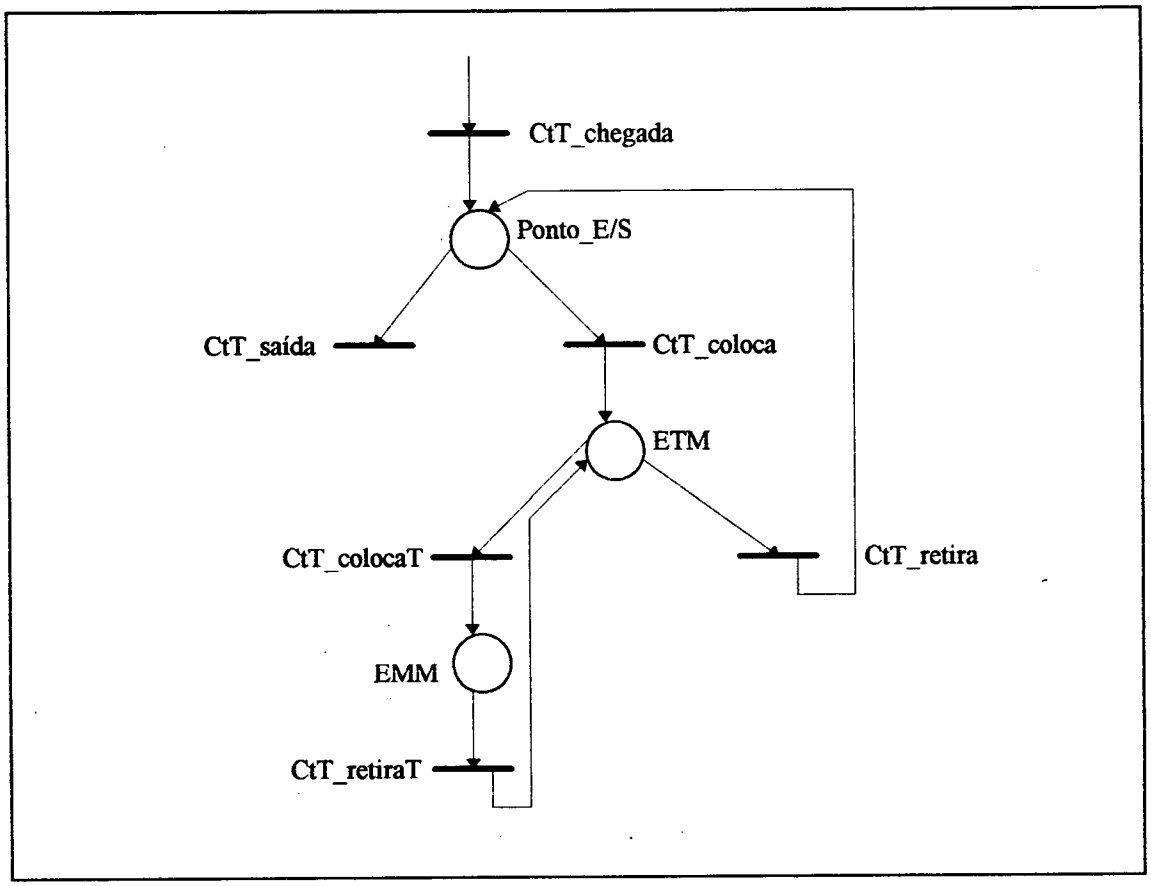


Figura 4.5 : Modelo de capacidades físicas de um CtT.

Entretanto, no modelo de capacidades físicas não existe esta distinção na tarefa de carregar uma peça para o equipamento. Neste caso, os atributos funcionais representam informações complementares como, por exemplo: os locais onde a peça pode ser colocada no equipamento e o tipo destes locais (processamento, entrada ou saída). A Figura 4.6 adaptada de [SMI92], mostra um equipamento de processamento de material e seus atributos funcionais. O EPM possui 3 locais onde a peça pode ser referenciada, um de entrada, um de saída e um de processamento. Estes três locais podem ser referenciados ou não, dependendo da situação (livre, bloqueado). Outras condições como o estado do equipamento também são descritas nos atributos funcionais.

Em relação aos centros de trabalho, os atributos fornecem detalhes dos equipamentos que compõem o Ct. Estas informações dizem respeito a localização dos equipamentos, a capacidade e tipo de equipamento. A Figura 4.7 mostra uma representação dos atributos de um centro de trabalho de processamento. Os atributos e o modelo de capacidades físicas estão disponíveis no modelo funcional descrito no Capítulo 3.

Finalmente, a interface entre os componentes planejamento/escalonamento e o componente execução detalha as tarefas que devem ser executadas pelos equipamentos ou centros de trabalho. Esta interface pode estar na forma de uma lista onde o escalonador coloca as suas requisições de serviço e o executor as retira para execução.

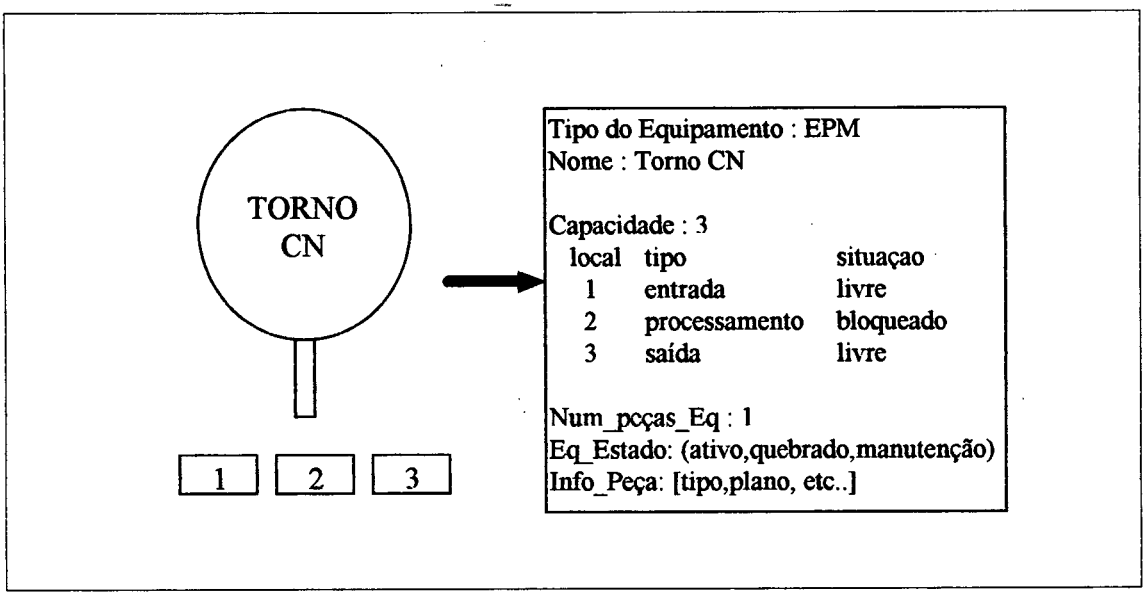


Figura 4.6 : Atributos funcionais de um EPM.

Tipo do Centro de Trabalho : CtP
Nome : Processamento 1
Capacidade : 3

Equipamentos : 6

Eq_Id	tipo	nome	Local	Estado	...
1	EPM	CN 1	(L1,L2,L3)		
2	EPM	CN 2	L4		
3	EPM	CN 3	L5		
4	EMM	Rb 1	L6		
5	EAM	BufE	L7		
6	EAM	BufS	L8		

Pontos de E/S : 2 (E : L7, S : L8)
Info_Peça: [tipo,plano, etc..]
...

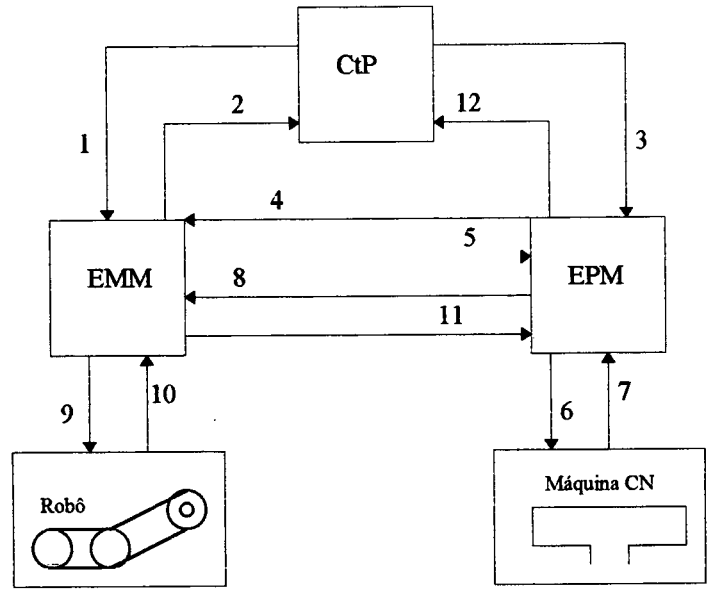
Figura 4.7 : Atributos de um CtP.

4.2 Modelos do comportamento dos Controladores

Enquanto o modelo de capacidades físicas descreve o comportamento geral do controlador, o seu comportamento detalhado é mostrado a partir da utilização de uma representação formal denominada rede de comportamento (Rc). As redes de comportamento foram introduzidas no Capítulo 3 e são utilizadas para representar o controle no ambiente distribuído proposto. Neste caso, o controle é representado pela troca de mensagens entre os controladores e pela execução das ações do controlador. Assim, uma sequência de mensagens e ações (eventos) nos controladores define a forma com que os mesmos interagem na realização de uma tarefa.

Por exemplo, a tarefa de *carga de uma peça* em um centro de trabalho, como o mostrado na Figura 1.2, envolve a interação de alguns componentes. O controlador de centro de trabalho (CtP) e os controladores de equipamento (EMM, EPM) desenvolvem um tipo de protocolo para realizar a tarefa. A Figura 4.8 mostra os protocolos estabelecidos entre as três entidades. O controlador CtP estabelece um protocolo formado pelas requisições 1,2,3,12. O controlador de equipamento EPM está envolvido na comunicação com o controlador CtP (3,12), o controlador EMM (4,5,8,11) e a máquina CN (7,6). Por último, o controlador EMM comunica-se com o controlador CtP (1,2), o controlador EMP (4,5,8,11) e o robô (9,10). Desta forma, uma tarefa genérica é traduzida numa sequência de mensagens e ações que envolve as entidades necessárias para a realização da mesma.

Na representação das redes de comportamento, cada lugar interno representa uma condição/estado (variável de estado) da entidade sendo descrita e cada atividade representa um evento que ocorre no sistema



Requisição	Serviço	Origem	Destino
1	Retira_peça	CtP	EMM
2	Peça_retirada	EMM	CtP
3	Carga_peça	CtP	EPM
4	Coloca_peça	EMP	EMM
5	Pegar_peça	EMM	EPM
6	Fechar_disp	EPM	Máquina CN
7	Disp_fechado	Máquina CN	EPM
8	Peça_presa	EPM	EMM
9	Abre_garra	EMM	Robô
10	Garra_aberta	Robô	EMM
11	Zona_livre	EMM	EPM
12	Carga_ok	EPM	CtP

Figura 4.8 : Representação de uma tarefa em um CtP.

(mensagem/sinal + ação). Os canais externos representam a fonte/destino de mensagens/sinais da entidade. Um canal é identificado pela classe de componentes que recebe/envia as mensagens/sinais (EPM, EMM, EAM, ETM, CtP, CtA, CtT, Maq). Quando o canal externo pode receber/enviar mensagens para mais de uma classe o canal é identificado apenas como de entrada ou saída (Ce,Cs). As entidades envolvidas na descrição dos comportamentos incluem : peças, equipamentos, centros de trabalho, pontos de E/S e programas CN.

4.2.1 Os Equipamentos

O modelo de capacidades físicas dos equipamentos de processamento de material define que as peças são carregadas no equipamento, processadas e depois retiradas do equipamento. Esta descrição é suficiente para uma visão geral das capacidades do equipamento. Entretanto, não contém detalhes suficientes para descrever completamente o comportamento do controlador do equipamento. A rede de comportamento incorpora detalhes específicos da operação do controlador. A Figura 4.9 mostra a rede de comportamento genérica para equipamentos da classe EPM.

Considerando que o EPM está na condição de recepção de peças, o controlador de centro de trabalho pode atribuir uma peça para o EPM (? Carga_p). O controlador EPM, por sua vez, determina um local de carga (a_localiza). Esta opção de carga pode existir em equipamentos que tenham mais de um local de carga. Ainda como parte da atividade **A1**, o EPM requisita o serviço responsável pela colocação da peça no local indicado (! Coloca_p). A tarefa de carga da peça em um EPM pode envolver uma ação síncrona ou assíncrona. No primeiro caso, existe a necessidade de os equipamentos envolvidos estarem sincronizados

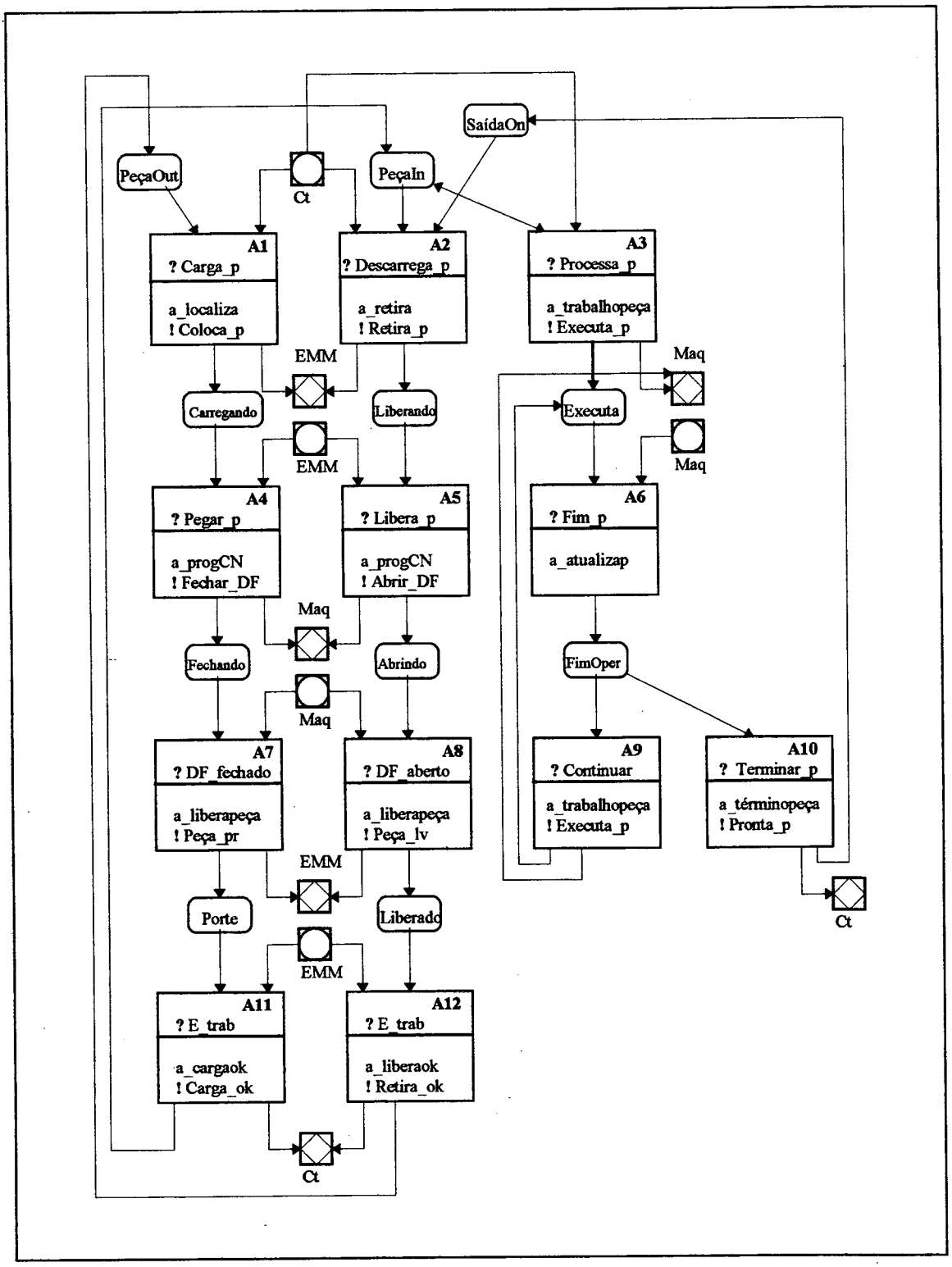


Figura 4.9 : Rede de comportamento genérica para um EPM.

para a realização da tarefa. No segundo caso, o EPM apenas indica o local de carga e espera que a tarefa seja completada. Nos modelos descritos estamos considerando o primeiro caso. A tarefa de carga é continuada quando o EPM recebe uma requisição para pegar a peça (? Pegar_p). Isto faz com que o EPM requisite, para a máquina, o fechamento do dispositivo de fixação da peça (! Fechar_DF) para que a mesma esteja segura (? DF_fechado). Após prender a peça, o EMP requisita a liberação do espaço de trabalho (! Peça_pr). O EPM notifica o centro de trabalho a realização da tarefa (! Carga_ok) e atualiza o estado do controlador. Neste ponto, a peça está sob o controle do controlador EPM. As atividades movimentação da peça para outra posição do equipamento (? Mover_p) e processamento da peça (? Processa_p) podem acontecer. As ações envolvidas na atividade de processamento (**A3**) incluem o envio de programa CN para a máquina e a requisição de execução da peça (! Executa_p). O término da execução é informado para o EPM (? Fim_p) quando a máquina termina a tarefa requisitada. Se, neste ponto, outras operações são necessárias o EPM pode continuar o processamento da peça (? Continuar). Esta decisão pode ser resultado da interação com os componentes planejamento/escalonamento. Quando o EPM termina o processamento (? Terminar_p), informa ao centro de trabalho que a peça está pronta (! Pronta_p). A retirada da peça do EPM pode ser realizada em função da existência de peças prontas para deixarem o EPM. Esta tarefa é semelhante à de carga da peça.

A representação da rede utilizando a notação da teoria dos conjuntos fica como segue:

$\mathbf{E}$ = { PeçaOut, PeçaIn, SaídaOn, Carregando, Liberando, Executando, Fechando, Abrindo, FimOper, Porte, Liberado }

$\mathbf{C} = \mathbf{C_e} \cup \mathbf{C_s}$ onde:

$\mathbf{C_e}$ = { Ct_e, EMM_e, Maq_e, Ce } onde:

Ct_e = { Carga_p, Descarrega_p, Processa_p},
EMM_e = { Pegar_p, Libera_p, E_trab },
Maq_e = { DF_fechado, DF_aberto, Fim_p }
Ce = { Continuar, Terminar_p } }

$\mathbf{C_s}$ = { Ct_s, EMM_s, Maq_s } onde:

Ct_s = { Carga_ok, Retira_ok, Pronta_p},
EMM_s = { Coloca_p, Retira_p, Peça_lv, Peça_pr },
Maq_s = { Fechar_DF, Abrir_DF, Executa_p } }

A = { **A1**, **A2**, **A3**, ..., **A12** }, onde :

A1 = { (PeçaOut, Ct?Carga_p), (a_localiza, EMM! Coloca_p, Carregando) }

A2 = { (PeçaIn,SaídaOn, Ct?Descarrega), (a_retira,EMM! Retira_p,Liberando) }

A3 = { (PeçaIn, Ct?Processa_p), (a_trabalhopeça, Maq!Executa_p,Executando) }

A4 = { (Carregando, EMM?Pegar_p), (a_progCN, Maq!Fechar_DF, Fechando) }

A5 = { (Liberando, EMM?Libera_p),(a_progCN, Maq! Abrir_DF, Abrindo) }

A6 = { (Executando, Maq?Fim_p), (a_atualizap,FimOper) }

A7 = { (Fechando, Maq?DF_fechado), (a_prendepeça, EMM! Peça_pr, Porte) }

A8 = { (Abrindo, Maq?DF_aberto), (a_liberapeça, EMM! Peça_lv, Liberado)}

A9 = { (FimOper,Ce?Continuar), (a_trabalhopeça,Maq!Executa_p,Executando)}

A10 = { (FimOper, Ce?Terminar_p), (a_términopeça, Ct! Pronta_p, SaídaOn) }

A11 = { (Porte, EMM?E_trab), (a_términopeça, Ct! Pronta_p, SaídaOn) }

A12 = { (Liberado, EMM?E_trab), (a_términopeça, Ct! Pronta_p, SaídaOn) }

Algumas condições são também estabelecidas com o objetivo de impor certas restrições, sejam elas físicas ou lógicas. Por exemplo, *PeçaOut* é uma condição que indica que existe um lugar de entrada disponível no EPM; a *carga* de peças na máquina só pode ser realizada se esta condição estiver satisfeita. *PeçaIn* é uma condição que restringe a execução das tarefas de processamento e liberação de peças

a situações onde existe uma peça no local de trabalho da máquina. No caso da liberação de peças esta condição é combinada com *SaídaOn* permitindo assim que uma peça seja retirada de um EPM.

Formalmente estas condições são assim definidas :

PeçaOut : $\exists$ x | local.entrada(x) $\wedge$ local.livre(x)
PeçaIn : $\exists$ x | local.processamento(x) $\wedge$ local.livre(x)

Uma outra forma de representação do comportamento dos componentes é a utilização da descrição textual, ou seja, onde os componentes são representados por programas que são movidos por eventos. Esta forma de representação foi introduzida no Capítulo 3. O comportamento de um componente da classe EPM ficaria assim representado:

Componente Controlador_EPM

[*entrada* : [Carga_p,
Descarrega_p,
Processa_p,
Pegar_p,
Libera_p,
E_trab,
DF_fechado,
DF_aberto,
Fim_p,
Continuar,
Terminar_p],

saída : [Carga_ok,
Retira_ok,
Pronta_p,
Coloca_p,
Retira_p,
Peça_lv,
Peça_pr
Fechar_DF,

Abrir_DF,
Executa_p],

interno : [PeçaOut,
PeçaIn,
SaídaOn,
Carregando,
Liberando,
Executando,
Fechando,
Abrindo,
FimOper,
Porte,
Liberado]

var : [Local]

inicialização : [Local.In := 1]

Carga_p & PeçaOut ==> [a_localiza, *externo*(Coloca_p), *interno*(Carregando)]

Descarrega&PeçaIn,SaídaOn==>[a_retira,*externo*(Retira_p),*interno*(Liberando)]

Processa_p&PeçaIn==>[a_trabalhopeça,*externo*(Executa_p),*interno*(Executando)]

Pegar_p&Carregando==>[a_progCN,*externo*(Fechar_DF),*interno*(Fechando)]

Libera_p&Liberando==>[a_progCN,*externo* (Abrir_DF), *interno*(Abrindo)]

Fim_p &Executando==>[a_atualizap,*interno*(Fim_oper)]

DF_fechado&Fechando==>[a_prendepeça,*externo*(Peça_pr), *interno*(Porte)]

DF_aberto &Abrindo==>[a_liberapeça,*externo*(Peça_lv),*interno*(Liberado)]

Continuar&Fim_oper==>[a_trabalhopeça,*externo*(Executa_p),*interno*(Executando)]

Terminar_p&Fim_oper==>[a_términopeça, *externo*(Pronta_p),*interno*(SaídaOn)]

E_trab &Porte==>[a_términopeça, *externo*(Pronta_p),*interno*(SaídaOn)]

E_trab &Liberado==>[a_términopeça, *externo*(Pronta_p),*interno*(SaídaOn)]]

Conforme já referido, os equipamentos usados para a movimentação de peças dentro de um centro de trabalho e para carga/descarga de peças em equipamentos do centro são chamados EMM. O modelo de capacidades físicas destes elementos prevê que as peças são retiradas de uma localização e colocadas em outra.

Na descrição do comportamento de um EMM, Figura 4.10, o processo de movimentação é iniciado com uma requisição de retirada de uma peça (? Retira_p). O controlador EMM executa a atividade **A1** a partir da aproximação do equipamento para o local de carga da peça e da requisição do fechamento do dispositivo de fixação do equipamento (! Fechar_GR). O fechamento do dispositivo de fixação, quando concluído, é informado para o controlador EMM (? GR_fechado). Estando com a peça, o controlador EMM deve requisitar a liberação da mesma por parte do controlador que está de posse da peça (! Libera_p). Neste caso, a liberação da peça significa que o controlador EMM pode então requisitar o afastamento do equipamento da área de carga da peça. Normalmente o equipamento move-se para uma localização de repouso onde não interfere com a área de trabalho de outros equipamentos. Após o afastamento (? Retrair_ok), o controlador EMM está numa condição de carregado, ou seja, segurando uma peça. Esta condição permite o EMM receber requisições de retirada da peça (? Coloca_p). Na retirada de peças o controlador EMM inicialmente aproxima do local de entrega, requisita que a peça seja presa, libera a peça a partir da abertura do dispositivo de fixação da peça e, finalmente, move-se para um local de repouso.

A Rc da Figura 4.10 fica assim representada :

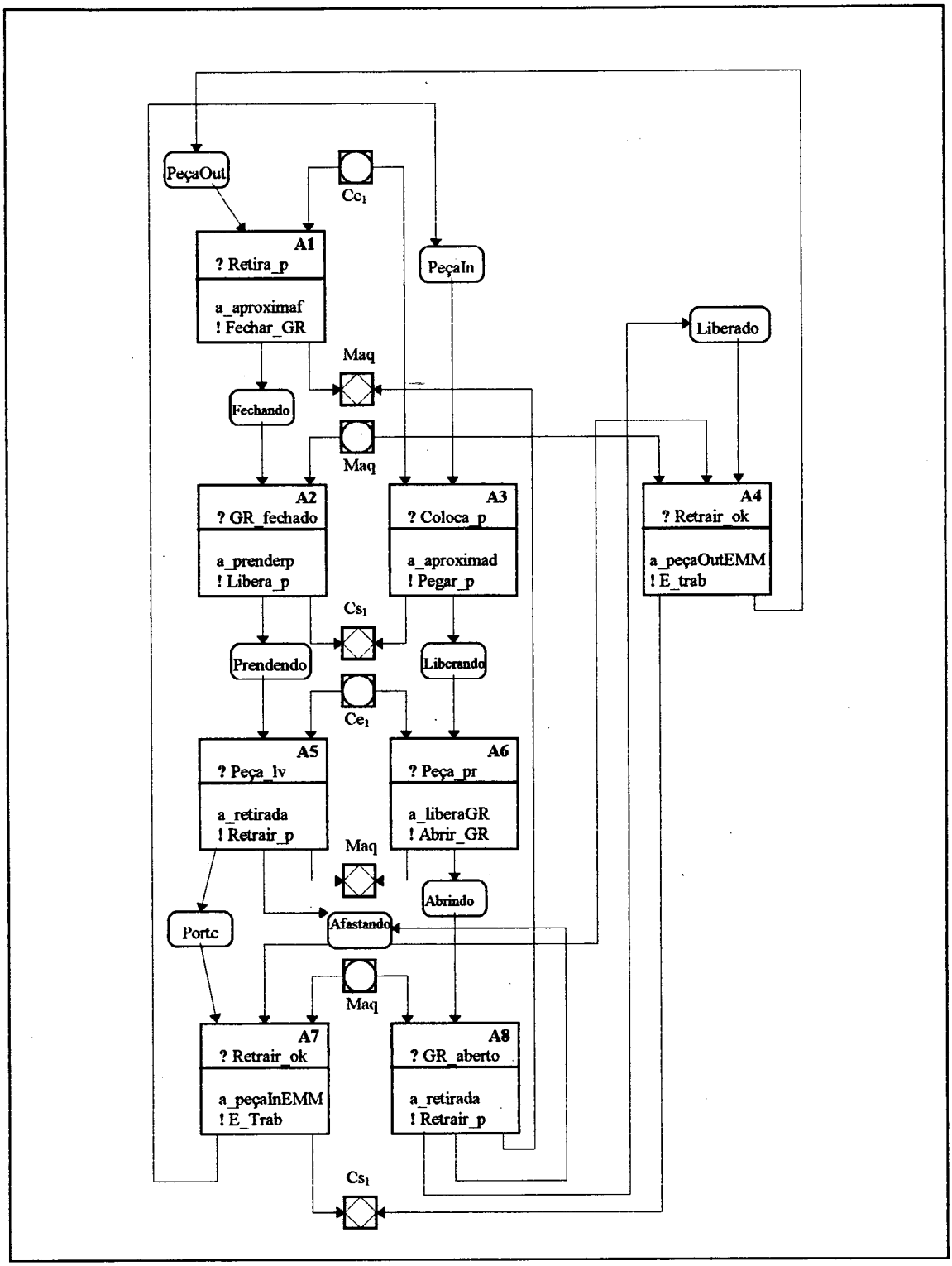


Figura 4.10 : Rede de comportamento genérica para um EMM.

E = { PeçaOut, PeçaIn, Fechando, Prendendo, Liberando, Porte,
Abrindo, Afastando, Liberado }

C = $\mathbf{C_e} \cup \mathbf{C_s}$ onde:

$\mathbf{C_e}$ = { Ce, Maq_e }, onde:
Ce = { Retira_p, Coloca_p , Peça_lv, Peça_pr}
Maq_e = { GR_fechado, GR_aberto, Retrair_ok }

$\mathbf{C_s}$ = { Cs, Maq_s }, onde:
Cs = { Libera_p, Pegar_p , E_trab}
Maq_s = { Fechar_GR, Abrir_GR, Retrair_p }

A = { **A1**, **A2**, **A3**, ..., **A8** }, onde :

A1 = { (PeçaOut, Ce?Retira_p), (a_aproximaf, Maq! Fechar_GR, Fechando)}

A2 = { (Fechando, Maq?GR_fechado), (a_prenderp, Cs! Libera_p, Prendendo) }

A3 = { (PeçaIn, Ce?Coloca_p), (a_aproximad, Cs! Pegar_p, Liberando) }

A4={(Liberado,Afastando,Maq?Retrair_ok),
(a_peçaOutEMM,Cs!E_trab,PeçaOut) }

A5 = { (Prendendo, Ce?Peça_lv),(a_retirada, Maq! Retrair_p, Porte,Afastando) }

A6 = { (Liberando, Ce?Peça_pr), (a_liberaGR, Maq! Abrir_GR,Abrindo) }

A7 = {(Porte,Afastando,Maq?Retrair_ok), (a_peçaInEMM, Cs! E_trab, PeçaIn) }

A8={(Abrindo,Maq?GR_aberto),(a_retirada,Maq!Retrair_p,Liberado,Afastando)}

PeçaOut é uma condição que indica que o EMM não tem nenhuma peça em seu poder, ou seja, está disponível para a *carga* de peças. *PeçaIn* é uma condição que restringe a execução da tarefa de carga e habilita o EMM a executar uma tarefa de liberação de peças.

Nos equipamentos de armazenagem de material, segundo o modelo de capacidades físicas, as peças são colocadas nestes equipamentos e depois retiradas.

O processo considera que as peças são retiradas/colocadas de/para um ponto de Entrada/Saída. Desta forma, o controlador de um EAM recebe uma requisição para colocação de peça(s) em um EAM (? Arm_In), determina uma localização para a peça e requisita a colocação da mesma no equipamento (! Coloca_p). A Figura 4.11 mostra a Rc de um EAM.

Na recepção da requisição para prender a peça (? Pegar_p) o controlador EAM insere a peça no equipamento e pede a liberação de seu espaço de trabalho. A execução da requisição de colocação de peça no EMM é completada e informada para a entidade cliente (! ArmIn_ok). A tarefa de retirada de peça(s) é semelhante. A representação da rede fica como segue:

E = { ArmOn, PeçaIn, Pegando, Liberando, Porte, PeçaOut }

C = $\mathbf{C_e} \cup \mathbf{C_s}$ onde:

$\mathbf{C_e}$ = { Ce, EMM_e }, onde:
Ce = { Arm_In, Arm_Out }
EMM_e = { Pegar_p, Libera_p, E_trab }

$\mathbf{C_s}$ = { Cs, EMM_s }, onde:
Cs = { ArmIn_ok, ArmOut_ok, }
EMM_s = { Coloca_p, Retira_p, Peça_lv, Peça_pr }

A = { **A1**, **A2**, ..., **A6** }, onde :

A1 = { (ArmOn, Ce?Arm_In), (a_alocalugar, EMM! Coloca_p, Pegando) }

A2 ={(PeçaIn,ArmOn,Ce?Arm_Out),(a_liberalugar,EMM!Retira_p,Liberando) }

A3 = { (Pegando, EMM?Pegar_p), (a_inserepeça, EMM! Peça_pr, Porte) }

A4 = { (Liberando, EMM?Liberar_p), (a_liberapeça, EMM! Peça_lv, PeçaOut) }

A5 = { (Porte, EMM?E_trab),(a_cargaok, Cs! ArmIn_ok, ArmOn,PeçaIn) }

A6 = { (PeçaOut, EMM?E_trab), (a_retiraok, Cs! ArmOut_ok,ArmOn) }

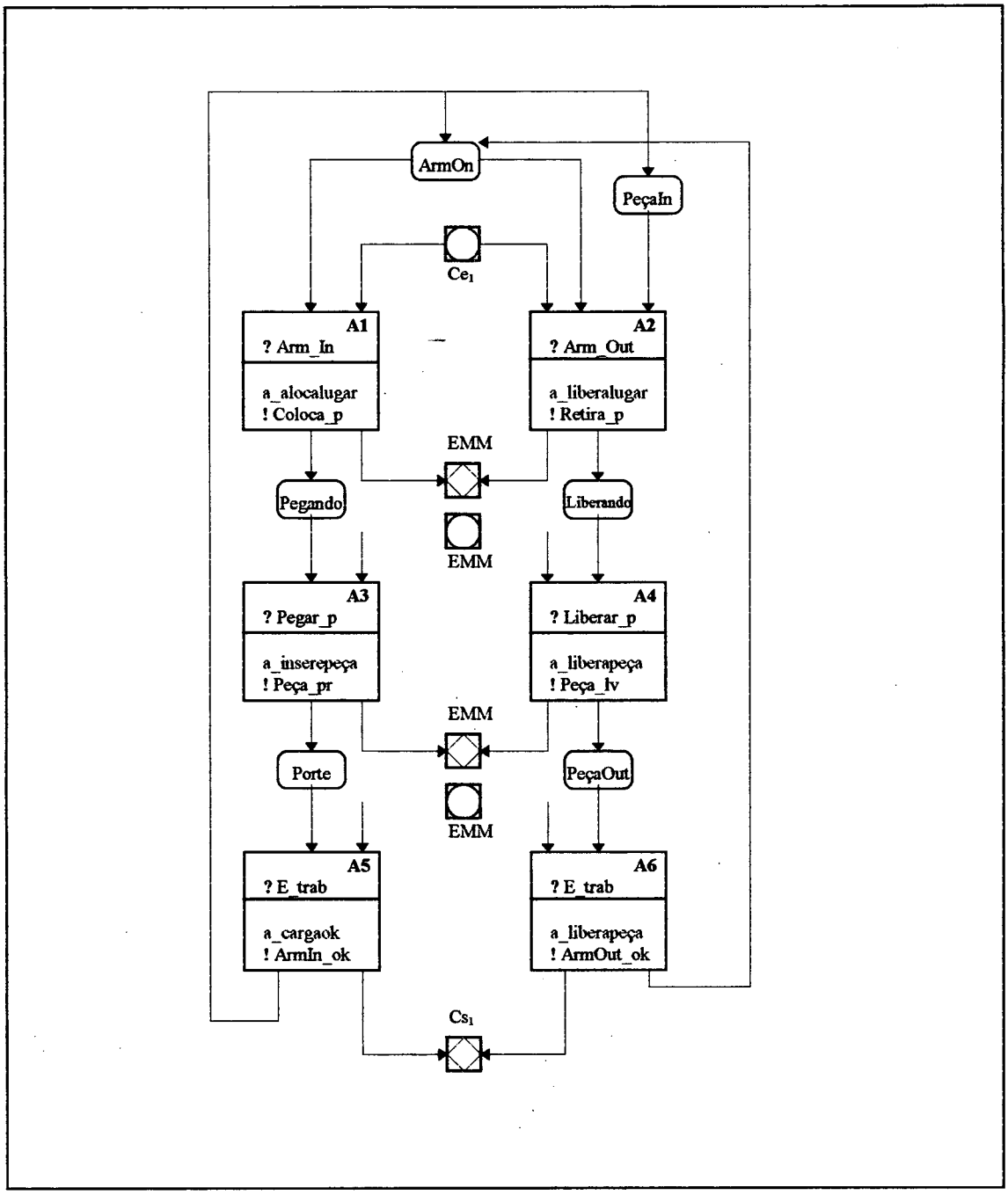


Figura 4.11 : Rede de comportamento genérica para um EAM.

A classe de equipamentos de transporte tem como responsabilidades o transporte de peças entre os centros de trabalho. Segundo o modelo de capacidades físicas, os equipamentos ETM são similares aos EPM, ou seja, as peças são colocadas nos equipamentos, transportadas para um determinado ponto e depois retiradas do equipamento.

A rede de comportamento da Figura 4.12 descreve o processo de transporte. Inicialmente, o controlador ETM recebe uma requisição de transporte (? Transp_In). Neste caso, o ETM deve mover-se até o local de captura da peça e requisitar a liberação da peça por parte do responsável pelo ponto de E/S. A movimentação pode ser feita em etapas (Mover) até que o veículo esteja no local desejado. Chegando ao local, o controlador requisita a liberação da peça (? Libera_p). Quando o controlador ETM é requisitado para prender a peça ele então o faz e sinaliza o atendimento da mesma (?Pegar_p, !Peça_pr). Finalmente o ETM está livre para mover-se e realizar a entrega da peça carregada. Na tarefa de entrega o controlador ETM movimenta-se até o ponto de entrega e negocia com o responsável pelo ponto a retirada da peça do veículo transportador. Note que o veículo transportador não é necessáriamente automático. A representação da rede fica como segue:

E = { PeçaIn, PeçaOut , Movendo, Recebendo, Entregando, Porte, Liberado}

$\mathbf{C} = \mathbf{C_e} \cup \mathbf{C_s}$ onde:

$\mathbf{C_e}$ = { Ce, Maq }, onde:
Ce = { Transp_In, Transp_Out, Pegar_p, Liberar_p, E_trab }
Maq = { Local_On }

$\mathbf{C_s}$ = { Cs }, onde:
Cs = { Libera_p, Pegar_p, Peça_pr, Peça_lv }

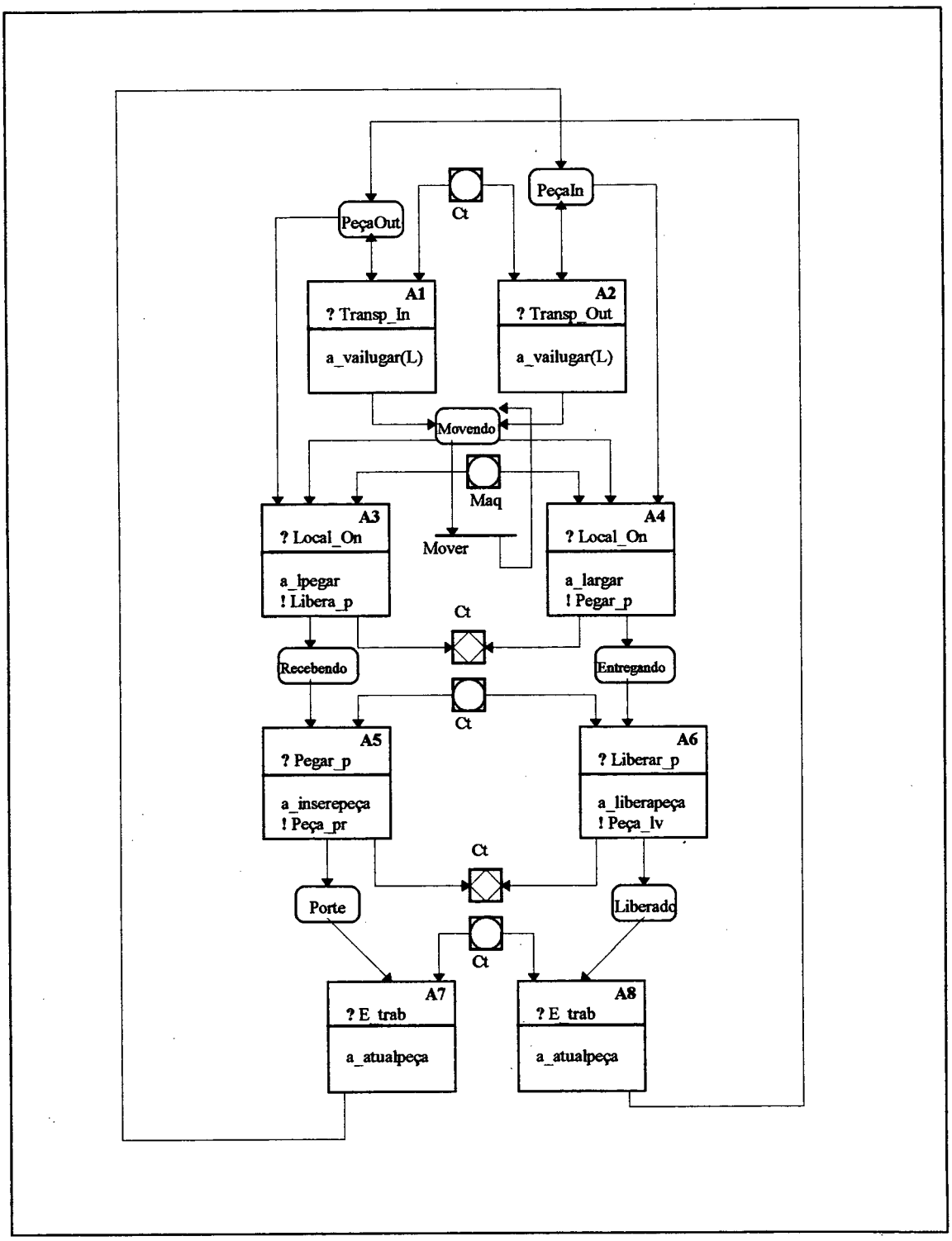


Figura 4.12 : Rede de comportamento genérica para um ETM.

A = { A1, A2, ..., A8 }, onde :

A1 = { (PeçaOut, Ce?Transp_In), (a_vailugar, Movendo, PeçaOut) }

A2 = { (PeçaIn, Ce?Transp_Out), (a_vailugar, Movendo, PeçaIn) }

A3 = {(Movendo,PeçaOut,Maq?Local_On),(a_pegar, Cs!Libera_p, Recebendo)}

A4 = { (Movendo,PeçaIn, Maq?Local_On), (a_largar,Cs! Pegar_p,Entregando) }

A5 = { (Recebendo, Ce?Pegar_p),(a_inserepeça, Cs! Peça_pr,Porte) }

A6 = { (Entregando, Ce?Liberar_p), (a_liberapeça, Cs! Peça_lv,Liberado) }

A7 = { (Porte, Ce?E_trab), (a_atualpeça,PeçaIn) }

A8 = { (Liberado, Ce?E_trab), (a_atualpeça,PeçaOut) }

PeçaOut é uma condição que indica que o ETM não tem nenhuma peça em seu poder, ou seja, está disponível para a *carga* de peças. *PeçaIn* é uma condição que restringe a execução da tarefa de carga e habilita o ETM a executar uma tarefa de liberação de peças. Durante a movimentação (Movendo) estas condições não são desabilitadas, ou seja, um ETM move-se para a carga sem peças e move-se para o destino com peças.

4.2.2 Centros de Trabalho

Visto que os centros de trabalho não tratam diretamente com equipamentos físicos, as funções (tarefas) executadas pelos controladores são basicamente de coordenação e sincronização dos controladores de equipamento. Os controladores de centro recebem requisições de outros centros e requisitam serviços de seus componentes controladores de equipamento. Por outro lado, eventos como a chegada de uma nova peça no centro, uma peça terminada ou a mudança do estado de uma peça podem também gerar ações por parte do controlador de Ct. Por exemplo, a

saída de uma peça de um centro faz com que o mesmo requisite serviços de outros controladores.

O modelo de capacidades físicas de um centro de trabalho de processamento estabelece que o CtP pode receber peças no seu ponto de E/S, processar as peças e retira-las do centro. Considerando a arquitetura adotada para o nível de chão-de-fábrica, algumas requisições são originadas na Supervisão Global. A Figura 4.13 mostra a Rc para um controlador de centro de trabalho de processamento.

As requisições de serviço de entrada e saída de peças são recebidas a partir de comunicação com a Supervisão Geral (?Chegada, ?Saída). A retirada de uma peça do ponto de E/S se realiza realmente quando requisitada pelo controlador responsável pela entrega da mesma (?Retira_pE/S). Esta tarefa é realizada em conjunto com o EMM que integra o centro, a partir do protocolo estabelecido com o controlador do EMM e com o controlador responsável pelo equipamento de entrega. Desta forma o controlador CtP negocia a inserção da peça no centro. Depois de estar com a peça o CtP pode realizar as operações de processamento necessárias para a peça (?Processa_p), manter a peça armazenada no centro (?Armazena_p) quando o centro tem algum equipamento de armazenamento e retirar a peça do centro (?CtOut_p). Estas requisições são atendidas pelo controlador CtP e envolvem a participação dos integrantes do centro. Uma requisição de processamento faz com que o controlador CtP requisite os serviços de um EPM através de seu controlador (!Carga_p). O processamento de uma peça termina quando a peça esta pronta (?Pronta_p) ou por um pedido de parada forçada (?Pára_proc). A requisição de parada forçada é resultado de algum problema detectado pelo controlador de equipamento ou parte da interação com os componentes de planejamento/escalonamento. O armazenamento é requisitado em

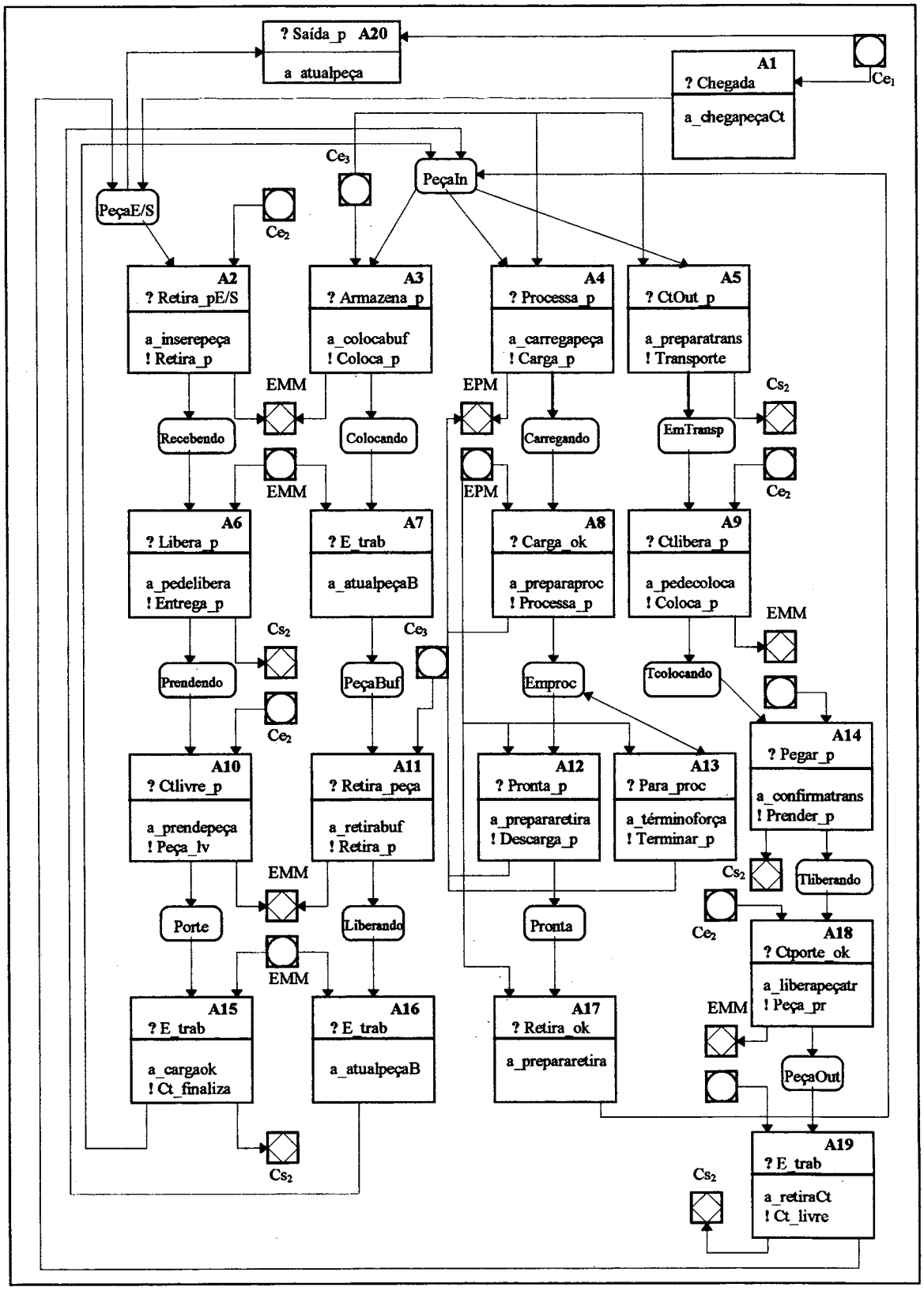


Figura 4.13 : Rede de comportamento genérica para um CtP.

função de alguma necessidade de armazenar uma peça temporariamente à espera de processamento. Isto acontece quando o CtP tem EAM's associados a ele. A realização desta requisição envolve o controlador do equipamento responsável pela movimentação de peça e o equipamento de armazenamento (aqui está sendo considerado um EAM do tipo *buffer* que não tem comportamento, é uma entidade passiva). Neste caso, o controlador CtP requisita a colocação de peças no *buffer* (!Coloca_p). Finalmente, a retirada de peças é realizada a partir da interação com o controlador responsável pelo transporte da peça (!Transporte) e o controlador responsável pela movimentação da peça (!Coloca_p). O resultado da retirada é a peça no ponto de E/S. A rede fica assim representada:

E = { PeçaE/S, PeçaIn, Recebendo, Colocando, Carregando, EmTransp, Prendendo, PeçaBuf, Emproc, Tcolocando, Porte, Liberando, Pronta, Tliberando, PeçaOut, }

$\mathbf{C} = \mathbf{C_e} \cup \mathbf{C_s}$ onde:

$\mathbf{C_e}$ = { Ce_1, Ce_2, Ce_3, EMM_e, EMP_e }, onde:
Ce_1 = { Chegada, Saída_p }
Ce_2 = { Retira_pE/S, Ctlivre_p, Ctlibera_p, Ctporte_ok }
Ce_3 = { Armazena_p, Processa_p, CtOut_p, Retira_pç }
EMM_e = { Libera_p, Pegar_p, E_trab }
EMP_c = { Carga_ok, Pronta_p, Pára_proc, Retira_ok }

$\mathbf{C_s}$ = { Cs_2, EMM_s, EPM_s }, onde:
Cs_2 = { Entrega_p, Ct_finaliza, Transporte, Prender_p, Ct_livre }
EMM_s = { Coloca_p, Retira_p, Peça_lv, Peça_pr }
EPM_s = { Carga_p, Processa_p, Descarga_p, Termina_p }

A = { **A1**, **A2**, **A3**, ..., **A20** }, onde :

A1 = { (Ce_1?Chegada), (a_chegapeçaCt, PeçaE/S) }

A2 = {(PeçaE/S,Ce_2?Retira_pE/S), (a_inserepeça, EMM!Retira_p, Recebendo)}

A3 = { (PeçaIn, Ce_3?Armazena_p), (a_colocabuf, EMM!Coloca_p,Colocando) }

A4 = { (PeçaIn, Ce_3?Processa_p), (a_carregapeça, EPM!Carga_p,Carregando) }

A5 = { (PeçaIn, Ce_3?CtOut_p),(a_preparatrans, Cs_2!Transporte, EmTransp) }

A6 = {(Recebendo, EMM?Libera_p), (a_pedelibera, Cs_2!Entrega_p,Prendendo)}

A7 = { (Colocando, EMM?E_trab), (a_atualpeça, PeçaBuf) }

A8 ={(Carregando, EPM?Carga_ok),(a_preparaproc, EPM!Processa_p,Emproc)}

A9 ={(EmTransp, Ce_2?Ctlibera_p), (a_pedecoloca, EMM!Coloca_p,Tcolocando)}

A10 = { (Prendendo, Ce_2?Ctlivre_p), (a_prendepeça, EMM!Peça_lv,Porte) }

A11 = { (PeçaBuf, Ce_3?Retira_pç), (a_retirabuf, EMM!Retira_p,Liberando) }

A12 = {(Emproc, EPM?Pronta_p),(a_prepararetira, EPM!Descarga_p,Pronta)}

A13 ={(Emproc, EPM?Para_proc), (a_terminaforça, EPM!Termina_p,Emproc)}

A14={(Tcolocando,EMM?Pegar_p),(a_confirmatrans,Cs_2!Prender_p,Tliberando)}

A15 = { (Porte, EMM?E_trab), (a_cargaok, Cs_2!Ct_finaliza,PeçaIn) }

A16 = {(Liberando, EMM?E_trab),(a_atualpeçaB,PeçaIn)}

A17 ={(Pronta, EPM?Retira_ok), (a_prepararetira,PeçaIn)}

A18 = {(Tliberando, Ce_2?Ctporte_ok), (a_liberapeçatr, EMM!Peça_pr,PeçaOut)}

A19 = {(PeçaOut, EMM?E_trab),(a_retiraCt, Cs_2!Ct_livre,PeçaE/S)}

A20 ={(PeçaE/S, Ce_1?Saída_p), (a_atualpeça)}

PeçaE/S é uma condição que indica que o CtP tem peça disponível em seu ponto de E/S, ou seja, pode efetuar uma atividade de *carga* de peças ou uma peça está pronta para deixar o centro. *PeçaIn* é uma condição que se torna verdadeira no momento em que um EMM do centro está de posse de uma peça. Esta condição habilita as atividades que dependem dos serviços do EMM. A condição *PeçaBuf* é verdadeira quando existem peças no *buffer* do CtP. *EmProc* implica que deve existir uma peça sendo trabalhada em um EPM pertencente ao centro.

O centro de trabalho de armazenamento tem como atribuições o armazenamento temporário de peças ou matéria prima que estão sendo trabalhadas

nos centros de processamento. Desta forma, segundo o modelo de capacidades físicas, o CtA recebe requisições de armazenagem e para retirada de peças.

O comportamento do controlador de CtA é semelhante ao CtP nas tarefas de colocação de peças no centro e de retirada das mesmas. O controlador CtA também interage com os controladores responsáveis pelo transporte (?Retira_pE/S, !Transporte) da peça, e pela movimentação da mesma (!Retira_p, !Coloca_p). De posse da peça o controlador CtA é requisitado para a armazenagem em entidades passivas (*buffer*) ou ativas (AS/RS). A colocação e retirada de peça em uma entidadade passiva é realizada a partir da interação com o controlador de equipamento responsável pela movimentação das peças (EMM). No caso dos equipamentos automáticos, ativos, o controlador CtA requisita serviços para o controlador do EAM para a realização da tarefa de armazenagem/retirada. Estes serviços são requisitados quando existe necessidade de uma peça ser armazenada temporariamente no sistema ou quando uma nova peça é requisitada por um centro de processamento. A Figura 4.14 mostra a rede de comportamento de um controlador de CtA. A rede fica assim representada:

E = { PeçaE/S, PeçaIn, Retirando, Armazenando, EsperaT, Recebendo,
PeçaBuf, PeçaAS, Colocando, Porte, Liberando, Entregando,
PeçaOut }

C = $\mathbf{C_e} \cup \mathbf{C_s}$ onde:

$\mathbf{C_e}$ = { Ce_1, Ce_2, Ce_3, EMM_e, EAM_e }, onde:
Ce_1 = { Retira_pE/S, Ctlivre_p, Ctlibera_p, Ctporte_ok }
Ce_2 = { Chegada, Saída_p }
Ce_3 = { Arm_Buf, Arm_AS, CtOut_p, Retira_Buf, Retira_RS }
EMM_e = { Libera_p, Pegar_p, E_trab }
EAM_e = { ArmIn_ok, ArmOut_ok }

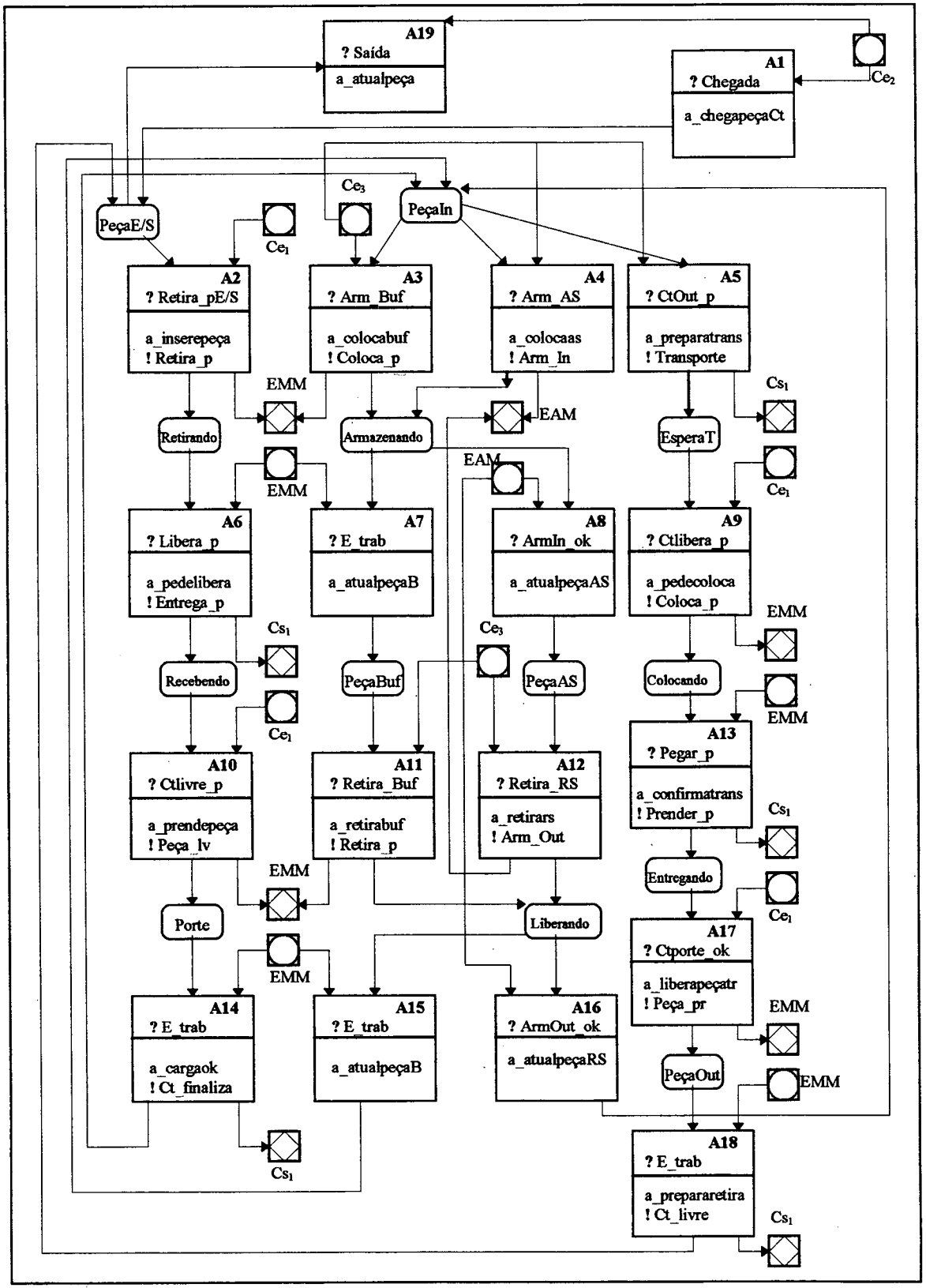


Figura 4.14 : Rede de comportamento genérica para um CtA.

$\mathbf{C_s}$ = { Cs_1, EMM_s, EAM_s }, onde:
Cs_1 = { Entrega_p, Ct_finaliza, Transporte, Prender_p, Ct_livre }
EMM_s = { Coloca_p, Retira_p, Peça_lv, Peça_pr }
EAM_s = { Arm_In, Arm_Out }

A = { **A1**, **A2**, **A3**, ..., **A19** }, onde :

A1 = { (Ce_2?Chegada), (a_chegapeçaCt, PeçaE/S) }

A2 = {(PeçaE/S,Ce_1?Retira_pE/S), (a_inserepeça, EMM!Retira_p, Retirando)}

A3 = { (PeçaIn, Ce_3?Arm_Buf), (a_colocabuf, EMM!Coloca_p,Armazenando) }

A4 = { (PeçaIn, Ce_3?Arm_AS), (a_colocaas, EAM!Arm_In,Armazenando) }

A5 = { (PeçaIn, Ce_3?CtOut_p),(a_preparatrans, Cs_1!Transporte, EsperaT) }

A6 = {(Retirando, EMM?Libera_p), (a_pedelibera, Cs_1!Entrega_p,Recebendo)}

A7 = { (Armazenando, EMM?E_trab), (a_atualpeçaB, PeçaBuf) }

A8={(Armazenando,EAM?ArmIn_ok),(a_atualpeçaAS, PeçaAS)}

A9 ={(EsperaT, Ce_1?Ctlibera_p), (a_pedecoloca, EMM!Coloca_p,Colocando)}

A10 = { (Recebendo, Ce_1?Ctlivre_p), (a_prendepeça, EMM!Peça_lv,Porte) }

A11 = { (PeçaBuf, Ce_3?Retira_Buf), (a_retirabuf, EMM!Retira_p,Liberando) }

A12 = {(PeçaAS, Ce_3?Retira_RS),(a_retirars, EAM!Arm_out, Liberando)}

A13={(Colocando,EMM?Pegar_p),(a_confirmatrans,Cs_1!Prender_p,Entregando)}

A14={(Porte,EMM?E_trab),(a_cargaok,Cs_1!Ct_finaliza,PeçaIn)}

A15 = { (Liberando, EMM?E_trab), (a_atualpeça, PeçaIn) }

A16 = {(Liberando, EAM?ArmOut_ok),(a_atualpeçaRS, PeçaIn)}

A17 ={(Entregando, Ce_1?Ctporte_ok), (a_liberapeçatr,EMM?Peça_pr,PeçaOut)}

A18 = {(PeçaOut, EMM?E_trab), (a_prepararetira, Cs_1!Ct_livre,PeçaE/S)}

A19 ={(PeçaE/S, Ce_2?Saída_p), (a_atualpeça)}

As pré-condições no caso do CtA são semelhantes ao CtP. Quando existe uma peça em algum EAM ou *buffer* as condições *PeçaBuf* ou *PeçaAS* são ligadas.

O centro de trabalho de transporte (CtT) é responsável pelo transporte de peças entre outros centros de trabalho como de processamento e de armazenamento. O processo se desenvolve a partir das requisições de transporte e entrega de peças (?Transporte, ?Entrega_T) que são feitas por outros centros. As requisições são atendidas com a participação dos controladores responsáveis pelos equipamentos de transporte e a partir da interação com os outros centros. A Rc de um CtT é mostrada na Figura 4.15.

A descrição da rede fica assim representada:

E = { TrPeçaOut, TrPeçaIn, Movendo, Pegando, Liberando, Prendendo, Largando, Porte, Liberado,PeçaIn, PeçaOut }

C = $\mathbf{C_e} \cup \mathbf{C_s}$ onde:

$\mathbf{C_e}$ = { Ce_1, ETM_e }, onde:
Ce_1 = { Transporte, Entrega_T, Prender_p, Entrega_p, Ct_finaliza }
ETM_e = { Libera_p, Pegar_p, Peça_pr, Peça_lv }

$\mathbf{C_s}$ = { Cs_1, ETM_s }, onde:
Cs_1 = { Ctlibera_p, Retira_E/S, Ctporte_ok, Ctlivre_p }
ETM_s = { Trans_In, Trans_Out, Pegar_p, Liberar_p, E_trab }

A = { **A1**, **A2**, ..., **A10** }, onde :

A1={(TrpeçaOut,Ce_1?Transporte),(a_preparatrans,ETM!Trans_In,PeçaIn,Movendo) }
A2={(TrpeçaIn,Ce_1?Entrega_T),(a_prepentrega,ETM!Trans_Out,PeçaOut,Movendo)}
A3={(PeçaIn,Movendo,ETM?Libera_p),(a_carregarpeça,Cs_1!Ctlibera_p,Pegando)}
A4={(PeçaOut,Movendo,ETM?Pegar_p),(a_liberapeça,Cs_1!Retira_E/S,Liberando)}

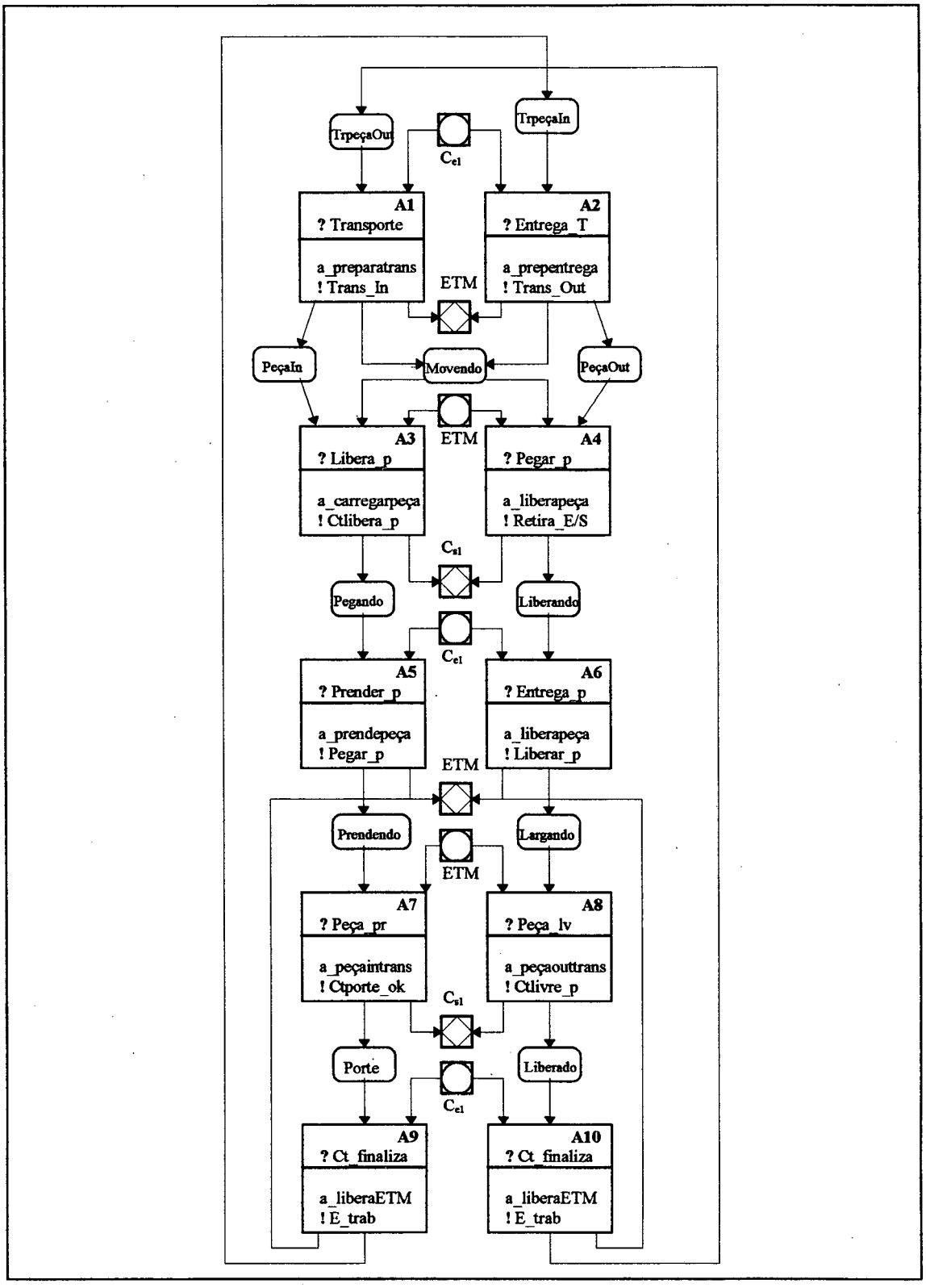


Figura 4.15 : Rede de comportamento genérica para um CtT.

A5 = { (Pegando, Ce_1?Prender_p),(a_prendepeça, ETM!Pegar_p, Prendendo) }

A6 = {(Liberando, Ce_1?Entrega_p), (a_liberapeça,ETM!Liberar_p,Largando)}

A7 = { (Prendendo, ETM?Peça_pr), (a_peçaintrans, Cs_1 !Ctporte_ok, Porte) }

A8={(Largando,ETM?Peça_lv),(a_peçaouttrans, Cs_1 !Ctlivre_p, Liberado)}

A9 ={(Porte, Ce_1?Ct_finaliza), (a_liberaETM, ETM!E_trab, TrpeçaIn)}

A10 = {(Liberado, Ce_1?Ct_finaliza), (a_liberaETM, EMM!E_trab,TrpeçaOut) }

A condição *TrpeçaOut* é atingida quando existe um ETM pertencente ao CtT que está disponível para efetuar um transporte (buscar alguma peça) e o local de carga é um local que o mesmo pode atingir. *TrpeçaIn* habilita a entrega de peça em um local determinado. Desta forma deve existir um ETM com a peça que foi requisitada e cujo local seja possível de ser alcançado.

4.3 Enfoque para a execução dos modelos de comportamento

Pode-se ver a execução de um modelo genérico Rc de um componente de chão-de-fábrica como um processo que atua sobre um *ambiente interno* e interage com um *ambiente externo*. Esta visão mais concreta da execução de um comportamento é descrita da seguinte forma.

O ambiente interno do componente é formado por seus atributos funcionais (variáveis de estado) , um conjunto de estados **E** e seus canais de entrada/saída **Ce** e **Cs**. A cada canal corresponde um conjunto de sinais/mensagens que possuem um nome e um conjunto de informações que definem o conteúdo dos sinais/mensagens. A interação com o ambiente externo é feita através dos canais **Ce** e **Cs** e seus respectivos componentes. O estado inicial de cada processo é definido por uma

operação de inicialização que estabelece valores iniciais para as variáveis afetando assim estados do conjunto **E**.

As atividades definidas no conjunto **A** são realizadas em função da interação com o ambiente externo. O ambiente externo estimula o processo através do envio de sinais/mensagens pelos canais em **Ce**. O processo reage executando atividades que são formadas por ações que podem modificar o valor de variáveis modificando o estado interno do componente e enviando sinais/mensagens para o ambiente externo através dos canais em **Cs**.

Uma atividade é realizada quando o sinal/mensagem definido como disparador da atividade e suas condições de realização (estados internos em **E**) estiverem satisfeitos, ou seja, acontecerem. Neste caso, as ações definidas para a atividade serão executadas, isto inclui a emissão de sinais /mensagens para o ambiente externo. A execução de atividades faz com que novos estados internos, definidos pelo comportamento do componente, sejam atingidos.

A execução do comportamento de um componente pode ser descrita pelo seguinte ciclo:

Espera a chegada de sinais/mensagens do ambiente exterior nos canais especificados. Na chegada executa os seguintes passos:

1) Identifica as atividades que podem ser executadas em função da mensagem recebida e do estado interno.
2) Realiza as atividades identificadas a partir da execução das ações associadas à mesma. Isto significa modificar variáveis, requisitar o envio de mensagens para o ambiente exterior e mudar o estado interno do componente.

As requisições de envio de mensagens para o ambiente exterior são atendidas no final do ciclo e enviadas para os canais desejados. Tanto a recepção quanto o envio de mensagens/sinais são suportadas por um conjunto de serviços de comunicação que permitem a conexão dos vários componentes através dos canais de E/S. Para isto o ambiente de implementação fornece serviços de suporte à comunicação que permitem a interação entre os componentes.

Por outro lado, a construção dos módulos de execução dos controladores de chão-de-fábrica, a partir da utilização dos modelos genéricos, pode ser facilitada pela utilização de um ambiente onde o usuário pode especificar suas necessidades a partir da configuração de seus controladores. As características de extensibilidade e reusabilidade são evidenciadas pelos modelos genéricos. Isto permite :

- a utilização dos modelos genéricos sem modificações, onde o usuário apenas fornece as funções dependentes de implementação necessárias para a execução das ações e as possíveis restrições físicas que possam existir,

- a modificação dos modelos genéricos com o objetivo de diminuir ou aumentar as atividades especificadas e

- a definição de novos componentes a partir da construção de novas redes de comportamento para os componentes introduzidos.

CAPÍTULO 5
PROTÓTIPO DO AMBIENTE DE IMPLEMENTAÇÃO

A partir da definição do modelo de implementação proposto para os controladores de chão-de-fábrica e da definição do modelo de comportamento genérico para os mesmos, este capítulo apresenta uma descrição do ambiente de implementação destes modelos. Este ambiente tem como base o trabalho apresentado em [COR93] e utiliza como plataforma de desenvolvimento o simulador do multicomputador propostono trabalho. São descritos os suportes necessários para a construção e execução dos modelos com base na abordagem distribuída. A construção diz respeito à conversão das especificações em uma representação interna com base em estruturas pré-definidas. A execução significa a realização, passo a passo, das interações entre controladores e controlados (clientes e servidores). O suporte de implementação envolve um conjunto de funções que implementam as características básicas do modelo. Além disso, a interface de usuário possibilita a definição e especificação de parâmetros a partir dos quais o software de controle deve ser construído.

5.1 Estrutura dos componentes de chão-de-fábrica

Como visto no Capítulo 3 os controladores realizam três funções básicas : planejamento, escalonamento e execução. Para executar estas funções os controladores necessitam de um suporte de implementação que é responsável por funções como comunicação e acesso a dados. Este suporte de implementação é caracterizado pelos sistemas de serviço e controle do modelo de implementação. Conforme o modelo de implementação proposto, as funções básicas dos componentes de controle do chão-de-fábrica (controladores) são representadas por

processos, os quais realizam interações do tipo cliente/servidor. A comunicação entre os processos se dá a partir da troca de mensagens através de canais de comunicação pré-definidos. Os processos que representam a função execução do controlador são classificados como mostrado na Figura 5.1.

A classe base da hierarquia de processos dos componentes de execução é a classe ***executor***. Esta classe de processos provê os mecanismos básicos para a execução dos modelos de execução genéricos definidos para os componentes do chão-de-fábrica. A classe de processos ***centro de trabalho*** e ***equipamento*** são especializações da classe ***executor*** com base nas necessidades da função execução dos níveis de Centro de Trabalho e Equipamento. Os processos executores podem pertencer à classe ***centro de trabalho*** (CtP, CtA, CtT) ou à classe ***equipamento*** (EPM, EMM, EAM, ETM), dependendo do nível de controle. Os processos são suportados por um conjunto de serviços básicos que são realizados pelos sistemas de serviços (acesso a dados) e de controle (comunicação, gerência de processos).

A classe de processos ***executor*** tem como estruturas básicas um conjunto de canais de comunicação a partir dos quais o processo se comunica e uma estrutura que descreve o comportamento (ou funcionalidade) a ser executado. Os canais de comunicação estabelecem as ligações que o processo (componente) realiza com o objetivo de trocar mensagens. A estrutura de comportamento descreve a **Rc** do componente que o processo irá executar, ou seja, as atividades que o componente realiza em função do estado interno e das mensagens recebidas. As ações, especificadas pelas atividades que um componente realiza, podem necessitar dos serviços do suporte de implementação. A funcionalidade dos processos da classe ***executor*** inclui as tarefas de construção do comportamento e de execução do comportamento dos componentes representados.

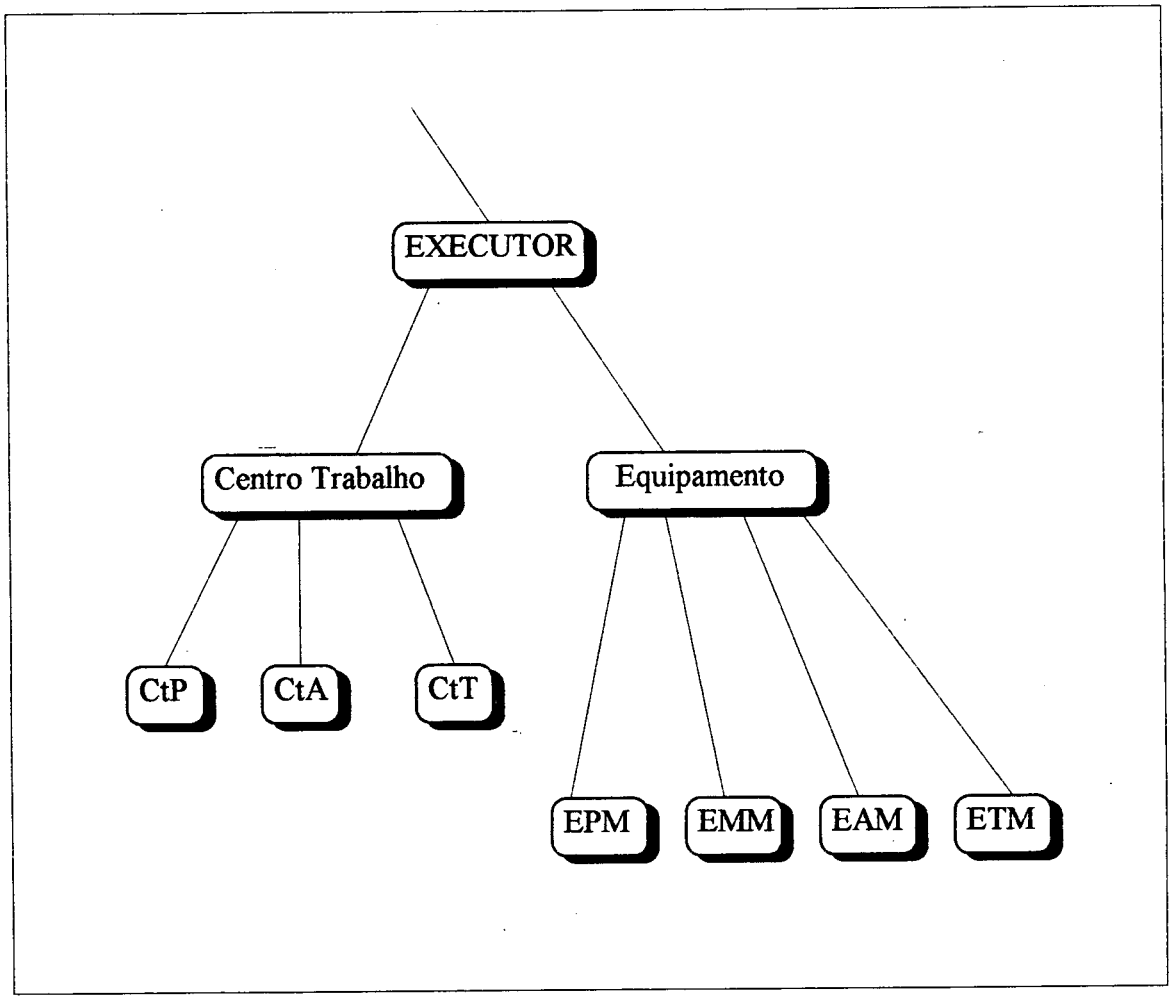


Figura 5.1: Classificação de processos executores.

Os processos da classe ***centro de trabalho*** são criados a partir de um componente de configuração (interface de usuário) que estabelece os componentes do centro (processos que devem ser criados) e determina os canais de comunicação que devem ser estabelecidos. Neste caso o procedimento básico de um processo da classe ***centro de trabalho*** é descrito como segue:

1. Identifica o número de componentes e canais de comunicação a serem estabelecidos.
2. Cria os canais de comunicação e suas respectivas filas de mensagens.
3. Cria os processos representantes dos componentes especificados.
4. Recebe a informação de criação dos processos dos componentes.
5. Constrói o comportamento a ser executado.
6. Executa o comportamento associado.

A partir da execução do comportamento associado, o processo incorpora o comportamento do componente especificado, recebendo requisições de serviço, executando estas requisições e enviando requisições para outros componentes. As conexões estabelecidas por esta classe de processos podem referenciar outros processos da mesma classe ou da classe ***equipamento***.

Os processos da classe ***equipamento*** são criados pelos processos da classe ***centro de trabalho*** dos quais são componentes. Estes processos são criados para realizar as atividades dos componentes do nível de equipamentos. Estes componentes têm capacidade de interagir com outros componentes do tipo equipamento e com o componente representado pelo seu processo criador. Para isto os processos da classe ***equipamento*** estabelecem comunicação a partir da

configuração dos seus canais de comunicação. Neste caso o procedimento básico de um processo da classe ***equipamento*** é descrito como segue:

1. Identifica os canais de comunicação a serem estabelecidos.
2. Cria os canais de comunicação e suas respectivas filas de mensagens.
3. Constrói o comportamento a ser executado.
4. Executa o comportamento associado.

As tarefas de construção do comportamento do componente e de execução do mesmo são realizadas por cada um dos processos criados e serão apresentadas a seguir.

5.2 Construção e execução do comportamento

O comportamento dos componentes controladores de centro de trabalho e de equipamento, são descritos a partir da utilização das Redes de comportamento. A idéia das Rc's é estabelecer uma descrição não-ambígua dos componentes de um sistema, de forma independente da linguagem de implementação. Esta descrição estabelece uma visão que especifica a funcionalidade e o comportamento dos componentes de um sistema de forma independente e livre de detalhes de implementação. Do ponto de vista de implementação, é necessário dar condições para que estas descrições sejam executadas permitindo que se consiga o comportamento especificado nos processos que representam os componentes do sistema. Desta forma, é preciso estabelecer procedimentos para que se possa, primeiro, construir o comportamento dos componentes a partir da descrição contida na Rc e, segundo, executar este comportamento de forma consistente com o estabelecido pela descrição.

5.2.1 A construção do comportamento

A construção do comportamento é uma tarefa genérica realizada pelos processos executores do comportamento dos componentes. Basicamente, a construção do comportamento consiste na leitura do comportamento descrito pelas Rc's e a tradução destas informações em estruturas de dados computacionalmente adequadas. Estas estruturas devem conter as informações necessárias para que o comportamento especificado na Rc possa ser executado. As estruturas devem conter a descrição dos eventos que podem ocorrer durante a execução do componente e a descrição das atividades a serem realizadas na ocorrência de um determinado evento. Desta forma, duas estruturas principais são construídas a partir das informações da Rc do componente que o processo executa. A primeira, ***Comportamento***, representa o comportamento do componente no que diz respeito aos eventos que chegam e as atividades que devem ser realizadas. A segunda, ***Atividades***, representa as atividades realizadas pelo componente.

A estrutura ***Comportamento*** é dirigida pela ocorrência das mensagens do componente e é formada por um par (*mensagem, estado*) que define uma mensagem e as condições da sua ocorrência. Desta forma, quando *mensagem* ocorre e as condiçoes atuais do componente casam com *estado*, a estrutura aponta para a atividade a ser realizada.

A estrutura ***Atividades*** apresenta, para cada atividade apontada pela estrutura ***Comportamento***, quais as ações e atribuições de saída devem ser realizadas. As ações são procedimentos que auxiliam na execução das atividades, por exemplo, numa atividade de carga de uma peça num *buffer*, a ação pode ser atualizar o número de peças que existem no mesmo. As atribuições de saída dizem respeito às

mensagens a serem enviadas para outros componentes (respostas ou novas requisições) e a ativação de novas condições de ocorrência para novas mensagens (mudança de estado).

Estas duas estruturas de dados são conseguidas a partir da leitura de um arquivo que contém as informações sobre a rede de comportamento. Cada classe de componentes definida tem o seu arquivo de definição do comportamento. O arquivo tem duas seções básicas: definições e descrição do comportamento. A primeira seção contém alguns parâmetros que definem a Rc: tipo do componente, estados e mensagens. A segunda seção apresenta os pares (*mensagem, estado*) e suas respectivas atividades, e as atividades com suas ações, mensagens e estados. Considerando o componente da classe EAM, o arquivo de definição do comportamento é definido como segue :

```
//* definições
/tipo       EAM
/canais     2

EMM         E3    S4
Ce          E2    S2

/estados    6
ArmOn       1
PeçaIn      0
Pegando     0
Liberando   0
Porte       0
PeçaOut     0
```

/msgin 5

Arm_In	1
Arm_Out	1
Pegar_p	2
Libera_p	2
E_trab	2

/msgout 6

ArmIn_ok	1
ArmOut_ok	1
Coloca_p	2
Retira_p	2
Peça_lv	2
Peça_pr	2

// comportamento

/ [mensagem, condições] atividade

[Arm_In, ArmOn]	A1
[Pegar_p, Pegando]	A3
[E_trab, Porte]	A5
[Arm_Out, PeçaIn]	A2
[Liberar_p, Liberando]	A4
[E_trab, PeçaOut]	A6

/atividades 6

[a_alocalugar]	[Coloca_p]	[Pegando]
[a_liberalugar]	[Retira_p]	[Liberando]
[a_inserepeça]	[Peça_pr]	[Porte]
[a_liberapeça]	[Peça_lv]	[PeçaOut]
[a_cargaok]	[ArmIn_ok]	[ArmOn,PeçaIn]
[a_liberapeça]	[ArmOut_ok]	[ArmOn]

O construtor do comportamento inicializa as estruturas de dados ***Comportamento*** e ***Atividades*** a partir da leitura do arquivo de entrada. Estas estruturas estão constituídas da seguinte forma:

Estrutura ***Comportamento***

Mensagem_de_entrada	: *MsgIn*
Marcação_de_espera	: {*estados*}
Atividade	: *índice_de_* ***Atividades***

Fim

Estrutura ***Atividades***

Ação	: {*nome_de_procedimento*}
Mensagem_de_saída	: *MsgOut*
Marcação_de_saída	: {*estados*}

Fim

5.2.2 A execução do comportamento

Após a construção e inicialização das estruturas de dados que descrevem o comportamento de um componente, o executor do comportamento assume o controle

do processo associado ao componente. O ciclo principal de execução do executor inclui a espera pela ocorrência de mensagens, a identificação da mensagem ocorrida e a execução das atividades associadas.

O processo executor inicialmente espera pela ocorrência de eventos (mensagens) que chegam através de seus canais de comunicação. Esta espera é realizada por uma primitiva de comunicação que identifica a chegada de mensagens nos diferentes canais de comunicação associados ao processo. Na chegada de uma mensagem o processo é desbloqueado para identificação da mesma. Tanto as mensagens originadas em outros componentes do sistemas (controladores) quanto aquelas que são originadas pelos componentes planejamento/escalonamento são recebidas a partir dos canais de comunicação.

A identificação das mensagens recebidas é realizada com o auxílio da estrutura de dados ***Comportamento***. O processo deve verificar se, dada uma mensagem, as condições para a execução da atividade associada estão satisfeitas. Diferentes mensagens podem ser aceitas em um determinado instante, mas a combinação (*mensagem, estado*) determina a execução de uma atividade apenas, evitando assim situações de conflito. Entretanto, o paralelismo de execução de tarefas globais de um componente não é afetado. A possibilidade de execução de tarefas paralelas é determinada pela existência das condições de execução das mesmas, o que pode incluir os recursos e restrições físicas existentes. Por exemplo, se um componente EPM de um centro de trabalho tem uma peça pronta para ser processada e outra pronta para ser retirada (tem pontos de entrada, saída, processamento) e estas duas tarefas não são conflitantes (uma operação não interfere na outra), a descrição do comportamento permite a realização destas duas tarefas globais em paralelo. Em resumo, a execução é dirigida pelas requisições de serviço que o componente recebe e pelas condições de atendimento das mesmas.

A execução da atividade identificada inclui: a execução das ações que estão associadas à atividade, o envio de mensagem para outros componentes e o estabelecimento das condições de saída. As ações são procedimentos que atualizam as estruturas de dados, buscam informações necessárias para a sua execução (arquivos) e estabelecem restrições de execução. Algumas ações são dependentes de dispositivo. Neste caso para cada dispositivo específico deverá existir a ação correspondente. As mensagens são enviadas através dos canais de comunicação e usam as primitivas de comunicação disponíveis no suporte de implementação. As condições de saída estabelecem a ativação de novos estados da Rc (marcação de lugares internos) em resposta à realização de uma atividade Estas etapas são realizadas com o auxílio das funções disponíveis no suporte de implementação. O executor realiza o ciclo mostrado a seguir :

Procedimento *Executor*

Faça

Espera requisição
Identifica requisição
Se *existe atividade para ser realizada*
Executa ações
Envia mensagens de saída
Atualiza condições de saída

Fim Faça

5.3 O suporte de implementação

Considerando o modelo de implementação apresentado no Capítulo 3, o suporte de implementação deve prover os serviços necessários para a implementação dos processos que representam os componentes do sistema. O suporte de implementação é parte integrante de cada um dos processadores (processos) que executam o comportamento dos componentes e oferece um conjunto de serviços básicos para os processos. O suporte de implementação aqui descrito é baseado em [COR93] [MON95] [CAM95].

O suporte de implementação é composto por dois sistemas básicos : sistema de controle (núcleo de controle) e sistema de serviços. O sistema de controle oferece as funções necessárias para permitir a implementação do modelo distribuído. O sistema de serviços define funções auxiliares para a execução dos comportamentos dos componentes de controle. A Figura 5.2 mostra os componentes do ambiente de implementação proposto.

Cada processador ou nó contém: o componente (comportamento) que está sendo executado, um conjunto de serviços locais e um núcleo que é parte do sistema de controle. Dois processadores têm funções específicas: o gerente de comunicação ou controlador de conexões é responsável pela gerência dos nós e estabelecimento das conexões entre os mesmos; e o nó servidor externo permite a comunicação com a estação de trabalho que fornece serviços do sistema de arquivos para os processos. Neste caso, a estação de trabalho funciona como interface entre o usuário e o sistema de arquivos.

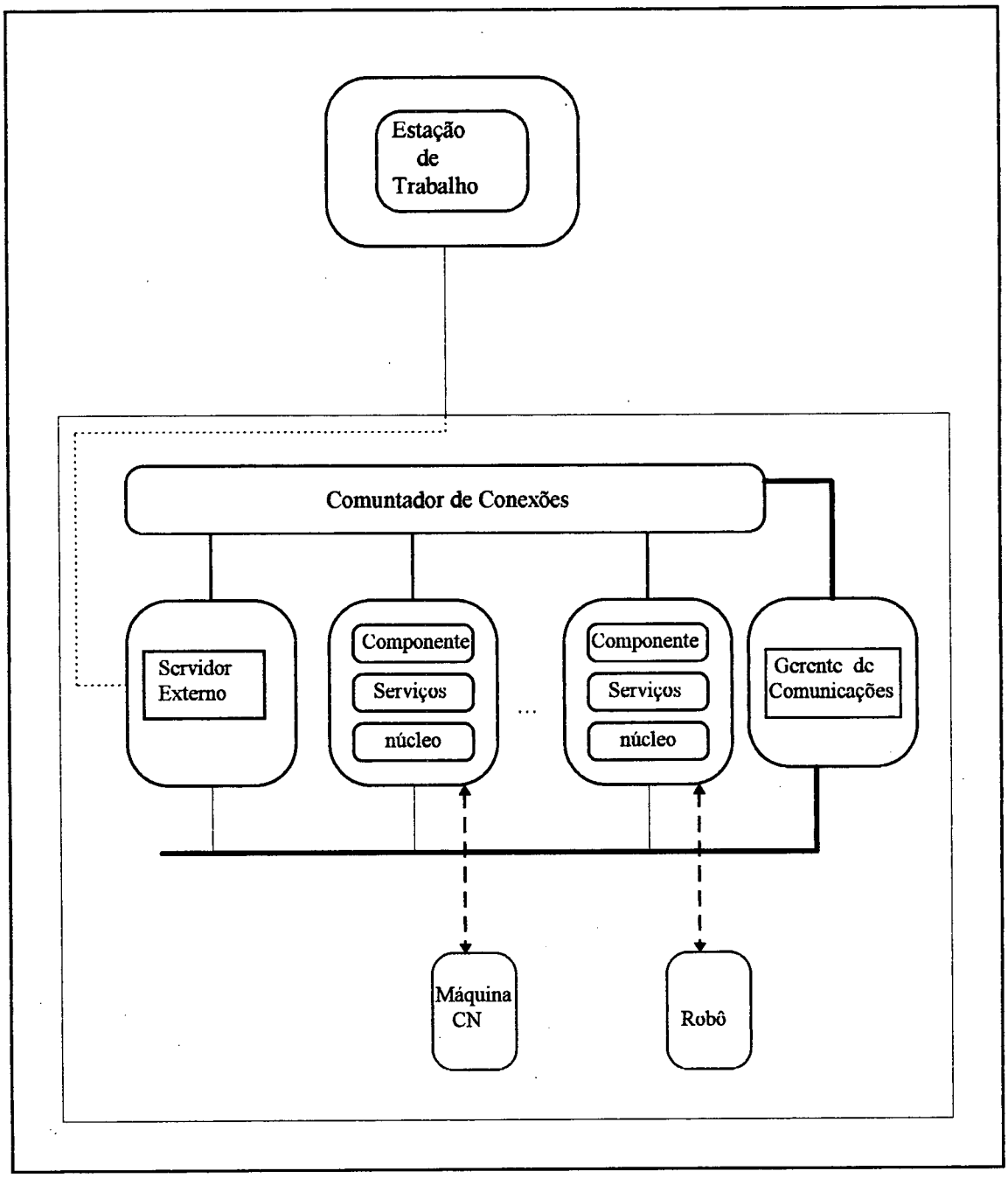


Figura 5.2 : Componentes do Ambiente de Implementação.

5.3.1 O sistema de controle

O sistema de controle fornece primitivas de controle para gerência de processos e para gerência de comunicações. Além disto, existe um conjunto de primitivas que fornece o suporte básico para a conexão entre os diferentes processadores (nós). A comunicação com o gerente de comunicações (GC) é feita a partir da utilização de uma função que envia uma mensagem e espera resposta. Esta função está assim definida :

- *BS_SendRec (msg)* Envia uma mensagem para o GC e aguarda resposta

O recebimento de uma mensagem deste tipo faz com que o GC atenda a requisição contida em *msg* e envie uma resposta. Para isto o GC possui internamente duas outras funções *BS_ReceiveAny* e *BS_Send.*

As primitivas de gerência de processos, reponsáveis pela criação e remoção de processos, são:

- *CreateProcess (nid)* Cria um processo no nó *nid*
- *CreateAnyProcess(nid)* Cria um processo num nó qualquer
- *RemoveProcess()* Remove um processo

Estas primitivas permitem que os processos executores do comportamento dos componentes possam ser criados e removidos de forma dinâmica. Por exemplo, um processo do tipo *centro de trabalho* cria um processo *equipamento* para cada um dos seus componentes e recebe os identificadores dos nós (*nid*) onde os mesmos

foram alocados. Estas funções são executadas através da requisição dos serviços do controlador de conexões. Os serviços necessários são :

- *Allocate (nid)* Aloca um nó específico *nid*
- *AllocateAny()* Aloca um nó qualquer
- *Deallocate ()* Libera o nó corrente

A comunicação entre os processos no modelo de implementação proposto acontence através dos canais de comunicação que são estabelecidos pelas redes de comportamento. Cada componente tem pelo menos um canal de comunicação através do qual recebe as mensagens que estabelecem a sua reação e outros canais de comunicação através dos quais envia suas requisições de serviço. Por exemplo, um controlador de centro de trabalho CtP cuja configuração inclui um controlador de equipamento EMM e dois controladores de equipamento EPM, tem um canal de comunicação *cCtP* através do qual receberá as mensagens dos outros componentes (EMM, EPM, outros Ct's) e as mensagens de saída serão enviadas para os canais dos componentes com os quais o CtP interage (EMM, EPM, etc.).

As funções usadas pelos processos para comunicar-se através dos canais estão assim definidas :

- *Send (canal,requisição, tam_req)* : envia ao canal de comunicação designado por *canal*, uma mensagem. O processo que está enviando a mensagem fica esperando que o processo responsável pelo atendimento do canal execute uma primitiva de recepção de mensagem.

- *Receive (canal,requisição,tam_req)* : recebe uma mensagem de um canal de comunicação designado por *canal*. O processo permanece bloqueado até que um processo qualquer envie uma mensagem para o canal especificado.

Estas funções são executadas a partir da criação dos canais de comunicação pelo gerenciador de comunicações. A comunicação entre os processos é suportada por um conjunto de primitivas que permite a conexão dos processadores (nós). As funções de suporte para a criação de canais de comunicação e conexão de processadores são :

- *Create(canal)* : requisita para o gerente de comunicações a criação de um canal de comunicação cujo identificador será retornado em *canal*.

- *Remove(canal)* : requisita a remoção do canal de comunicação *canal* para o gerente de comunicações.

- *Connect (canal)* Estabelece conexão com um canal

- *Disconnect ()* Libera conexão existente

O suporte de implementação para o conceito de canais de comunicação necessita, além das primitivas apresentadas, uma estrutura de dados que permita o gerenciamento das comunicações. Este gerenciamento é responsabilidade do gerente de comunicações o qual mantém a estrutura de dados, mostrada na Figura 5.3 , que define os canais de comunicação.

A estrutura *TabCanais* define para cada canal *j* (onde *j* é o índice da tabela) uma lista encadeada de mensagens que devem ser atendidas neste canal.

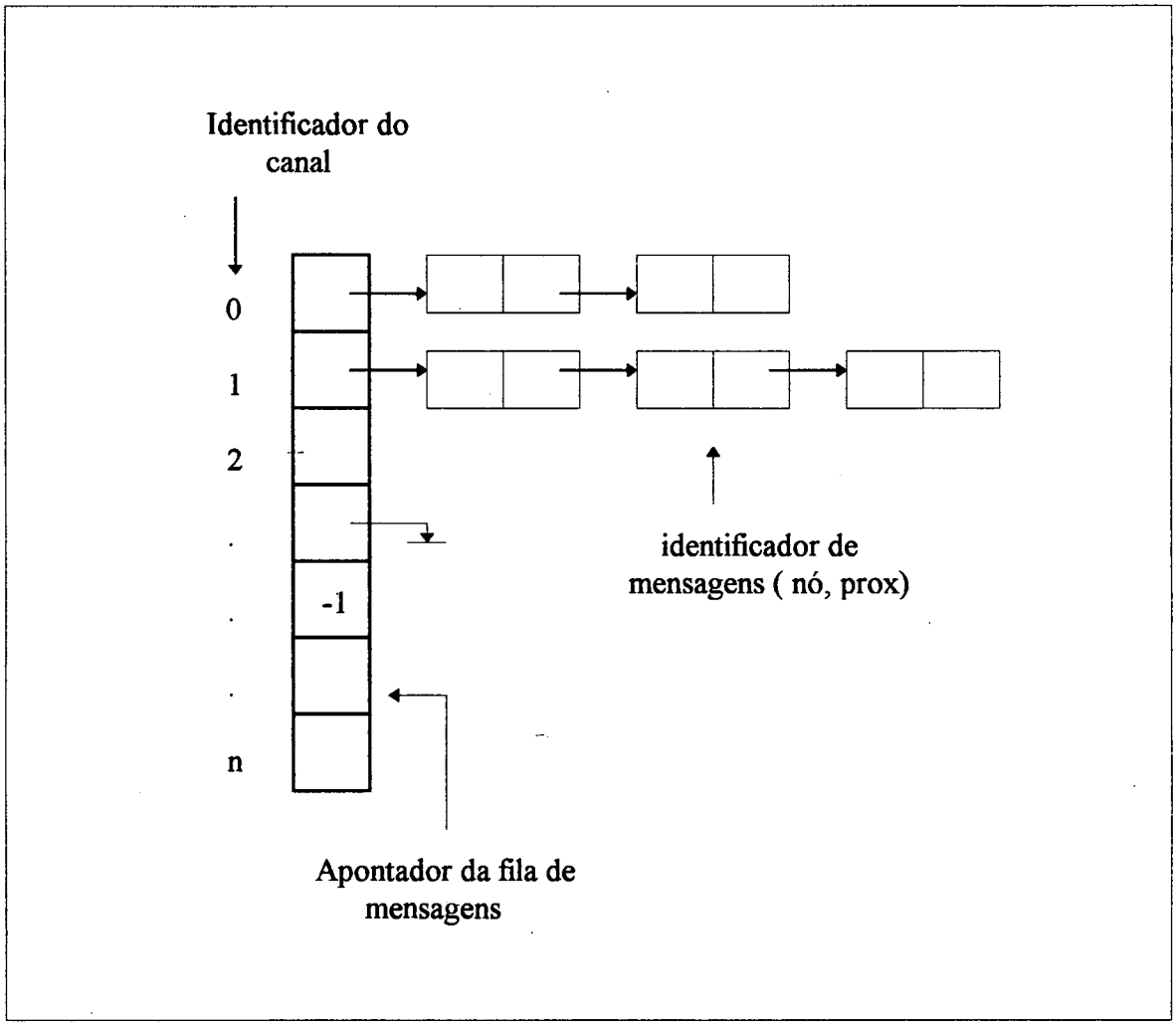


Figura 5.3 : Estrutura de dados para os canais de comunicação.

Cada elemento da lista contém o identificador do nó que está enviando a mensagem e o tipo de operação. Desta forma, é possível o envio de mensagens para um determinado canal mesmo que ele esteja sendo usado por outro processo. As mensagens que chegam a um determinado canal são atendidas a partir de uma política de atendimento do tipo FIFO (First In First Out). Inicialmente, os canais não criados contêm um valor negativo no campo que identifica a primeira mensagem da lista. Assim, a partir da sua criação este campo recebe um valor que identifica a sua criação. Remover uma canal significa atribuir novamente um valor negativo a este campo.

As funções que o sistema de controle fornece como suporte de implementação é responsabilidade do núcleo e do gerente de comunicações.

5.3.2 O sistema de serviços

O que é oferecido pelo sistema de serviços inclui funções gerais de interface de programação, assim como funções do sistema de arquivos. No que diz respeito ao sistema de serviços, as funções podem ser divididas em dois tipos principais : funções executadas localmente e funções executadas pelo *servidor externo*.

As funções resolvidas localmente envolvem informações que residem no próprio nó. Por exemplo, cada processo tem uma estrutura de dados associada que o descreve; esta estrutura contém informações sobre o processo que podem ser acessadas a partir de primitivas de serviço e estão disponíveis localmente ao processo. Uma destas primitivas *getpid* devolve ao processo o seu próprio identificador. O conjunto de ações que um componente executa, quando está realizando as suas atividades, também é parte integrante das funções do sistema de serviços executadas localmente. Entretanto, a execução de algumas funções pode

necessitar de serviços do *servidor externo* . Por exemplo, uma ação que necessita ler algum arquivo (arquivo CN) para enviá-lo a alguma máquina, precisará do *servidor externo* para acessar o arquivo.

As funções que não podem ser resolvidas localmente são encaminhadas para o *servidor externo*. Como exemplo de funções executadas pelo *servidor externo* têm-se as funções do sistema de arquivos que permitem ao processo o acesso a dados que estão disponíveis em arquivos. Neste caso, o *servidor externo* atende todas as requisições dos processos. Na arquitetura considerada, o *servidor externo* comunica-se com a *estação de trabalho* para executar as requisições. Entretanto, se o *servidor externo* apresentar alguma capacidade de armazenamento, é possível atender algumas das requisições no próprio servidor. Desta forma, a comunicação com a *estação de trabalho* só será necessária quando o servidor não estiver habilitado a atender uma requisição.

5.4 A Interface com o usuário

Com o objetivo de permitir que a tarefa de especificação do controle de chão-de-fábrica seja facilitada, o ambiente de implementação fornece uma interface de usuário que permite a definição dos componentes de controle, a partir de menus de escolha. A nível de protótipo, o ambiente permite ao usuário a especificação do sistema de controle através da escolha dos elementos que compõem o mesmo. Por exemplo, na especificação de um centro de trabalho o usuário pode escolher os componentes que fazem parte do mesmo (EPM, EMM, EAM, etc.). A partir daí, o ambiente estabelece a estrutura do sistema no que diz respeito à interação entre os componentes. Além disso, a interface permite que o usuário possa visualizar o estado de operação dos componentes de controle no nível de centros de trabalho. Por

exemplo, num determinado instante, o usuário poderá verificar qual é o estado de cada um dos componentes de um centro de trabalho. É importante deixar bem claro que o objetivo principal da interface de usuário, no protótipo do ambiente de implementação, é demonstrar as características de configurabilidade e flexibilidade da abordagem proposta.

O ambiente de implementação, através de sua interface de usuário, possibilita: a configuração dos centros de trabalho a partir da escolha dos elementos componentes do centro e da inicialização das informações pertinentes a cada um dos componentes (componentes dependentes do sistema) e a execução do programa de controle gerado a partir das especificações fornecidas pelo usuário.

5.4.1 A seção de configuração

Os componentes de controle são informados a partir da escolha do usuário, que é guiada pelo conjunto de componentes genéricos disponíveis no sistema. Para cada componente identificado, o usuário fornece informações que caracterizam o componente a ser instanciado. Por exemplo, na especificação de um centro de trabalho de processamento (CtP), o usuário fornece as informações (atributos) referentes ao centro que está sendo instanciado : equipamentos componentes do centro, plano a ser executado, etc. No caso de um CtP, a especificação dos componentes é feita com base nos tipos de equipamentos que podem estar agrupados neste tipo de centro. Para cada equipamento epecificado, o usuário informa o nome do equipamento e escolhe a classe de controlador associada. A Figura 5.4 mostra o tipo de interação com o usuário em uma seção de configuração de um controlador de centro de trabalho.

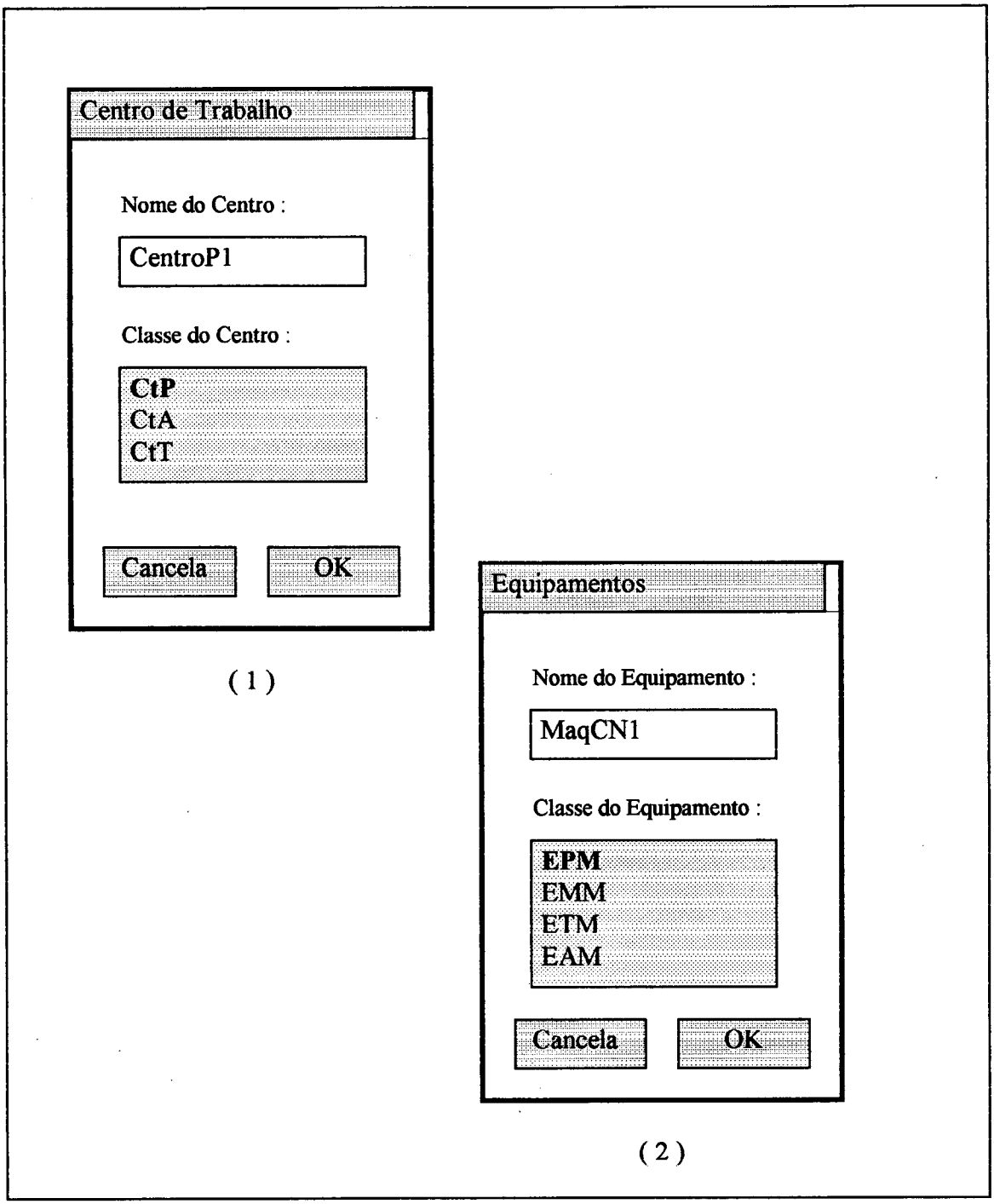


Figura 5.4 : Configuração de Centro de Trabalho.

O sistema inicia com a opção de especificação dos centros de trabalho. O usuário especifica o nome do centro de trabalho e a classe que o mesmo representa (1). A seguir, o usuário pode especificar os componentes do CtP *CentroP1*, sendo que, para cada nome de componente, deve existir uma classe de equipamentos associada (2) . Na especificação de CtP composto por duas máquinas CN e um robô, a configuração final é mostrada a seguir :

Início

Nome_Do_Centro = *CentroP1*

Classe = *CtP*

Rota = *Plano5*

Ponto_de_E/S = *L3*

N_Componentes = *3*

Nome	**Tipo**	**Local**	**Capacidade**
MaqCN1	*EPM*	*L1*	*1*
MaqCN2	*EPM*	*L2*	*1*
Robô	*EMM*	*L3*	*1*

Fim

A especificação apresentada pode ser armazenada em arquivo para posterior execução ou modificação. A seção de configuração também permite a inicialização das informações (atributos) dos componentes, ou a modificação das mesmas e a configuração dos planos de processo, que caracterizam os componentes dependentes de aplicação. Os planos de processo (rotas de produção) especificam o caminho produtivo em um centro de trabalho. Os planos estão associados às peças e, para cada peça, existe pelo menos um caminho. A configuração dos planos segue o mesmo esquema da configuração dos componentes dependentes do sistema. A

ligação entre os planos e o componente executor do controlador é realizada a partir da comunicação com os componentes planejador/escalonador. A execução de uma determinada peça por parte do centro de trabalho faz com que os componentes planejamento/escalonamento determinem o plano a ser seguido. A partir de então, o componente execução realiza o plano determinado. Cada nível de controle pode ter o seu plano de processo. Vamos considerar aqui, apenas o nível de controle de centro de trabalho, ou seja, a sequência das operações que são executadas no centro. Por exemplo, considere-se um controlador de CtP com dois robôs, duas máquinas CN e três *buffers*; a representação do plano de processo de uma peça poderia ser assim:

Peça : B1 R1 → M1 R1 → B2 R2 → M2 R2 → B3

onde B1, B2 e B3 são os *buffers* , M1 e M2 as máquinas e R1 e R2 são os robôs.A representação indica que o robô R1 deve mover a peça de B1 para M1, depois de M1 para B2; o robô R2 deve mover a peça de B2 para M2 e depois, de M2 para B3. A chegada de uma peça no centro de trabalho envolve outros centros, por exemplo, o centro de transporte, que entrega a peça no ponto de entrada do CtP (B1). Da mesma forma, a partir do ponto de saída (B3) a peça é transportada para fora do CtP.

5.4.2 A seção de execução

A execução é responsável pela inicialização dos componentes do sistema de controle e pela criação dos processos representantes dos controladores de centro. Neste caso, a interface com o usuário pode enviar requisições para os controladores de centro como se fosse ela o supervisor dos mesmos. No protótipo, estas requisições são estabelecidas pelo operador. A Figura 5.5 mostra o resultado de uma requisição do tipo *status* para um CtP.

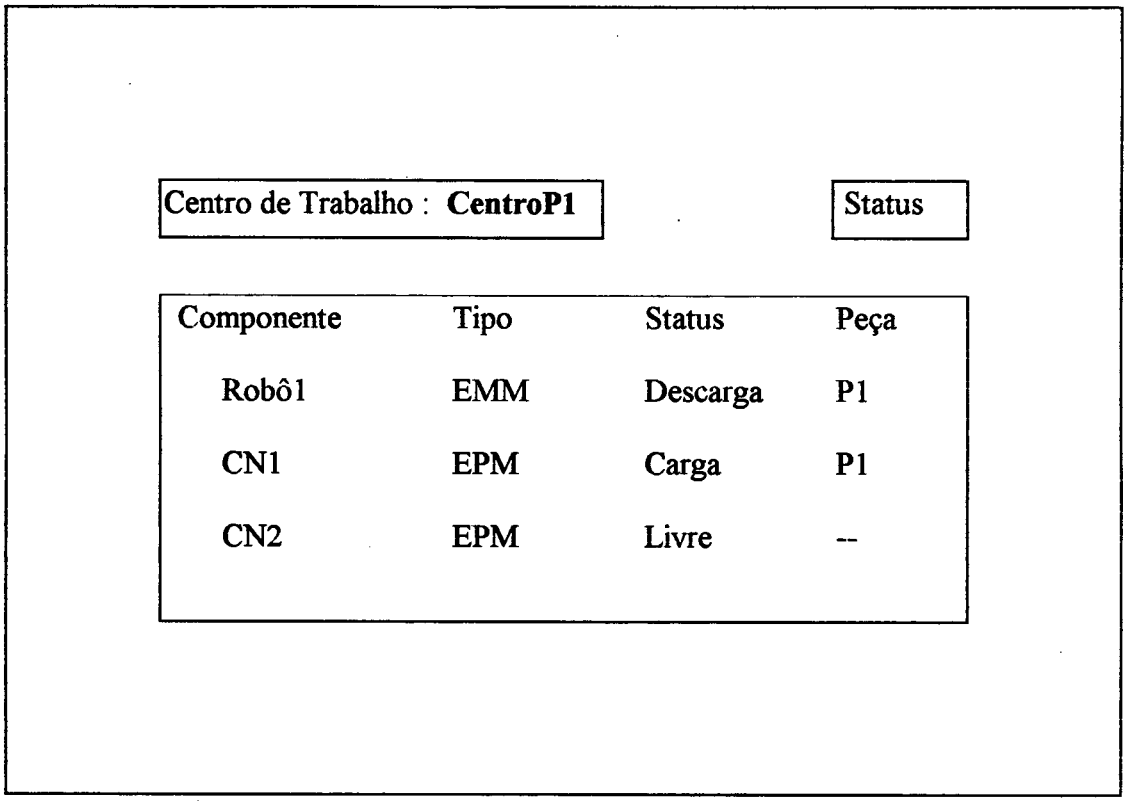


Figura 5.5 : Exemplo de uma requisição de status.

O processo interface de usuário é executado na estação de trabalho e comunica-se com os controladores através do *servidor externo*. As requisições de *status* ou qualquer outra requisição que não seja de controle não fazem parte do comportamento dos componentes descritos pelas Rc's. Entretanto estas requisições são atendidas a partir da identificação do tipo da mensagem.

Após a criação dos processos executores dos centros, estes passam a executar o seu ciclo de vida, ou seja, criam os processos componentes (filhos) e passam a atender as requisições que são recebidas pelos canais de comunicação. Estas requisições podem demandar serviços dos componentes do seu centro ou de outros centros. Por exemplo, numa tarefa de retirada de uma peça de um centro de processamento, o CtP requisita os serviços de um centro de transporte para a execução da mesma.

CAPÍTULO 6
CONCLUSÕES

A abordagem proposta neste trabalho fornece uma estrutura para o desenvolvimento do software de controle para sistemas de manufatura celular. A abordagem utiliza um modelo distribuído onde os componentes de controle são especificados de forma individual e depois ligados através de canais de comunicação para formar o sistema de controle. O trabalho envolveu as seguintes etapas principais:

- A definição da arquitetura de controle adotada que inclui a especificação de seus componentes e a estrutura dos mesmos.

- A especificação de modelos genéricos de execução das tarefas dos componentes de controle do sistema. O comportamento destes componentes é especificado através da utilização das redes de comportamento. Os modelos genéricos são utilizados como ponto de partida para a implementação do controle em sistemas de manufatura celular.

- A definição do modelo de implementação que suporte o modelo distribuído adotado e forneça os recursos necessários para a sua implementação.

- A construção de um protótipo do ambiente de implementação incluindo os mecanismos definidos para o modelo de implementação e uma interface de usuário que permite a configuração dos componentes de controle.

A arquitetura de controle adotada define três camadas de controle onde a comunicação entre componentes de uma mesma camada é permitida. Os camadas de centros de trabalho e equipamento foram especificadas incluindo seus componentes e estrutura.

A especificação dos modelos de execução dos componentes de controle resultou num conjunto de redes de comportamento que estabelecem de uma forma genérica o comportamento destes componentes.

O modelo de implementação definido estabelece um modelo distribuído de execução onde os componentes de controle comunicam-se para executar tarefas atribuídas ao sistema.

Na construção do protótipo do ambiente de implementação é utilizado um conjunto de funções que permitem a implementação do modelo distribuído adotado, ou seja, a possibilidade de execução dos componentes de controle de forma paralela com comunicação através de mensagens. Além disto, estabelece uma interface de usuário através da qual os componentes podem ser escolhidos e conectados.

São duas as contribuições principais deste trabalho. A primeira se relaciona com a especificação dos componentes de controle genéricos (redes de comportamento) através da definição do comportamento que os mesmos apresentam em função das requisições de serviço recebidas. Partindo da especificação de modelos genéricos o sistema de controle é conseguido a partir da ligação dos vários componentes de controle através das suas interfaces.

A segunda contribuição é importante do ponto de vista prático. A abordagem aqui proposta pode ser considerada como um procedimento que permite a construção e implementação de controle para sistemas de manufatura celular. Na construção são estabelecidos os componentes de controle e as suas ligações. A implementação do modelo construído é sustentada pelo modelo de implementação que estabelece um conjunto de funções e características que são necessárias para a execução dos modelos. Como cada componente do sistema é construído como um processo comunicante, o sistema pode facilmente mudar a sua configuração mudando as ligações estabelecidas entre os componentes.

Os componentes de controle são considerados como sendo dirigidos por eventos, por isto, restrições de temporização não são consideradas. Entretanto, para representar os aspectos de tempo-real, parâmetros extras devem ser adicionados na definição das atividades dos componentes e da ocorrência de seus eventos ativadores.

Com base nos resultados do trabalho aqui apresentado, futuras pesquisas podem ser realizadas para dar continuidade à idéia de se ter um ambiente de desenvolvimento de sistemas de controle genérico aliado a um suporte de implementação com características distribuídas. Os seguintes temas podem ser considerados possíveis extensões ao trabalho apresentado:

- O desenvolvimento dos módulos de planejamento e escalonamento dos componentes de controle. A partir da definição dos modelos de execução dos componentes de controle os módulos de planejamento e escalonamento podem ser definidos e incorporados ao sistema de controle a partir da utilização dos serviços definidos para os módulos de execução.

- O gerente de comunicações do ambiente de implementação atualmente é responsável pela comunicação entre os processos controladores de equipamentos e centros de trabalho. Um mecanismo que permita os controladores de equipamento se comunicarem com os respectivos dispositivos físicos, deve ser providenciado.

- A interface de usuário utilizada no protótipo teve como principal propósito permitir a configuração do sistema de controle. Sendo assim a incorporação de novas funções que permitam uma maior flexibilidade para o usuário em termos de definição e reconfiguração dos modelos de comportamento deve ser considerada.

- A possibilidade de descrição dos modelos genéricos , através de uma linguagem que represente as redes de comportamento, estabelece uma perspectiva no sentido de que os modelos possam passar por um processo de compilação e geração de código, compatível com o modelo de implementação. Desta forma, pode-se eliminar a etapa de construção do comportamento que os processos realizam antes de iniciar sua execução.

- Considerando que existem ações específicas para os diferentes tipos de dispositivos de chão de fábrica, a criação de uma biblioteca de ações representando estes tipos permitiria que os processos controladores de equipamentos pudessem ser configurados a partir da escolha do dispositivo a ser controlado.

Embora a estrutura de desenvolvimento e o ambiente de implementação apresentados representem um avanço na busca de uma nova geração de controladores de chão-de-fábrica, esforços adicionais devem ser realizados no sentido de verificar o comportamento da abordagem apresentada em um ambiente real de manufatura.

BIBLIOGRAFIA

[ADI89] Adiga, S., "Software Modelling of Manufacturing Systems: A case for an Object-Oriented Programming Approach". Annals of Operations Research, 17(363-378), 1989.

[AHO86] Aho, Alfred V.; Sethi, Ravi; Ullman, Jeffrey D., *Compilers: Principles, Techniques and Tools*. New York, Addison Wesley, 1986.

[ALF85] Alford, M.,"SREM at the Age of Eight: The Distributed Computing Design System". IEEE Computer, April 1985, pp. 36-46.

[ANG87] Ang, C. L., "Factory Data Communications". Computers in Industry, No. 9, 1987.

[AYR89] Ayres, R. U., "Technology Forecast for CIM". Manufacturing Review, Vol.2, No. 1, 1989, pp. 43-52.

[BAB89] Babuder, Dave, "Determining the Requirements for Cell Control Through Actual Applications". Proceedings of the 18th Annual Programable Controllers Conference, 1989.

[BAL91] Baldassari, M. and Bruno, G., "PROTOB : An Object Oriented Methodology for Developing Discrete Event Dynamic Systems". Computing Languages, Vol. 16, No.1, pp. 39-63, 1991.

[BAN88] Banerjee, S. K. and Al-Maliki, I., "A Structured Approach to FMS Modelling". International Journal of Computer Integrated Manufacturing, Vol. 1, No. 2, 1988, pp. 77-88.

[BAR94] Barbeau, M.; Custeau, G. and Saint-Denis, R., "Requirements engineering and synthesis of a control system". APII, Vol. 28, No. 1, 1994, pp. 37-52.

[BAS94] Bass, J. M.; Browne, A. R.; Croll, P. R. and Fleming, P.J., "A Prototype Framework of Tools for the Design of Real-Time Distributed Control Software". In: International Conference on Control'94, March, 1994, pp. 922-927 vol.2.

[BAS93] Bastide, R.; Sibertin-Blanc, C.; Palanque, P., "Cooperative Objects: A Concurrent, Petri-Net Based, Object-Oriented Language". Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol.3, pp.286-291, 1993.

[BED91] Bedworth, David D.; Henderson, Mark R. and Wolfe, Philip M., *Computer-Integrated Design and Manufacturing*. McGraw Hill, New York NY, 1991.

[BIG87] Bigou, J. M.; Courvoisier, M. Y. P.; Demmou, H.; Desclaux, C.; Pascal, J. C. and Valette, R. J., "A Methodology of Specification and Implementation of Distributed Discrete Control Systems". IEEE Transactions on Industrial Electronics, Vol.IE-34, No.4 (November),1987, pp. 417-421.

[BIE89] Biemans, F. and Vissers, C.A.,"Reference Model for Manufacturing Planning and Control Systems". Journal of Manufacturing Systems, Vol. 8, No. 1, 1989, pp. 35-46.

[BIR94] Birkinshaw, C. I.; Croll, P.R.;Marriott, D.G. and Nixon, P. A., "Parallel Processing: A Safer Option for Real-Time Control Software", In: International Conference on Control'94, March, 1994, pp. 916-921.

[BOU91] Boulet, B. et alli, "Cell controllers: Analysis and comparision of three major projects". Computers in Industry, Vol. 16, 1991, pp. 239-254.

[BRA89] Bradley, Michael S., "Cell Control - What Do Users Really Want". Proceedings of the 18th Annual Programmable Controllers Conference, 1989.

[BRA91] Brazil - Country Report No. 3, 1991, The Economist Intelligence Unit, London, 1991.

[BRO94] Browne, A. R. et alli, "A Prototype Framework of Design Tools for Computer-Aided Control Engineering". In: Proceedings IEEE/IFAC Joint Symposium on Computer-Aided Control System Design, Tucson (March), 1994, pp. 369-374.

[BRU85] Bruno, G. and Marchetto, G., "Rapid Prototyping of Control Systems Using High Level Petri Nets". The Proceedings of the 8th International Conference on Software Engineering, pp. 230-235 (August 1985).

[BRU86a] Bruno, G. and Balsamo, A., "Petri Net-Based Object-Oriented Modeling of Distributed Systems", OOPSLA '86: Object-Oriented Programming Systems, Languages and Applications, Portland, Oregon, pp. 284-293, 1986.

[BRU86b] Bruno, G. and Marchetto, G., "Process-Translatable Petri Nets for Rapid Prototyping of Process Control Systems". IEEE Transactions on Software Engineering, Vol. SE-12, No.2 (February),1986,pp. 346-357.

[BRU88] Bruyn, W. et alli, "ESML: An Extended Systems Modeling Language Based on the Data Flow Diagram", ACM SIGSOFT Software Engineering Notes, Vol. 13, No. 1(January),1988, pp. 58-67.

[CAM95] Campos, Rodrigo A., *Um Sistema Operacional Fundamentado no Modelo Cliente-Servidor e um Simulador Multiprogramado de Multicomputador*. Dissertação de Mestrado, CPGCC-UFSC, Maio, 1995.

[CAN92] Canfora, G.; Cimitile, A. and Carline, U. De, "On the Specification and Modelling of Discrete Control Systems". In: *Robotics and Flexible Manufacturing Systems*. Elsevier Science Publishers, 1992.

[CAR94] Carver, Doris L., "Integrated Modeling of Distributed Object-Oriented Systems". Journal of Systems Software, 1994, 24:233-244.

[CHA91] Chaar, J.K. et alli, "An Integrated Approach to Developing Manufacturing Control". Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, California (April),1991.

[CHA93a] Chaar, J.K et alli, "Developing Manufacturing Control Software: A Survey and Critique". The International Journal of Flexible Manufacturing Systems, 5(1993), pp. 53-88.

[CHA93b] Chaar, J.K et alli, "Real-Time Software Methodologies: Are They Suitable for Developing Manufacturing Control Software?". The International Journal of Flexible Manufacturing Systems, 5(1993), pp. 95-128.

[CHA93c] Chaar, J.K. et alli, "Developing Control Software for Efficient and Dependable Manufacturing Systems". In: *Intelligent Manufacturing : programming environments for CIM*. Springer-Verlag, 1993.

[CHA95] Changchien, S.W. et alli, "A Dynamic Control Model of Flexible Manufacturing Cells Using the Information Processing Object Hierarchy".International Journal FAIM, Vol. To appear, No., 1995.

[CHA93] Chapurlat, V.; Monneret, G. and Prunet, F., "Discrete Events System Modelling and Software Engineering: ACSY". 1993 CompEuro Proceedings. Computers in Design, Manufacturing, and Production, Prys-Evry (May), 1993, pp. 265-273.

[CHE94] Chen, C.; Lee S. and Santamarina G., "An object-oriented manufacturing control system". Journal of Intelligent Manufacturing, No. 5, 1994, pp. 315-321.

[CHO93] Cho, Hyuenbo, *An Intelligent Controller for Computer Integrated Manufacturing*. P.H.D. Thesis, Texas A&M University, December 1993.

[COR93] Corso, Thadeu B. *Ambiente para Programação Paralela em Multicomputador*. Relatório Técnico, UFSC-CTC-INE, Florianópolis, Novembro, 1993.

[DEM78] DeMarco, T., *Structured Analysis and System Specification*. Prentice-Hall, 1978.

[DES94] Desrochers, A.A. and Al-Jaar, R.Y., *Applications of Petri nets in automated manufacturing systems: modeling, control, and performance analysis*. Piscataway, NJ. IEEE Press, 1994.

[DIL91] Dilts, D. M.; Boyd, N. P. and Whorms, H.H., "The Evolution of Control Architectures for Automated Manufacturing Systems". Journal of Manufacturing Systems, Vol. 10, No. 1, 1991, pp. 79-93.

[DIC86] DiCesare, Frank. et alli, "Functions of a Manufacturing Workstation Controller". IEEE Conference, 1986.

[DOV88] Dove, R.K., "Process Design Automation: The Key to the 90's". The Proceedings of AUTOFACT'88 Conference, Chicago, Illinois (October), 1988.

[DUA93] Duan, Niu, *Extended Moore Machine Network Model for Discrete Event Control of Flexible Manufacturing Systems*. P.H.D. Thesis, The Pennsylvania State University, December 1993.

[DUF91] Duffie, Neil A., "Non-hierarchical cell control". In: *Manufacturing Cells: Control, Programming and Integration...*

[DUF88] Duffie, N. A.; Chitturi, R. and Mou, J., "Faut-tolerant Heterarchical Control of Heterogeneous Manufacturing System Entities". Journal of Manufacturing Systems, Vol.7, No.4, 1988, pp. 315-328.

[EZP92] Ezpeleta, J. and Martinez, J., "Petri Nets as a Specification Language for Manufacturing Systems". In: *Robotics and Flexible Manufacturing Systems*, Elsevier Science Publishers B. V.(North-Holland), 1992, pp. 427-436.

[FAB92] Fabian, M. and Lennartson B., "Control of Manufacturing Systems: An Object Oriented Approach". Proceedings of the 7th IFAC Symposium on Information Control Problems in Manufacturing Technology (INCOM 92), (Ed.: M. B. Zaremba), Pergamon Press, Oxford, England, 1992, pp. 47-52.

[FIS89] Fisher, J.P., "Zone Logic - Increased Machine Productivity through Artificial Intelligence". The Proceedings of the 18th Annual International Programable Controllers (IPC) Conference (April),1989.

[FIS89] Fisher,Thomas G., "Cell Controllers, MAP Communications and Sequential Function Chart Programming". ISA Transactions, Vol. 28, No. 3, 1989.

[FRI91] Friedrich, Luis F., "Modelagem de Sistemas de Manufatura : Uma Abordagem Orientada a Objetos". Conferência Latino Americana de Informática. Anais. Caracas, Venezuela, pp. 971-990, 1991.

[FU 82] Fu, K. S., *Syntatic Pattern Recognition and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1982.

[GAN82] Gane, C. and Sarson, T. *Structured Systems Analysis: Tools and Techniques*. Prentice Hall Publications, 1982.

[GIL93] Gilbert, J. W. and Wilhelm, R. G., "A concurrent object model for an industrial process-control application". Journal of Object-Oriented Programming, Vol.6, Nov-Dec, 1993.

[GIU93] Giua, A. and DiCesare, F., "GRAFCET and Petri Nets in Manufacturing". In: *Intelligent Manufacturing: Programming Environments for CIM*, Springer-Verlag, 1993, pp. 153-176.

[GLA89] Glassey, R.C. & Adiga, S., "Conceptual Design of a Software Object Library for Simulation of Semiconductor Manufacturing Systems". Journal of Object-Oriented Programming, SIGS, New York, Nov/Dec 1989, p.39-43.

[GRE86] Green, R. G., "Flexible Manufacturing Systems: Are They in Your Future?". In: *Tooling and Production*, 1986, pp. 35-38.

[GRO90] Groover, Mikell P., *Automation, Production Systems, and Computer Integrated Manufacturing*. Prentice Hall, Englewood Cliffs, 1990.

[HAR85] Harel, D. and Pneuli, A., "On the Development of Reactive Systems". Apt. K.R., (ed.), *Logics and Models of Concurrent Systems*, NATO ASI Series, Vol. F13(January), 1985,pp. 477-498.

[HAR88a] Harel, D. et alli, "STATEMATE: A Working Environment for the Development of Complex Reactive Systems". Proceedings of the 10th IEEE International Conference on Software Engineering, , Singapore (April), 1988,pp.396-406.

[HAR88b] Harel, D., "On Visual Formalisms". Communications of the ACM, Vol. 31, No. 5(May), 1988,pp.514-530.

[HEU88] Heuser, C.A., *Modelagem de Sistemas com Redes de Petri*. CPGCC,Porto Alegre, Maio, 1988.

[HOB92] Hoberecht, Walter C.;Joshi, Sanjay B. and Smith, Jeffrey S., " Architectures for Factory Control". Proceedings of Autofact '92, Detroit MI, 1992.

[ICA81] ICAM modelling Manual, IDEF0, IDEF1, IDEF2, ICAM Programme Library, AFWAR/MLTC, Wright Patterson Air Force Base, Ohio, 1981.

[ICO92] ICOOMS'92, International Conference on Object-Oriented Manufacturing Systems. Proceedings. Calgary, Alberta, Canada, 1992.

[JAS92a] Jasany, Leslie C., "Step Up to High-Level PLC Programming". Control & Systems, March, 1992.

[JAS92b] Jasany, Leslie C., "Today's Software: No Programming Required". Control & Systems, August, 1992.

[JOH89] Johansson, M.,"Communications Issues in Manufacturing". CIM Review, Vol. 6, No. 1, 1989.

[JON88] Jones, Vicent C., *MAP/TOP Networking - Achieving Computer Integrated Manufacturing*. McGraw Hill, New York, 1988.

[JOS90] Joshi, S. B.; Wysk, R. A. and Jones, A., "A Scaleable Architecture for CIM Shop Floor Control". Proceedings of CIMCOM'90, A. Jones Ed., National Instiute of Standards and Technology, May 1990, pp.21-33.

[JOS92] Joshi, S. B.; Mettala, E. G. and Wysk, R. A., "CIMGEN- A Computer Aided Software Engineering Tool for Development of FMS Control Software". IIE Transactions, Vol. 24, No. 3(July), 1992, pp. 84-97.

[KAV92] Kavi, M. K. and Yang, S., "Real-Time Systems Design Methodologies: An Introduction and a Survey". The Journal of Systems and Software, April 1992, pp. 85-99.

[KIM93] Kim, C.; Kim, K and Choi, I., "An Object-Oriented Information Modeling Methodology for Manufacturing Information Systems". Computers Industries Engineering, Vol. 24, No. 3, pp. 337-353, 1993.

[KIN90] King, Roger C., *An Integrated Approach to the Modelling, Design and Control of Industrial Systems*. PhD Thesis, The School of Electrical, Eletronics and Systems Engineering, University of Wales College of Cardiff, 1990.

[KOM87] Kompass, E.J., "Distributed Machine Control Uses Zoned Logic, Isolated Controllers, Fiber Optics". Control Engineering (August),1987.

[LAD90] Laduzinsky, Alan J., "Cell Controlers Proffer New Vistas for Plant Operations". Control Engineering, July, 1990.

[LAR89] Larin, David J., "Cell Control - What We Have, What We'll Need".Manufacturing Engineering, January, 1989.

[LED90] Lederhofer, Artur and Schwarz, Karlheinz, "Open Systems Protocols and Specifications - CNMA Technical Overview". Communications for Manufacturing - Proceedings of the Open Communications Congress, Stuttgart, Germany, 1990.

[LEV94] Levenson, N. G. et alli, "Requirements Specification for Process-Control Systems". IEEE Transactions on Software Engineering, Vol. 20, No. 9(September), 1994, pp. 684-706.

[LIN94] Lin, James Quo-Ping, *A Concurrent Design Environment for Flexible and Reusable Cell Control Software*. PhD Thesis, University of Missouri - Rolla, 1994.

[LIN94] Lin, L.; Wakabayashi, M. and Adiga, S., "Object-Oriented Modeling and Implementation of Control Software for a Robotic Flexible Manufacturing Cell". Robotics & Computer-Integrated Manufacturing, Vol.11, No.1, 1994, pp. 1-12.

[LHO93] Lhoste, P.; Iung, B.; Mayer, F.; Morel, G., "Reference Modelling for Distributed Intelligent Control System". Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. Vol. 3, 1993, pp.84-89.

[MAC92] Mackiewicz, Ralph E., "Applying the Manufacturing Message Specification". SME Workshop, Southfield MI, June 1992.

[MAN92] Man, Richard F., "Subsumption architecture-based real-time multitasking kernel for programming autonomous robots", In: Proceedings of SPIE-The International Society for Optical Engineering , Vol. 1829, 1992, pp.284-295.

[MAR89] Martin, John M., "Cells Drive Manufacturing Strategy". Manufacturing Engineering, January, 1989.

[McF89] McFadden, F.R., "Object-Oriented Techniques in Computer Integrated Manufacturing". Proceedings of the International Conference on System Sciences, 22,Alawaii,1989.

[McL86] McLean, C. and Jones, A., "A proposed hierarquical control model for automated manufacturing systems". Journal of Manufacturing Systems, Vol. 5, No. 1, 1986.

[MEY89] Meyer, B., *Object-Oriented Software Construction*. Prentice-Hall, Englewood Cliffs NJ, 1989.

[MEY86] Meyer, W., "Knowledge-based realtime supervision in CIM - the workcell controller". ESPRIT 86: results and achievements, 1986.

[MES90] Messina, Gaetano and Tricomi, Guido., "Manufacturing Communications Architectures". Computers in Industry, No. 13, 1990.

[MON95] Montez, Carlos B., *Um Sistema Operacional com Micronúcleo Distribuído e um Simulador Multiprogramado de Multicomputador*. Dissertação de Mestrado, CPGCC-UFSC, Maio, 1995.

[MUR89] Murata, T., "Petri Nets: Properties, Analysis and Applications". Proceedings of the IEEE, Vol. 77, No. 4, 1989, pp. 541-580.

[NAG92] Nagao, Y. et alli, "Petri net based programming system for FMS". In: *Robotics and Flexible Manufacturing Systems*, Elsevier Science Publishers B.V., 1992, pp. 295-304.

[NAY87] Naylor, Arch W. and Volz, Richard A., "Design of Integrated Manufacturing System Control Software". IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-17, No. 6, November/December, pp. 881-897.

[NOF94] Nof, S. Y., "Critiquing the potencial of object orientation in manufacturing". Int. J. Computer Integrated Manufacturing, Vol. 7, No. 1, pp.3-16, 1994.

[O'GR86] O'Grady, P.J., *Controlling Automated Manufacturing Systems*. Kogan Page Ltd, 1986. (TS155.8 034 1986)

[PAL87] Palframann, D., *FMS: Too Much, Too Soon, Manufacturing Engineering*, 1987, pp. 34-38.

[PAS86] Pascoe, G.A., " Elements of Object-Oriented Programming". BYTE , Aug. 1986, p.139-144.

[PET81] Peterson, J.L., *Petri net; theory and modelling of systems*. Englewood Cliffs, Prentice-Hall, 1981. 287p.

[PET62] Petri, C.A., *Kommunikation mit Automaten*. Bonn Institut fur instrumentelle Mathematik, 1962.

[PIR94] Pirklbauer, K.; Plosch, R. and Weinreich, R., "Object-oriented process control software". Journal of Object-Oriented Programming, Vol. 7, No. 2(May), 1994.

[PLE94] Pleinevaux, P., "An Analysis of the MMS Object Model". IEEE Transactions on Industrial Eletronics, Vol. 41, No. 3(June), 1994, pp. 265-268.

[QUA86] Quatse, Jesse T., "Requirements for Real-Time Cell Control".Proceedings of the 15th Annual Programmable Controllers Conference, 1986.

[RAN88] Rana, S. P. and Taneja, S. K., "A Distributed Architecture for Automated Manufacturing". International Journal of Advanced Manufacturing Technology, Vol. 3, No. 5, 1988.

[ROG91] Rogers, Paul, "Representation of the cell control task". In: *Manufacturing Cells: Control, Programming and Integration*, Butterworth-Heinemann Ltd, 1991, pp. 173-194.

[ROZ86] Rozenberg, G., *Advances in Petri nets 1985*. Berlin, Springer Verlag, 1986. 498p. (Lectures notes in computer science, 266).

[ROZ87] Rozenberg, G., *Advances in Petri nets 1987*. Berlin, Springer-Verlag, 1987. 498p. (Lectures notes in computer science, 266).

[SAH92] Saharaouri, A.E.K. and Ould-Kaddour, N.,Control software prototyping".Computers in Industry, Vol. 20, No. 3(October), 1992, pp. 327-334.

[SHA84] Shaw, M., "Abstraction techniques in modern programming languages". IEEE Software, Oct 1984, p.10-26.

[SCH84] Scharbach, P.,"Formal Methods and the Specification and Design of Computer Integrated Manufacturing Systems". Proceedings of the International Conference on the Development of Flexible Automation Systems, 1984.

[SMI92] Smith, Jeffrey S., *A Formal Design and Development Methodology for Shop Floor Control in Computer Integrated Manufacturing*. PhD Thesis, The Pennsylvania State University, December 1992.

[TAK89] Takahashi, T., *Introdução a Programação Orientada por Objeto*.VIII Jornada de Atualização em Informática. Uberlândia, 1989.

[THO89] Thomas, D., "What's in an Object?". Byte. March, 1989, pp.231-240.

[TOS93] Toscani, Simão S., *RS : Uma Linguagem para Programação de Núcleos Reactivos*. Tese de Doutorado, Universidade Nova de Lisboa, Lisboa, 1993.

[VAL87] Valette, R., "Nets in Production Systems". In: *Advanced Course on Petri nets*. Bad Honnef, Sep. 1987, p.191-217, Springer-Verlag.

[VAL90] Valette, R. & Silva, M., "A Rede de Petri: uma Ferramenta para a Automação Fabril". CONAI, 1990.

[VAL92] Valenzano A. et alli, *MAP and TOP Communications : Standards and Applications*. Addison-Wesley, 1992.

[VIL93] Villarroel, J.L. et alli, "KRON: An Approach for the Integration of Petri Nets in Object Oriented Models of Discrete Event Systems". Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol.3, 1993,pp.268-273.

[VIL94] Villarroel, J.L. and Muro-Medrano, P.R., "Using Petri Net Models at the Coordination Level for Manufacturing Systems Control". Robotics & Computer-Integrated Manufacturing, Vol. 11, No. 1, 1994,pp. 41-50.

[VIT94] Vitale, Michael J., *A Formal Methodology for Constructing Decoupled Manufacturing System Control Software Components*. PhD Thesis, University of Michigam, 1994.

[WIL88] Wilczynski, D., "A Common Device Control Architecture - The Savoir Actor".The Proceedings of the AUTOFACT'88 Conference, Chicago, Illinois (October),1988.

[WIL93] Wilczynski, D. and Wallace, D.K., "OOPS in Real-Time Control Applications". In: *Object-oriented Software for Manufacturing Systems*. Edited by S. Adiga, Chapman and Hall, 1993.

[WIL88] Willians, David J., *Manufacturing Systems - An Introduction to the Technologies*. Halstead Press, New York, NY, 1988.

[YON88] Yonezawa, A. & Tokoro, M., *Object-Oriented Concurrent Programming*. Computer Systems Series. The MIT Press.

[YOU88] Yourdon, E., *Managing the System Life Cycle*. Englewood Cliffs, NJ, Yourdon Press, 1988.

[ZHO91] Zhou, MengChu, "Combination of Petri Nets and Intelligent Decision Makers for Manufacturing Systems Control". Proceedings of the 1991 IEEE International Symposium on Intelligent Control, Arlington, Virginia - USA (August), 1991, pp. 146-151.