UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ESTUDO DA UTILIZAÇÃO DO PADRÃO MMS EM APLICAÇÕES FABRIS

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA PARA OBTENÇÃO DE GRAU DE MESTRE EM ENGENHARIA ELÉTRICA

JONY LAUREANO SILVEIRA

Florianópolis, 20 de Dezembro de 1991

ESTUDO DA UTILIZAÇÃO DO PADRÃO MMS
EM APLICAÇÕES FABRIS

JONY LAUREANO SILVEIRA

ESTA DISSERTAÇÃO FOI JULGADA PARA A OBTENÇÃO DO TÍTULO DE

MESTRE EM ENGENHARIA ELÉTRICA

ESPECIALIDADE ENGENHARIA ELÉTRICA, ÁREA DE CONCENTRAÇÃO SISTEMAS DE CONTROLE E AUTOMAÇÃO INDUSTRIAL, E APROVADA EM SUA FORMA FINAL PELO CURSO DE PÓS-GRADUAÇÃO

Prof. Jean-Marie Farines, Dr. Ing.
Orientador

Prof. João Pedro Assumpção Bastos, Dr. D'Etat
Coordenador do Curso de Pós-Graduação em Engenharia Elétrica

BANCA EXAMINADORA:

Prof. Jean-Marie Farines, Dr. Ing.
Orientador

Prof. Joni da Silva Fraga, Dr.
Co-Orientador

Prof. Vitório Bruno Mazzola, Dr.

Profa. Elizabeth Sueli Specialski, M.Sc.

Prof. Aloysio de Castro Pinto Pedrosa, Dr.

À Zeferino e Daura, meus pais.

AGRADECIMENTOS

Muitas pessoas colaboraram para a realização deste trabalho. Sou muito grato a todas, em especial:

Aos professores Jean-Marie Farines e Joni da Silva Fraga pela orientação e coorientação deste trabalho.

Aos membros da banca, pelos comentários e sugestões feitos a este trabalho.

À minha família, pela compreenção e apoio nas horas díficeis.

A todos os amigos, em especial aos do LCMI, pelo incentivo e paciência (Valeu, Galera!).

RESUMO

A presente dissertação tem por objetivo estudar a utilização dos serviços oferecidos na camada de aplicação pelo padrão MAP (*Manufacturing Automation Protocol*) para especificar as mensagens de manufatura (protocolo MMS). Este protocolo tem a função de dar suporte à transmissão de mensagens entre equipamentos inteligentes nos ambientes de manufatura e controle de processos.

Com este intuito, propõe-se neste trabalho uma metodologia, destinada a auxiliar o implementador na utilização do Padrão MMS em aplicações da área de automação fabril.

Uma ferramenta destinada a simulação de sistemas de comunicação baseados no Padrão MMS é também proposta no sentido de dar apoio à metodologia apresentada. Esta ferramenta utiliza uma biblioteca de módulos Estelle configuráveis que permitem construir, de forma automática, a especificação do sistema de comunicação usando os Serviços MMS em vista da simulação.

Com a finalidade de exemplificar e testar a metodologia e a ferramenta propostas, são apresentadas as várias etapas desta metodologia para uso do Padrão MMS no sistema de comunicação de uma Célula Flexível de Usinagem. Destaca-se em particular a especificação formal em Estelle dos elementos que formam o sistema de comunicação desta aplicação específica.

A simulação desta especificação Estelle permitiu mostrar o interesse do uso desta na escolha dos Serviços MMS destinados a comunicação na célula flexível, bem como no modelamento de seus dispositivos através dos Objetos MMS.

ABSTRACT

This work is devoted to studying the utilization of the services provided by the application layer of the MAP (Manufacturing Automation Protocol) standard for specification of manufacturing messages (MMS protocol). This protocol has functionalities to support message transmission among intelligent equipments in manufacturing and process control environments.

Within this scope, a methodology for helping the implementor in using the MMS standard in industrial automation applications is proposed.

Also, it is proposed a tool for communication systems simulation based on the MMS standard, which gives support to the suggested methodology. This tool uses a library of configurable Estelle modules that allows one to automatically build the communication system specification using the MMS services regarding to the simulation.

As a mean to illustrate and test both the methodology and the tool proposed, various steps of this methodology for using the MMS standard in communication systems within a Flexible Manufacturing Cell are presented. Some particular attention is pay to the formal specification of the elements which form the communication system for this specific application.

Simulation of the Estelle specification has shown the interest of using it for choosing the MMS services for communication in the Flexible Cell, as well as in modeling its devices through MMS objects.

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

A realidade mercadológica atual vem sendo marcada por uma diversificação cada vez maior e mais rápida das necessidades dos clientes. Este comportamento tem forçado as empresas manufatureiras a lançarem produtos com ciclos de vida mais curtos, custos industriais reduzidos, alta tecnologia e fabricação em pequenos lotes.

Para atingir estes resultados, é necessário garantir a integração das informações provenientes de vários setores de uma empresa, segundo princípios conhecidos como CIM (*Computer Integrated Manufacturing*).

A comunicação entre os vários equipamentos heterogêneos dentro de uma empresa é a parte deste problema que nos preocupa neste trabalho.

Para suprir os requisitos de comunicação de uma abordagem CIM e ao mesmo tempo garantir a compatibilidade de todos os equipamentos e sistemas da empresa (independente de fabricantes ou tecnologias), é necessária a padronização do suporte de comunicação. Os padrões MAP [MAP 88] e TOP [TOP 88] são aceitos hoje como base para sistemas de comunicação na área fabril.

O objetivo deste trabalho é estudar a utilização dos serviços oferecidos na camada de aplicação pelo padrão MAP (*Manufacturing Automation Protocol*) para especificar as mensagens de manufatura (protocolo MMS). Este protocolo tem a função de dar suporte à transmissão de mensagens entre equipamentos inteligentes nos ambientes de manufatura e controle de processos.

No capítulo 2 são apresentados conceitos gerais acerca dos Sistemas de Manufatura Flexível, e introduzidos alguns conceitos básicos necessários à compreensão do Padrão MMS.

O capítulo 3 propõe uma metodologia para a utilização do Padrão MMS em aplicações da área de automação fabril e ainda uma ferramenta de apoio a esta metodologia que será destinada a simulação dos Sistemas de Comunicação baseados no Padrão MMS.

Com a finalidade de exemplificar a utilização da metodologia e da ferramenta propostas, o capítulo 4 apresenta o uso do Padrão MMS no sistema de comunicação de uma Célula Flexível de Usinagem. A especificação formal e a simulação dos elementos que formam o sistema de comunicação, decorrente do uso do Padrão MMS na célula flexível, serão ainda mostrados neste capítulo.

CAPÍTULO 2

OS SISTEMAS DE MANUFATURA E O PADRÃO DE COMUNICAÇÃO MMS

2.1 Introdução

O constante progresso tecnológico é fator decisivo no processo de maximização da produção, e de minimização dos custos desta. Os Sistemas Flexíveis de Manufatura (FMS) são elementos fundamentais deste processo.

Os Sistemas Flexíveis de Manufatura, geralmente formados por máquinas e equipamentos ligados entre sí por um sistema de transporte, permitem a realização de operações diversificadas sobre peças diferentes, sem que haja necessidade de interromper o processo de fabricação para adaptação das instalações.

Neste capítulo, apresenta-se algumas características dos Sistemas Flexíveis de Manufatura e aborda-se as questões relacionadas aos sistemas de comunicação. Estudar-se-á, mais particularmente, o Padrão de Especificação de Mensagem da Manufatura (MMS).

2.2 Os Sistemas Flexíveis de Manufatura

O controle de um FMS é realizado em geral através de uma rede hierarquizada de computadores, composta por um computador que atua como supervisor de processos e dos computadores que atuam como remotos. Esta estrutura pode ser vista na figura 2.1.

O caminho mais empregado na formação de um FMS é a utilização de grupos individuais chamados de Células Flexíveis. Cada uma destas células é responsável pela realização de uma operação sobre os elementos do produto a ser fabricado. Um computador Supervisor é responsável pelo controle e interligação do conjunto de células flexíveis, formando assim o sistema global denominado FMS.

Diversas estruturas automatizadas presentes de maneira desvinculada em vários setores de uma empresa, tais como projeto e engenharia (CAD/CAE), planejamento de

processos (CAPP), Controle de produção (CAM) e o próprio processo de produção (FMS) podem ser integradas usando o conceito de Manufatura Integrada por Computador (CIM).

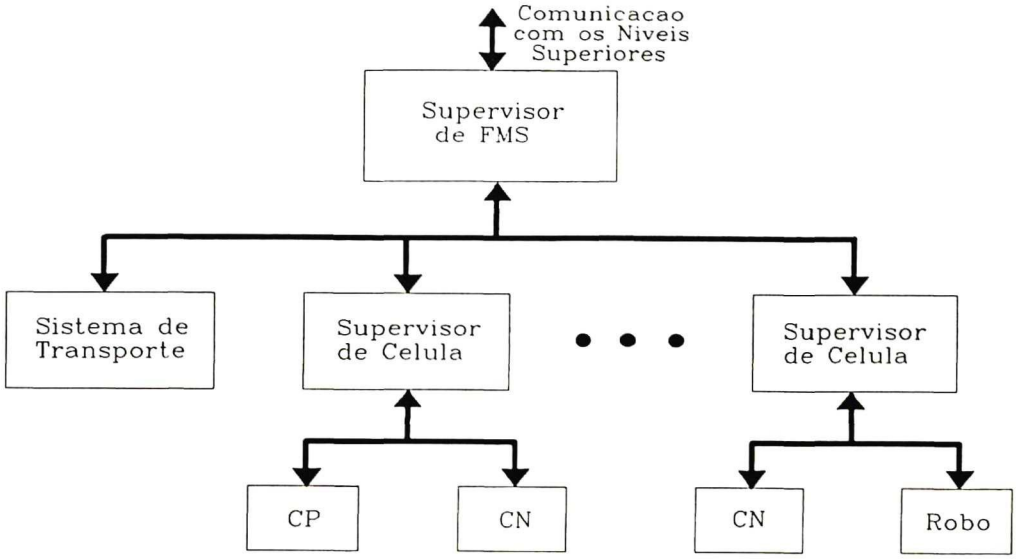


Fig. 2.1 Estrutura de Controle de um FMS

A decomposição funcional numa abordagem CIM requer uma estrutura organizacional hierárquica, abrangendo todos os setores da empresa. O modelo sugerido pela ISO (*International Standard Organization*) para automação fabril, divide a empresa segundo seis níveis hierárquicos (figura 2.2).

Para cada nível desta hierarquia, o modelo atribui uma lista de funções básicas, o que implica na existência de uma estrutura computacional que satisfaça a hierarquia organizacional e acomode a separação geográfica das entidades computacionais. A interligação destes vários níveis torna necessário o uso de soluções distribuídas tanto em termos de sistema de comunicação, quanto de base de dados.

2.3 O Sistema de Comunicação nos Sistemas de Manufatura

Redes Locais de comunicação são o meio que permite esta troca de informações entre as diversas entidades computacionais de um ambiente CIM. Para suprir os requisitos de comunicação do CIM, garantindo a compatibilidade entre os vários equipamentos e sistemas da empresa, a rede de comunicação deve seguir a padronização internacional.

Nivel	Hierarquia	Funcao Basica
6	Empresa	Gerenciamento da corporacao como um todo.
5	Instalacao	Projeto, engenharia e planejamento da producao
4	Secao	Alocacao de material e recursos para a producao
3	Celula	Coordenacao a nivel de celula
2	Estacao	Coordenacao a nivel de maquina
1	Equipamento	Coordenacao a nivel de sensores e atuadores

Fig. 2.2 Modelo ISO para Automação Fabril

2.3.1 Os Padrões de Comunicação nos Sistemas de Manufatura

Os esforços dispensados no sentido de chegar a uma padronização consistente para uso em ambientes fabris, baseiam-se no modelo de referência de sete camadas destinado a interconexão de sistemas abertos (RM-OSI) [ISO 84].

Os requisitos de comunicação, no entanto, não são os mesmos em cada nível da hierarquia organizacional. Os níveis mais baixos são caracterizados por mensagens curtas, frequentes e transmitidas sobre distâncias curtas (tais como atribuições de controle, medidas e alarmes). As mensagens nos níveis mais altos incluem normalmente transferência de grandes arquivos e interação homem-máquina. A figura 2.3 mostra esta diversidade de características ao longo dos vários níveis da hierarquia.

Evidentemente um único padrão de comunicação não seria capaz de suprir este conjunto de requisitos ambíguos em toda a hierarquia organizacional da empresa. Os padrões MAP (*Manufacturing Automation Protocols*) e TOP (*Technical and Office Protocols*) [MAP 88 / TOP 88] foram propostos para suprir as necessidades em termos de integração dos vários sistemas computacionais dentro de uma empresa.

O Padrão MAP consiste de uma seleção, em cada camada do modelo RM-OSI, dos protocolos mais apropriados para aplicações nos níveis inferiores da hierarquia do CIM. O

TOP, também baseado no modelo RM-OSI, destina-se aos departamentos de engenharia, contabilidade, *marketing* e outros.

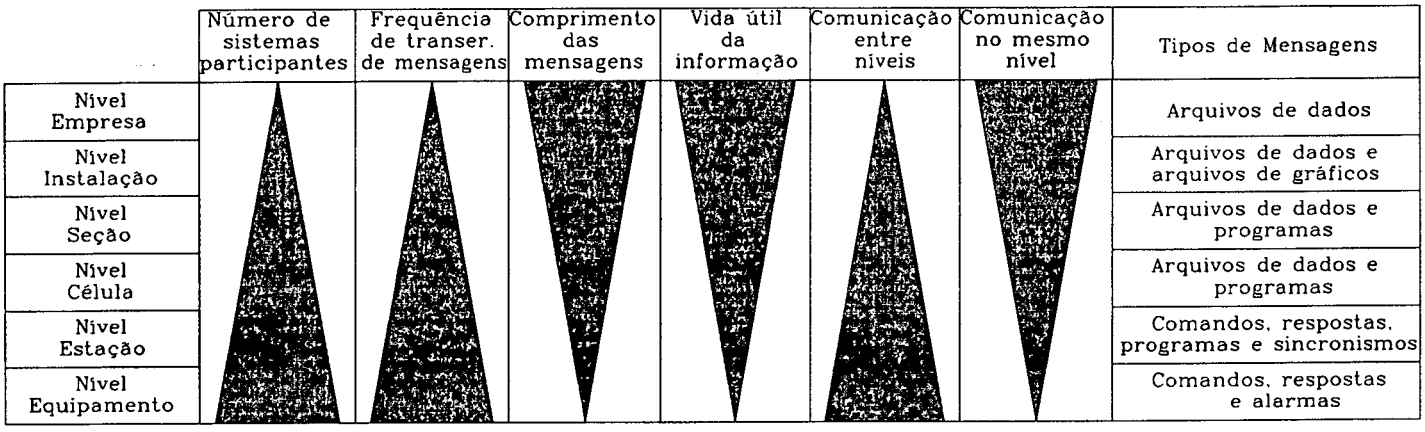


Fig. 2.3 Características de Comunicação [Leite 87]

Com a evolução da especificação do MAP, usuários de Controles de Processos sentiram a necessidade de tempos de resposta mais rápidos em relação à troca de mensagem no caso de aplicações em tempo real. Assim através do sacrifício de algumas funcionalidades do protocolo MAP, surgiram as arquiteturas MAP/EPA e MINI-MAP [Pinho 89], com intuito de diminuir o *overhead* ocasionado pelo número excessivo de camadas do modelo de referência OSI.

A arquitetura Mini-MAP é formada apenas pelas camadas física, enlace e aplicação. Por não apresentarem todas as camadas, os nós MINI-MAP devem ser conectados às redes MAP através de estruturas intermediárias denominadas *gateways*. O fato de exigirem estruturas adicionais encarece seu uso, por isso, nós Mini-MAP só devem ser usados para atender requisitos temporais restritivos.

Os nós MAP/EPA apresentam a desvantagem de apresentar custos mais elevados, devido ao fato de comportarem uma dupla pilha de protocolos (uma pillha MAP e outra Mini-MAP), porém podem ser aproveitados como *gateways* para nós do tipo Mini-MAP.

2.3.2 A Camada de Aplicação: Definições e Conceitos

O RM-OSI (*Reference Model for Open Systems Interconnection*) é um padrão internacional cuja finalidade básica é servir como base para o desenvolvimento de protocolos-padrão para a interconexão de Sistemas Abertos Reais.

Um Sistema Aberto Real é definido como um sistema informatizado (*hardware* e *software*) capaz de se comunicar com outros sistemas através do Padrão RM-OSI.

O modelo é apresentado sob a forma de uma arquitetura estratificada, na qual são definidas sete camadas que decompõem o problema da interconexão entre sistemas abertos. Desta forma, cada camada é responsável por uma ou mais funções do processo de comunicação, e deve se preocupar apenas em usar os serviços da camada imediatamente inferior para desempenhar suas funções, sem levar em conta a maneira como estes serviços foram implementados.

Os aspectos abordados neste trabalho estão ligados à última camada deste modelo, conhecida como camada de Aplicação. Nesta camada estão os protocolos diretamente relacionados com a aplicação do usuário. Estes protocolos fornecem a única maneira do usuário acessar o sistema de comunicação OSI [Roisemberg 88]. Desta forma, a camada de aplicação deve prover todos os serviços diretamente utilizáveis pelos Processos de Aplicação do usuário.

Um Processo de Aplicação (AP) é um elemento lógico de um Sistema Aberto Real que realiza um determinado processamento de informação requerido por uma aplicação específica do usuário. Se a aplicação é distribuída, então cada uma das partes distribuídas forma um AP. É importante observar que a totalidade da aplicação do usuário não reside completamente nesta camada, mas apenas a parte que precisa se comunicar com entidades remotas do Sistema de Comunicação [Moura 86].

Do ponto de vista do RM-OSI, um AP é um elemento da classe definida por um Tipo-AP, responsável pela execução de uma ou mais tarefas de processamento de informações. A invocação, num certo instante, de uma operação deste AP, dará origem a uma instância específica, denominada Instância de Processo de Aplicação (AP-I). Desta forma, uma aplicação distribuída consiste em interações entre AP-Is.

Para permitir a interação entre duas AP-Is, estas devem compartilhar um Universo de Discurso, ou seja, eles devem ter características comuns de comunicação. O conjunto de funções de comunicação de um AP é representado no RM-OSI por uma Entidade de Aplicação (AE). Para tanto, uma AE deve comportar um ou mais Elementos de Serviço de Aplicação (ASE). Cada ASE é formado por um conjunto coerente de funções capazes de fornecer subsídios de comunicação para um propósito particular. Estes ASEs podem ser separados em duas classes:

- Elemento de Serviço de Controle de Associação (ACSE), que deve fornecer os serviços para estabelecimento, término ou aborto de uma associação lógica entre dois APs; e

- Elementos de Serviços de Aplicação Específicos (SASE), que engloba todos os ASEs dedicados a funções específicas como é o caso do Elemento de Serviço de Mensagem da Manufatura (MMSE ou apenas MMS como é mais referenciado).

Do ponto de vista do RM-OSI, uma AE é um elemento da classe definida por um Tipo-AE. As atividades de uma dada AE são suportadas por uma invocação desta AE, a qual dá origem a uma instância específica (AE-I), que provê suas funções quando de uma determinada comunicação.

Cada AE é identificável através de um ou mais Títulos-AE e deverá conhecer o Título-AE remoto, de modo que possa obter o endereço de apresentação correspondente à estação remota, através das funções de diretório. A AE, usa então este endereço para estabelecer uma conexão com a AE remota. Esta conexão, chamada de Associação de Aplicação (AA), estabelece um ambiente de comunicação negociado entre dois APs que necessitam se comunicar remotamente.

2.4 O Padrão MMS

O Padrão MMS (*Manufacturing Message Specification*), é especificado como um Protocolo genérico da camada de aplicação destinado à transmissão de mensagens no ambiente de Manufatura e Controle de Processos, permitindo o acesso a dados e a execução de controle em uma vasta gama de sistemas e equipamentos do chão de fábrica.

As organizações de padronização especializadas são responsáveis pela elaboração dos padrões contendo os detalhes e aspectos específicos de aplicação para cada equipamento (não cobertos pelo padrão genérico). Tais documentos são normalmente referidos na bibliografia como "*Companion Standards to* ISO-9506" (Padrões Associados ao MMS).

A especificação do padrão baseia-se numa representação baseada em objetos, onde os elementos são descritos através de objetos abstratos (**Objetos MMS**), contendo características externamente visíveis (**Atributos MMS**) e operações sobre estes objetos (**Serviços MMS**).

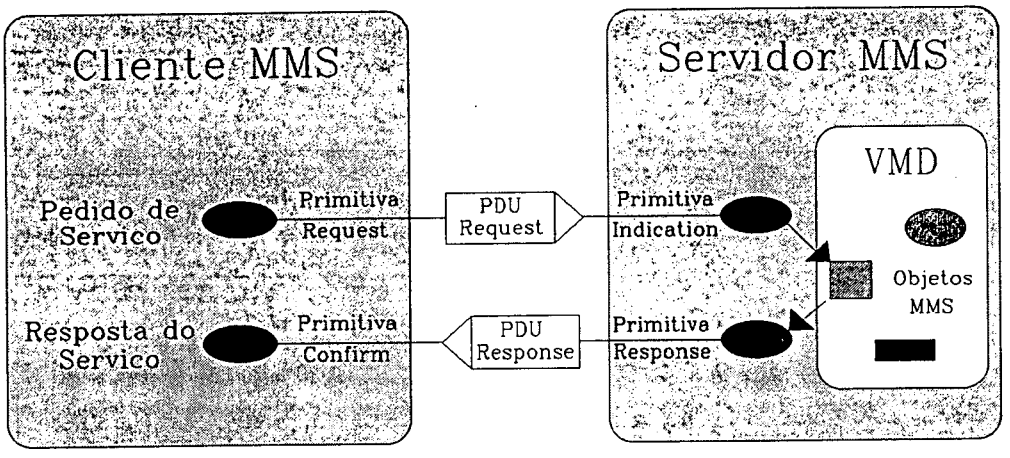


Fig. 2.4 Modelo Cliente/Servidor

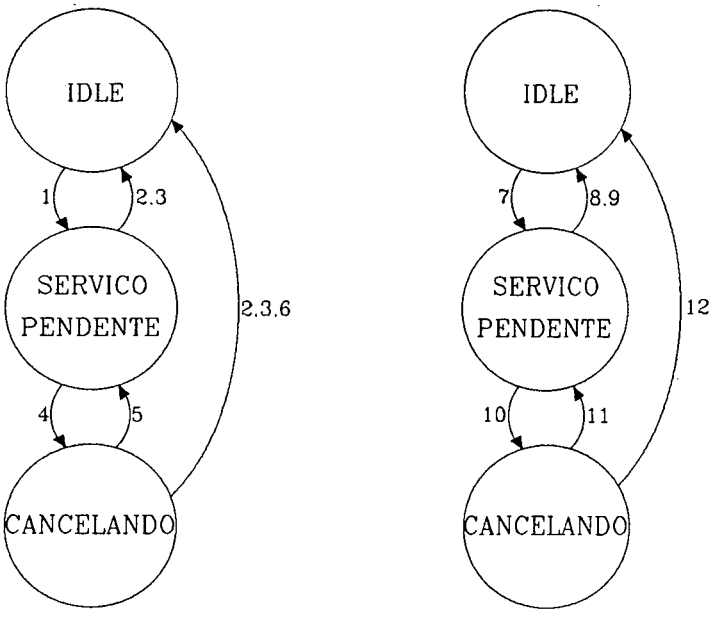
2.4.1 Serviços MMS

O Padrão MMS segue o modelo de interação **Cliente/Servidor**. Segundo este modelo, o processo de comunicação se dá quando um AP requisita uma operação particular a um AP remoto por meio de um pedido de serviços MMS. Neste caso, o AP que requisita o serviço é denominado AP Cliente e o que realiza a operação é denominado AP Servidor. A figura 2.4 apresenta o modelo Cliente/Servidor e as quatro primitivas básicas de um serviço.

Os serviços MMS podem ser confirmados ou não-confirmados. Os serviços não-confirmados são aqueles que não implicam na espera por resposta do usuário remoto (exigem apenas as primitivas *request* e *indication*). Os serviços confirmados caracterizam-se pela espera de uma resposta; para tanto, o provedor define uma máquina de protocolo para cada invocação de serviço (figura 2.5).

A fim de associar um determinado serviço pendente à uma máquina de protocolo, o provedor utiliza-se de um parâmetro contido nas primitivas e PDUs denominado *Invoke ID*. O *invoke ID* é único para todas as primitivas e PDUs de um mesmo pedido de serviço MMS em uma mesma associação.

Os serviços MMS são agrupados em 10 classes principais, que são listadas no anexo A. Estes serviços tem caráter genérico pois servem indistintamente para vários sistemas e equipamentos fabris. Cabe a cada um dos *Companion Standards* criar serviços específicos e particularizar aqueles definidos pelo padrão genérico.



TRANSICOES:

1. x.req/Confirmed_RequestPDU(x)
2. Confirmed_ResponsePDU/x.cnf (+)
3. Confirmed_ErrorPDU/x.cnf (-)
4. Cancel.req/Cancel_RequestPDU
5. Cancel_ErrorPDU/Cancel.cnf (-)
6. Cancel_ResponsePDU & Confirmed_ErrorPDU/Cancel.cnf (+) & x.cnf (-)
7. Confirmed_RequestPDU(x)/x.ind
8. x.rsp (+)/Confirmed_ResponsePDU(x)
9. x.rsp (-)/Confirmed_ErrorPDU(x)
10. Cancel.requestPDU/Cancel.ind
11. Cancel.rsp (-)/Cancel_ErrorPDU
12. Cancel.rsp (+) & x.rsp (-)/ Cancel_ResponsePDU & Confirmed_ErrorPDU(x)

Fig. 2.5 Máquina de Protocolo para Serviços Confirmados

2.4.2 Objetos MMS

O MMS define classes contendo um determinado número de objetos que por sua vez são caracterizadas através de atributos. Estes atributos descrevem as características externamente visíveis dos objetos de uma determinada classe.

Cada objeto MMS pode ser instanciado mais de uma vez, sendo possível modificar as características de cada uma destas instâncias através dos serviços MMS.

Cada objeto é identificado univocamente dentro do ambiente OSI através de um ou mais atributos chaves (*key attributes*). Alguns destes atributos tem sua existência dependente da verificação de certas condições expressas através do uso de *Constraints*.

Um exemplo de definição sintática genérica de um objeto MMS pode ser a seguinte:

Object: <nome da classe>
    Key Attribute: <nome do tipo-atributo> {VALORES}
    :
    :
    Key Attribute: <nome do tipo-atributo> {VALORES}
    Attribute: <nome do tipo-atributo> {VALORES}
    :
    :
    Attribute: <nome do tipo-atributo> {VALORES}
    Constraint: <expressão de restrição>
        Attribute: <nome do tipo-atributo> {VALORES}
        :
        :
        Attribute: <nome do tipo-atributo> {VALORES}

Alguns objetos contém atributos que fazem referência a outros objetos (atributos de referência), provendo um mecanismo de ligação entre dois objetos.

Um dos atributos mais comuns nos Objetos MMS é o atributo MMS Deletable, que indica se o objeto pode ou não ser "deletado" através de algum Serviço MMS. Os Objetos MMS ficam assim divididos em objetos Estáticos que são aqueles permanentemente presentes na representação MMS do dispositivo; e os objetos Dinâmicos, que devem ser carregados antes de serem utilizados.

O anexo B mostra os Objetos MMS usados neste trabalho, organizados segundo as classes descritas no Padrão MMS genérico.

2.4.3 Dispositivo Virtual de Manufatura (VMD)

Para que o AP Cliente possa acessar os objetos do AP Servidor por meio de um pedido de serviços MMS, é necessário que o AP servidor torne disponível, os recursos e funcionalidades ligados a este serviço MMS. Isto é feito através do conceito de Dispositivo Virtual de Manufatura (**VMD**).

O VMD (figura 2.6) é uma representação virtual das potencialidades físicas e funcionais encontradas em um equipamento fornecedor de serviços. Um AP Servidor MMS pode conter mais de um VMD, sendo cada um logicamente separado dos demais.

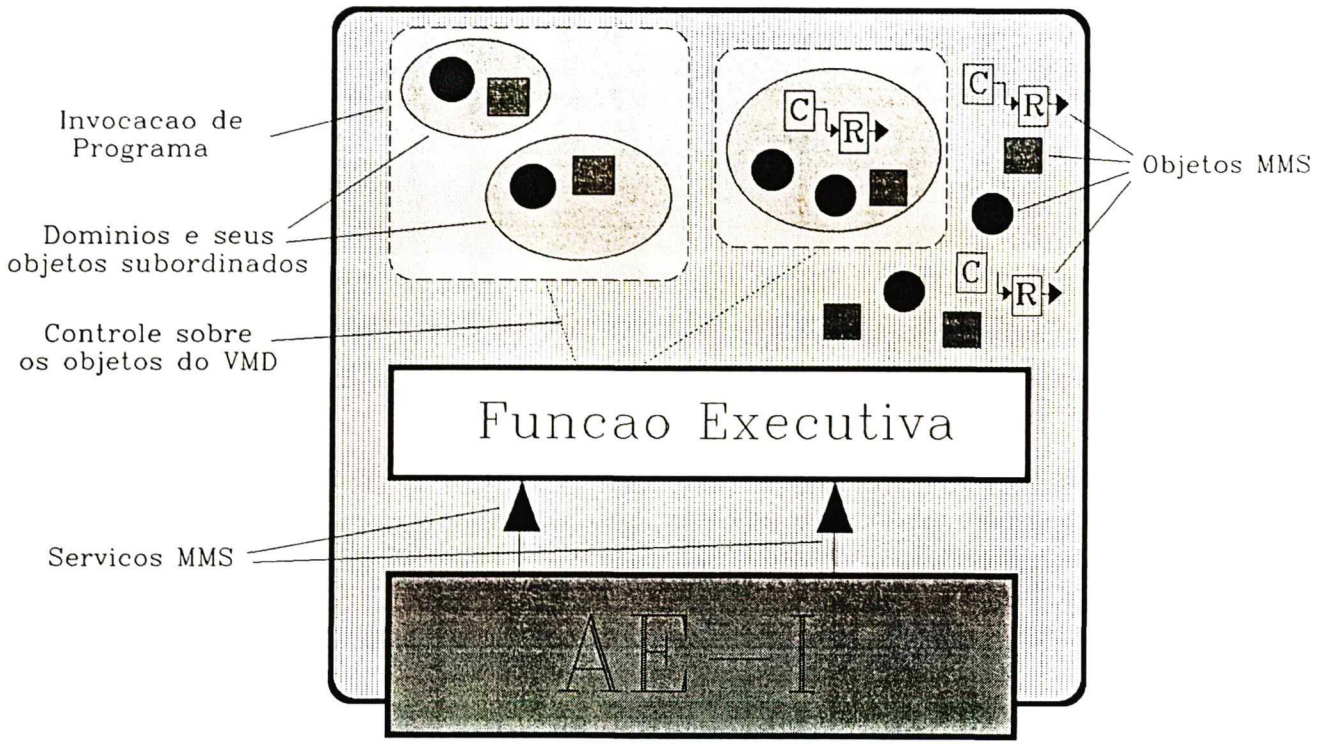


Fig. 2.6 Representação Gráfica de um VMD Genérico

Um VMD é composto de uma Função Executiva, que gerencia o acesso aos objetos definidos no VMD; zero, uma ou mais invocações de programa (cada uma das quais controlando um ou mais domínios) e outros objetos subordinados.

Para se comunicar, um VMD deve estar ligado a uma ou mais AEs. Neste caso, cada AE (o Padrão MMS restringe a ligação de uma dada AE a um único VMD) representa um conjunto das funções de comunicação do AP usadas pelo VMD.

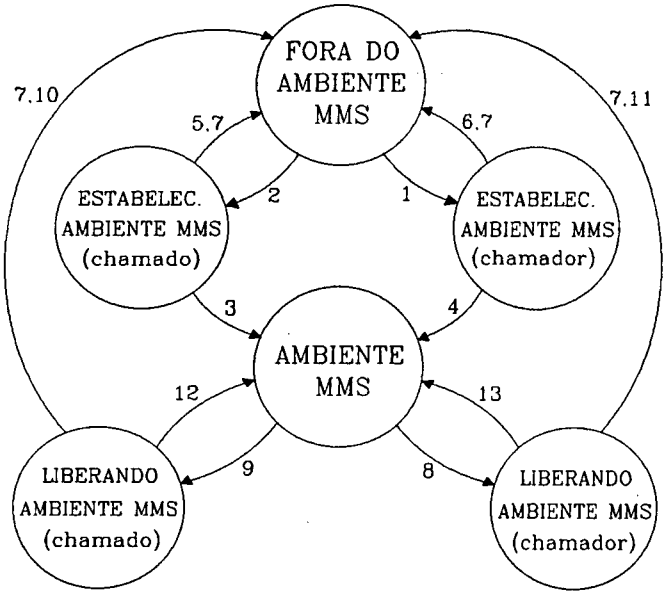
Um modelo para o AP Cliente não é fornecido pelo padrão, pois os serviços MMS não restringem seu comportamento, a não ser pela sequência válida das primitivas.

2.4.4 Ambientes MMS

Segundo o Padrão MMS, cada AP possui um conjunto de informações acerca das suas capacidades e limites de comunicação.

Para poderem trocar informações, dois APs deverão primeiramente negociar um contexto comum chamado Ambiente MMS que é formado por um sub-conjunto do universo de discurso de cada um dos APs.

Um usuário do sistema de comunicação pode estabelecer mais que um ambiente MMS, sendo cada um deles empregado na comunicação com um usuário MMS remoto diferente. Para cada ambiente é criado uma máquina de protocolo de contexto, como aquela mostrada na figura 2.7.



TRANSICOES:

1. Initiate.req
2. Initiate.ind
3. Initiate.rsp (+)
4. Initiate.cnf (+)
5. Initiate.rsp (-)
6. Initiate.cnf (-)
7. Abort.req (Abort.ind)
8. Conclude.req
9. Conclude.ind
10. Conclude.rsp (+)
11. Conclude.cnf (+)
12. Conclude.rsp (-)
13. Conclude.cnf (-)

Fig. 2.7 Máquina do Protocolo de Contexto

2.4.5 Provedor MMS

O Provedor MMS (**MMPM**) serve como intermediador nas interações entre um programa de aplicação e o sistema de comunicação. As principais funções do Provedor MMS em uma arquitetura MAP são [Leite 88]:

- Tratar as primitivas recebidas do usuário local mapeando-as em PDUs, a serem enviadas ao usuário remoto (em geral o provedor MMS repassa de forma transparente os parâmetros das PDUs);

- Tratar as PDUs corretas recebidas do usuário remoto, mapeando-as em primitivas a serem enviadas ao usuário local;

- Tratar as PDUs inválidas (o que caracteriza um erro de protocolo), notificando os usuários local e remoto via serviço *Reject*. Entende-se por PDU inválida as PDUs com erro estrutural ou com recebimento em estado incorreto;

- Tratar as primitivas recebidas do ACSE relativas ao contexto de aplicação;

2.5 Conceitos MMS Relativos à Arquitetura Mini-MAP

O interfaceamento direto entre as camadas de aplicação e enlace da arquitetura Mini-MAP, causa algumas restrições aos protocolos de Aplicação, na medida em que os serviços das camadas intermediárias não podem ser oferecidos à camada de Aplicação da mesma forma que na arquitetura MAP. Com relação ao Padrão MMS, as alterações mais significativas podem ser sentidas em termos das entidades de aplicação, tipos de associação, provedor de serviços e até mesmo na necessidade de serviços adicionais.

2.5.1 Entidades de Aplicação Mini-MAP

A primeira restrição decorrente da ausência das camadas intermediárias está relacionada com o endereçamento entre uma AE da camada de aplicação e um LSAP da camada de enlace. No caso de uma arquitetura Mini-MAP, todas as instâncias de uma AE deverão ser ligadas a um mesmo LSAP. A comunicação se torna possível com o conhecimento a priori do LSAP destino, via serviço de diretório, gerenciamento local ou simplesmente a partir de acordo prévio entre os programadores das aplicações em questão [MAP 88].

Os parâmetros de gerenciamento de conexão específicos de AE são:

- número de associações por invocação de AE (AE-I), que para o Padrão MMS é restrito a uma. Desta forma, os usuários cliente e servidor devem instanciar tantas AEs quantas forem as associações que devam estabelecer.

- número de invocações de AE por LSAP. Na arquitetura Mini-MAP cada par formado pelo endereço LLC (LSAP) e pelo endereço de MAC pode oferecer até

oito associações, cada uma identificada por um dos oito níveis de prioridade possíveis para mensagem (níveis que também são atribuídos à associação utilizada na transferência).

Uma segunda restrição está relacionada à ausência do protocolo ACSE, que leva a necessidade de pré-definição da sintaxe abstrata de todas as AEs. A figura 2.8 mostra o comportamento de uma Entidade de Aplicação dentro de uma arquitetura Mini-MAP.

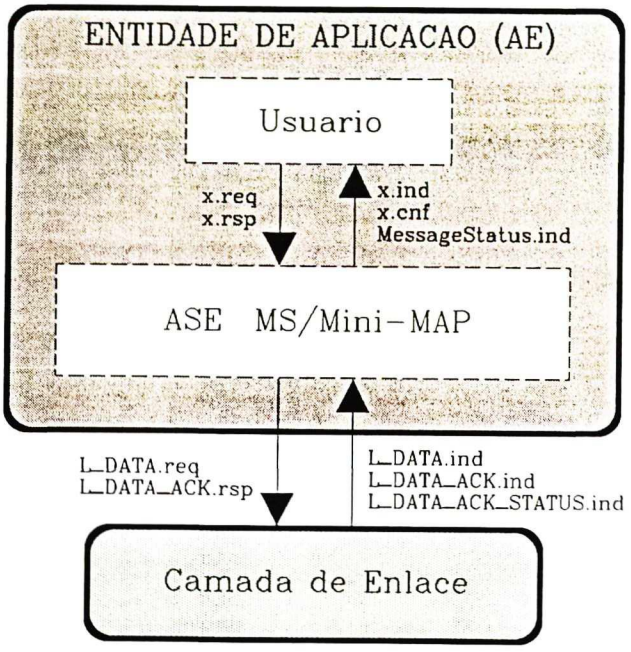


Fig. 2.8 ASE MMS na Arquitetura Mini-MAP [MAP 88]

2.5.2 Tipos de Associação no Mini-MAP

Com a inexistência das camadas de apresentação e sessão, os serviços de estabelecimento de uma AA não são disponíveis na arquitetura Mini-MAP. Surge então o conceito de Associação Explícita, permitindo que a máquina de protocolo MMS/Mini-MAP opere num ambiente similar àquele criado pelo ACSE. A figura 2.9 apresenta as primitivas usadas no estabelecimento de uma Associação Explícita.

A comunicação MMS pode ser feita ainda sem o estabelecimento de associação. Neste caso, as informações normalmente trocadas durante o estabelecimento da associação são substituídas por valores *default*, definidos por uma tarefa de gerenciamento. Este tipo de comunicação é mais rápido, porém, limita-se a alguns serviços MMS apenas.

Cada LSAP pode garantir apenas um tipo de comunicação, assim, deve-se definir a priori, quais LSAPs utilizarão comunicação sem associação e quais utilizarão associação explícita.

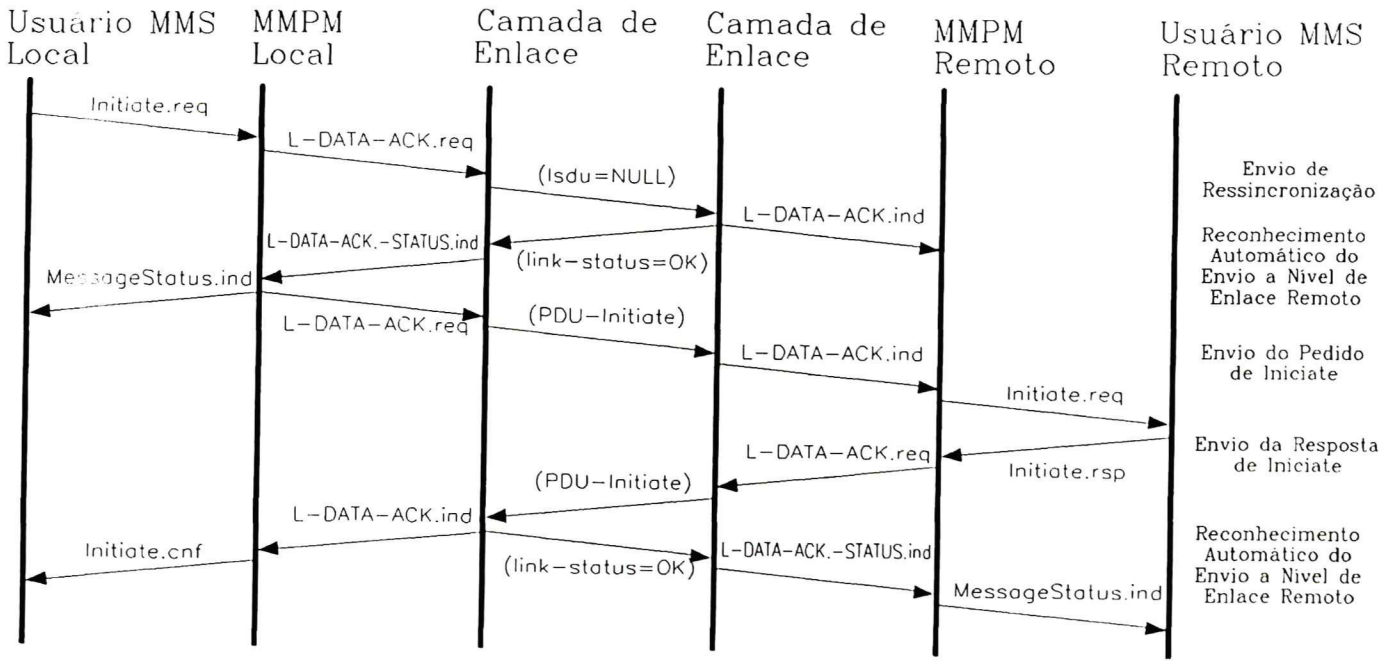


Fig. 2.9 Diagrama de Tempo: Estabelecimento de uma Associação Explícita

2.5.3 Provedor MMS Mini-MAP

Caso seja implementado sobre uma arquitetura Mini-MAP, o Provedor MMS (**MMPM/Mini-MAP**) terá as seguintes funções adicionais [MAP 88]:

- construção e desmembramento de PDUs;

- informe de status da camada de enlace para o usuário;

- coordenação e cancelamento de transação sobre condições normais e aborto devido a erros de enlace;

- estabelecimento e término de associação explícita sobre condições normais e aborto devido a erros de enlace.

Em geral, as operações envolvendo mais que um serviço, devem ser coordenadas a nível de usuário Mini-MAP. Além destas operações, algumas funções são deixadas para o usuário respondedor, não sendo providas pela MMPM/Mini-MAP [MAP 88]:

- interpretação da semântica da PDU, exceto para aspectos de baixo nível, tais como criação/término de associação ou cancelamento de transação;

- funções de alto nível, tais como a coordenação de abertura, leitura e fechamento de arquivos e a coordenação de envio de serviço *Information_Report* requisitado por uma operação de ação de evento;

- tratamento de erros referentes ao usuário Mini-MAP (que podem ser detectados localmente), tais como a passagem de uma primitiva de resposta que não tem uma indicação correspondente e violação das regras referentes ao uso de uma determinada associação;

A figura 2.10 apresenta a máquina de estados da MMPM/Mini-MAP. As transições de estados são sempre dirigidas pela chegada de primitivas de serviço vindas: de um usuário MMS, da camada de enlace ou do gerenciamento de estação.

A máquina de protocolo é específica de associação, há portanto uma instância da máquina de protocolo para cada associação. Cada instância de serviço pendente sobre a máquina de protocolo é governada por uma máquina de estados denominada Máquina de Transação, podendo num determinado instante, existir várias máquinas de transação [MAP 88].

Cada primitiva passada para ou vinda da Máquina de protocolo, contém informações tais como endereço de enlace, prioridade e identificação da Máquina de Transação (*Invoke ID*).

2.5.4 Serviços MMS Mini-MAP

Os seguintes serviços adicionais são introduzidos para suprir as defiências causadas pela falta das camadas intermediárias [MAP 88]:

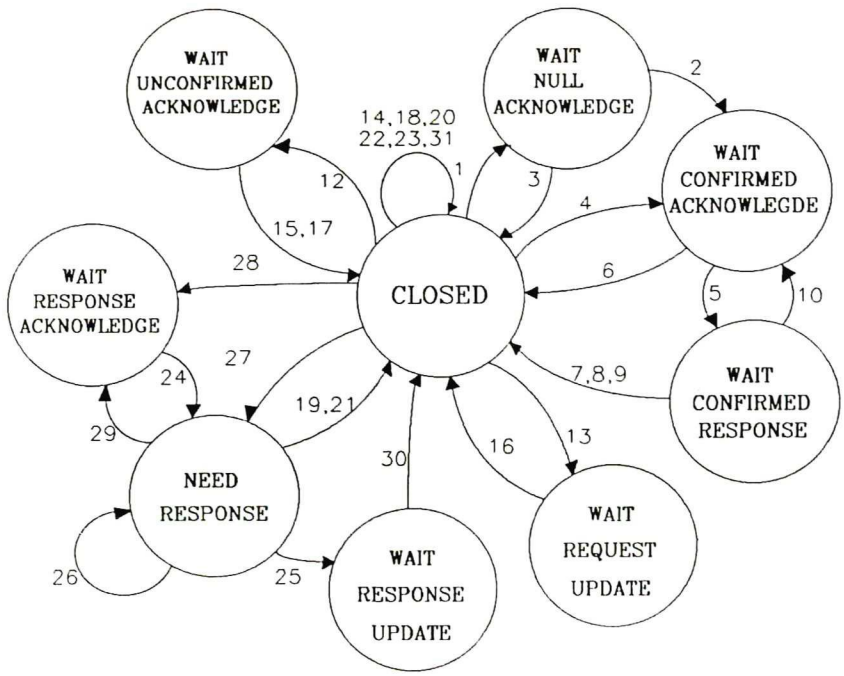
- **Message_Status**: efetua o reconhecimento imediato enviado pela camada de enlace ao usuário informando o estado de um serviço requisitado.

- **Force_Close**: permite ao usuário fornecedor do serviço cancelar uma instância de serviço localmente, sem abandonar a associação.

- **Reject_Sent**: informa o usuário da rejeição de uma PDU inválida por parte da máquina de protocolo Mini-MAP, ou ainda no caso de PDUs sem associação e PDU *Initiate* inválida.

- **Obtain_Reply**: permite a um usuário obter a resposta a uma requisição feita previamente a uma estação que não tem direitos de acesso ao anel lógico da rede de comunicação.

- **Receive_PDU**: usada na troca de informações entre o ASE-MMS e o ASE-*Obtain Reply*.



TRANSIÇÕES:

1. send-resynchronization
2. see-null-ack
3. bad-null-ack
4. send-confirmed-request
5. see-confirmed-ack
6. request-finds-bad-link
7. accept-response
8. confirmed-failure
9. station-failure-while-waiting
10. send-cancel-request
11. (deletada)
12. send-unconfirmed-request
13. prepare-unconfirmed-request
14. send-broadcast-request
15. see-unconfirmed-ack
16. see-request-update
17. deletada
18. station-failure
19. receive-confirmed-request
20. receive-unconfirmed-request
21. receive-initiate-request
22. initiate-not-permeted
23. association-not-present
24. send-response
25. send-response
26. receive-cancel-request
27. cancel-send-response
28. see-response-ack
29. response-finds-no-resourse
30. see-response-update
31. receive-erroneus-PDU

Fig. 2.10 Máquina de Protocolo MMS (MMPM/Mini-MAP)

2.6 Conclusão

Este capítulo apresentou alguns aspectos gerais sobre os Sistemas Flexíveis de Manufatura e seus requisitos de comunicação. Em particular foram mostrados os principais conceitos do Padrão MMS que se apresenta como uma das soluções para os sistemas de comunicação empregados neste tipo de aplicação.

CAPÍTULO 3

UMA METODOLOGIA PARA UTILIZAÇÃO DO PADRÃO MMS

3.1 Introdução

A geração de um sistema de comunicação para aplicações de automação fabril que seja livre de erros e eficiente torna necessário a adoção de uma metodologia para utilização dos serviços MMS já apresentados anteriormente.

Neste capítulo, será descrita a metodologia proposta neste trabalho e destacadas as características de ferramentas que ajudarão na utilização desta. Esta metodologia será testada e validada num exemplo de comunicação numa célula flexível a ser apresentada no capítulo seguinte.

3.2 A Metodologia Proposta

A partir da especificação funcional da aplicação e dos seus requisitos em termos de comunicação, esta metodologia define os Processos de Aplicação (APs) seguindo uma abordagem de refinamentos sucessivos e a seguir as interações entre estes APs utilizando os serviços MMS. Uma etapa de validação, que fará uso de Técnicas de Descrição Formal (TDF), permitirá a seguir verificar o comportamento da especificação do sistema de comunicação, antes deste ser implementado.

A metodologia de utilização dos Serviços MMS proposta se decompõe nas etapas seguintes:

- Descrição funcional da aplicação e dos requisitos de comunicação;

- Especificação dos Processos de Aplicação;

- Especificação das interações entre Processos de Aplicação;

- Validação da Especificação;

- Implementação dos Processos de Aplicação.

3.2.1 Etapa 1: Descrição Funcional da Aplicação e dos Requisitos de Comunicação

O ponto de partida desta metodologia é uma descrição precisa e sem ambiguidades das funcionalidades de cada dispositivo e do comportamento global da aplicação dada em termos funcionais e dos seus requisitos de comunicação. A descrição através de uma TDF é aconselhada para evitar ambiguidades, garantir clareza e precisão e por facilitar a análise do comportamento dinâmico da aplicação através do uso de ferramentas adequadas, como simuladores e verificadores.

3.2.2 Etapa 2: Especificação dos Processos de Aplicação

Esta especificação será feita a partir de uma abordagem do tipo *top-down*. Dos requisitos do sistema de comunicação da aplicação dada, determina-se, num primeiro nível de abstração, os elementos lógicos que fazem parte do sistema de comunicação bem como a qualificação destes para a aplicação. A seguir cada elemento é refinado a partir das funcionalidades de cada dispositivo físico da aplicação e do sistema de comunicação seguindo o modelo definido pelo Padrão MMS.

A etapa de especificação dos Processos de Aplicação pode ser decomposta nos seguintes ítens:

I. Determinação dos APs e do papel destes (cliente/servidor) dentro da aplicação;

II. Definição dos VMDs para cada AP servidor;

III. Definição dos Objetos MMS para cada VMD;

IV. Definição das Entidades de Aplicação (AEs).

I. Determinação e classificação dos APs

Um AP é, segundo o modelo ISO, um elemento lógico que caracteriza um certo processamento da informação necessária ao funcionamento da aplicação. Se a aplicação do usuário for distribuída, cada uma das componentes distribuídas será representada por um AP. Num sistema de comunicação de uma célula flexível, por exemplo, cada dispositivo deve ser decomposto em tantos APs quantos forem os processos concorrentemente executáveis neste dispositivo.

Definidos todos os APs para a aplicação dada, é necessário preocupar-se com o papel (cliente ou servidor) desempenhado por cada um deles dentro da aplicação. Cada

dispositivo será analisado segundo as funcionalidades que deverá prover e que estão contidas na descrição funcional da aplicação. Assim, um AP será classificado como cliente se ele requisita uma ou mais operações a serem executadas por outro AP; ou como servidor se ele realiza alguma operação fornecida a um AP remoto; ou ainda cliente-servidor se satisfaz a ambos os comportamentos.

A figura 3.1(a) ilustra as conclusões desta etapa sobre uma aplicação elementar composta de três Processos de Aplicação. O AP denominado "C1", durante seu processamento normal, necessita de operações a serem executadas pelos outros dois APs, tendo portanto um comportamento unicamente cliente. O AP denominado "S2" limita-se a responder as operações requisitadas pelos outros dois APs, tendo portanto um comportamento unicamente servidor. E finalmente o AP denominado "CS3" que fornece resposta às operações requeridas pelo AP "C1" e por outro lado necessita para seu processamento normal de operações executadas pelo AP "S2",tendo assim um comportamento misto cliente/servidor.

Esta classificação em AP cliente e/ou servidor é de suma importância para a etapa seguinte, responsável pela definição dos VMDs pois estes existem somente num AP servidor.

II. Definição dos VMDs para cada AP Servidor

Segundo o Padrão MMS, os APs clientes requisitam uma operação particular a um AP servidor por meio de instâncias de pedido de serviços MMS. Em consequência, cada AP servidor deve conter um ou mais Dispositivos Virtuais de Manufatura (VMD), contendo os objetos MMS que fornecem a visão externa das funcionalidades ligadas a cada serviço MMS requerido pelo AP cliente.

A representação através de VMDs é então necessária apenas para APs com comportamento servidor ou cliente/servidor. Um AP servidor pode eventualmente ter mais que um VMD, cada um deles representando um dispositivo real diferente associado ao mesmo processo de aplicação.

A figura 3.1(b) mostra a configuração dos VMDs para a aplicação elementar dada como exemplo na etapa anterior. Considera-se que cada AP Servidor está associado a apenas um dispositivo real da aplicação elementar.

Após ter definido todos os VMDs da aplicação, deve-se definir os recursos e funcionalidades de cada um dos dispositivos servidores envolvidos através da definição dos Objetos MMS associados a cada VMD.

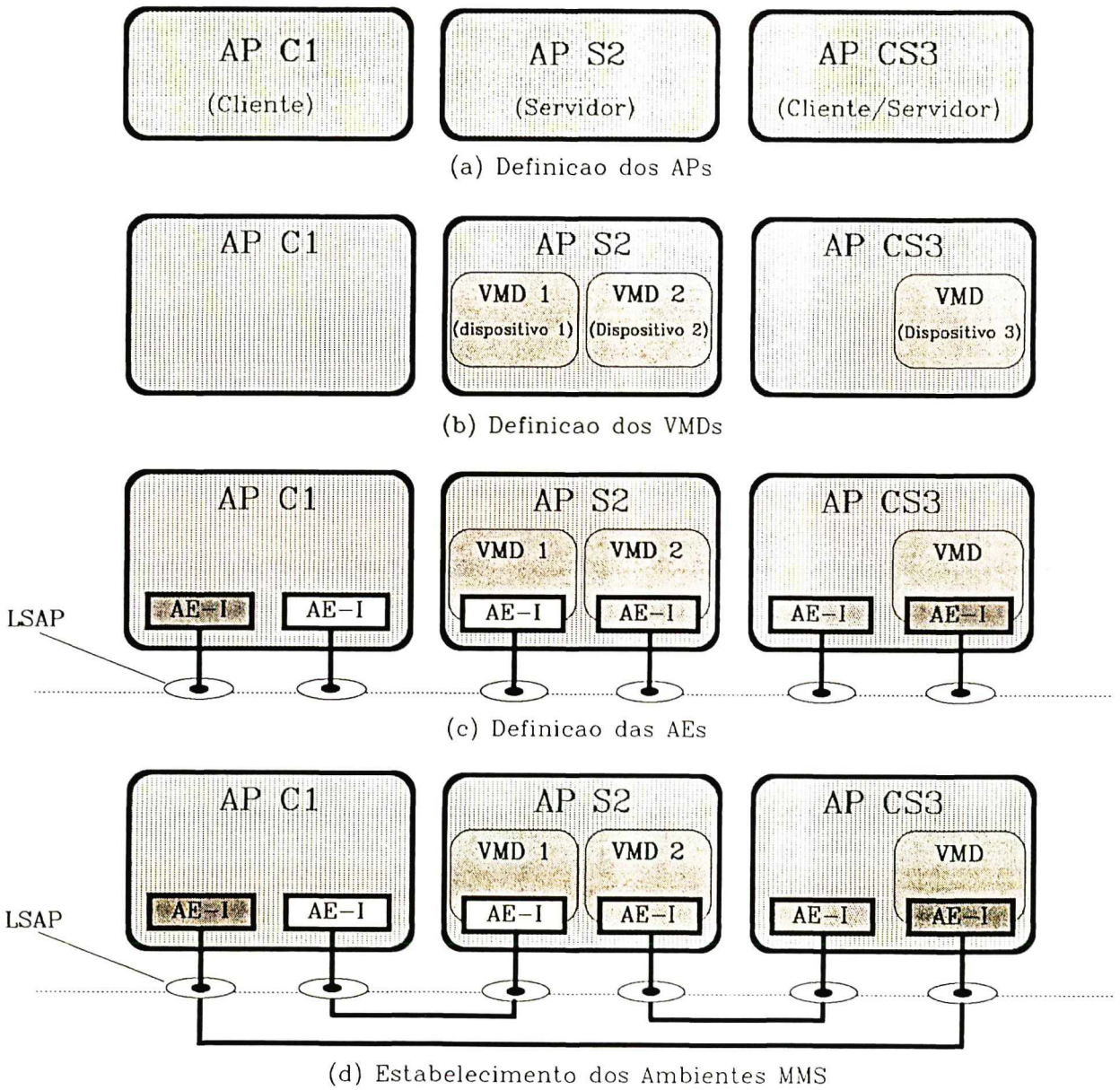


Fig. 3.1 Etapas da Metodologia numa Aplicação Elementar

III. Definição dos Objetos MMS para cada VMD

Nesta etapa serão selecionados entre os objetos MMS, aqueles que atendem às necessidades de modelamento de cada um dos dispositivos servidores, de maneira a tornar visível aos serviços MMS todos os recursos e funcionalidades relacionados com a aplicação.

Este modelamento deve ser realizado em dois níveis diferentes: um ligado às necessidades de inicialização de cada dispositivo e outro ligado às funcionalidades exigidas pela aplicação específica para cada dispositivo.

Os objetos e serviços definidos como parte dos procedimentos de inicialização, estão ligados normalmente a operações de verificação de *status* e de definição de variáveis. A escolha dos objetos e serviços relacionados com os requisitos exigidos pela aplicação específica dependerá principalmente do detalhamento de cada operação envolvida na aplicação.

A especificação contida no Padrão MMS [ISO 88] provê um conjunto de serviços e objetos comuns a todos os dispositivos e sistemas do ambiente de manufatura. De forma que o mapeamento específico para NCs, CPs, robôs e outros sistemas deve ser uma tarefa baseada no uso de cada um dos *Companion Standards*.

Para facilitar a definição de objetos e serviços MMS, os *Companion Standards* baseiam-se na padronização de cada equipamento em Classes de Conformidade, que dependem do tipo de aplicação a que se destina o equipamento. As classes de conformidade para os principais equipamentos e sistemas utilizados em ambientes fabris são apresentadas a seguir:

Controladores Programáveis (CPs)

Devido a sua grande flexibilidade de utilização, este tipo de equipamento pode prover várias funções. O *Companion Standard* para CP [IEC 89] classifica os controladores programáveis em cinco classes hierárquicas, cada uma delas adicionando mais capacidades à anterior. A tabela 3.1 mostra as classes de conformidade para os CPs e as funções adicionadas em cada uma delas. Este *Companion Standard* define ainda os serviços disponíveis para cada aplicação e para cada classe de conformidade.

Comandos Numéricos (CNs)

A classificação para estes equipamentos se dá em três níveis: *Attended NC* onde o dispositivo está inteiramente assistido por um operador local, não podendo ser controlado remotamente; *Unattended NC* onde o dispositivo coloca à disposição de um cliente remoto, um determinado número de funções; e *Distributed NC* onde o cliente remoto provê o planejamento e os recursos enquanto o dispositivo deve obter os recursos necessários à execução dos procedimentos, permitindo um

controle distribuído de várias máquinas. O *Companion Standard* para CNs [ISO 89a] define ainda sub-classes e as capacidades suportadas em cada uma delas.

Classes de Conformidade	Funções do CP
1	Aquisição de dados Controle paramétrico
2	Gerenciamento de programas
3	Aquisição de dados não solicitada Controle intertravado
4	Aquisição de dados configurada Alarmes Provimento de semáforos
5	Instanciação remota de objetos MMS Definição de Semáforos

Tab. 3.1 Classes de Conformidade para CPs

Robôs

A classificação para este tipo de equipamento é baseado nos modelos de aplicação dos robôs. Num modo servidor o robô age como se estivesse ligado a um computador supervisor que por sua vez atua como cliente. Neste caso o *Companion Standard* para Robô [ISO 89b] define uma Classe de Conformidade Básica que reune um conjunto de funções mínimas para que um robô esteja em conformidade com este padrão. Sub-classes de nível mais alto adicionam serviços dependendo das aplicações exigidas. No modo cliente o robô provê serviços de supervisão que serão usados em sensores e atuadores. Os serviços MMS suportados em cada uma destas classes pode ser encontrado em [ISO 89b].

Sistemas de Controle de Processos

O *Companion Standard* para Sistemas de Controle de Processos (SCP) define três classes de conformidade. A classe *Supervisory S0* consiste dos serviços necessários a sistemas de controle de processos relativamente simples. A classe *Supervisory S1* adiciona aos serviços da classe anterior, os serviços para comunicação com a Estação de Operador, gerência de eventos e sequência de

carregamento de programa. A classe *Supervisory S2* adiciona aos serviços da classe *Supervisory S1*, os serviços de gerência de programa, semáforos e jornal. Os serviços MMS suportados em cada uma destas classes pode ser encontrado em [ISA 90].

A partir da descrição detalhada das operações exigidas pela aplicação, o usuário deverá classificar seu equipamento numa das classes de conformidade definidas pelo respectivo *Companion Standard*. O passo seguinte consiste em considerar os objetos e serviços fornecidos para a classe escolhida e determinar quais deles devem ser implementados.

No caso de implementação de uma interface para Processos de Aplicação (API), existe ainda a possibilidade de definição local dos objetos específicos de associação do tipo Acesso a Variáveis e do tipo Semáforo. Esta definição local ou "imagem" do objeto MMS na API permite que o usuário remoto acesse o objeto sem a intervenção do usuário local, agilizando os processos de escrita/leitura de variáveis e controle de semáforo.

IV. Definição das Entidades de Aplicação

Nesta etapa, definem-se todas as Entidades de Aplicação envolvidas no sistema de comunicação. No caso do Padrão MMS apenas uma associação pode ser estabelecida sobre cada AE-I [ISO 88]. Desta forma, os programas usuários cliente e servidor devem instanciar tantas AEs, quantas forem as associações a serem estabelecidas. Além disso, durante a definição dos Tipos-AE de cada Processo de Aplicação deve-se notar que o estabelecimento posterior de cada associação deve ser feito sobre AE-Is do mesmo AE-Tipo.

Devido à inexistência das camadas de Apresentação e Sessão na arquitetura Mini-MAP, a sintaxe abstrata de cada Tipo-AE deve ser pré-definida. A especificação desta sintaxe abstrata é fornecida pelo padrão MMS genérico, sendo complementada pelos *Companion Standards*.

Neste trabalho propos-se a manutenção de um Tipo-AE para cada tipo de dispositivo diferente usado em aplicações fabris. Define-se assim, os Tipos-AE denominados AE_CN, AE_CP, AE_Robô e AE_SCP (para Sistemas de Controle de Processos), de forma que cada AE comporte a sintaxe abstrata definida em apenas um *Companion Standard* MMS. Esta abordagem reduz a necessidade de memória local (em cada dispositivo) destinada ao sistema de comunicação, pois cada Tipo-AE contém apenas as funções de comunicação especificadas no Padrão MMS genérico e as funções especificadas num determinado *Companion Standard*.

A figura 3.1(c) mostra as instâncias de AE que deverão ser ativadas para garantir as interações necessárias no exemplo da aplicação elementar definida anteriormente. Deve-se notar a ligação das AE-Is aos LSAPs é definida a priori, e que AE-Is do mesmo Tipo-AE devem ser ligadas ao mesmo LSAP.

Além de conter a máquina de estados do provedor e a sintaxe abstrata definida em cada *Companion Standard*, cada AE deve ter definido o tipo de associação lógica usado. O fato de se considerar uma arquitetura reduzida, limita as opções de comunicação àquelas baseadas em Associação Explícita e Sem Associação. A escolha de qual tipo de comunicação deverá ser definido em cada AE, depende basicamente dos requisitos definidos para o sistema de comunicação. Neste caso é importante notar a necessidade de uma comunicação orientada à conexão quando ao menos uma das seguintes operações é exigida:

- Objetos específicos do contexto de uma AA devem ser definidos;

- Parâmetros de comunicação devem ser alterados de seus valores *defaults*;

- Recursos de comunicação devem ser alocados entre dois usuários MMS.

Além disso, apesar de fornecer um meio mais rápido, o uso comunicação sem associação é limitado a alguns serviços apenas (*Read*, *write*, *Information_Report*, *Identify*, *Event_Notification*, *Acknowledge_Event_Notification*, *Input*, *Output*, *Cancel*, *Status* e *Unsolicited_Status*).

3.2.3 Etapa 3: Especificação das Interações entre os APs

Após a definição dos elementos que formam o Processo de Aplicação, passa-se ao estudo da utilização propriamente dita dos Serviços MMS. A utilização dos serviços MMS determina a interação entre os vários dispositivos que fazem parte do sistema de comunicação e envolve as etapas de:

I. Estabelecimento do ambiente de comunicação;

II. Inicialização dos dispositivos;

III. Funcionamento da aplicação.

I. Estabelecimento dos Ambientes MMS

Um ambiente MMS é obtido pelo estabelecimento de uma Associação de Aplicação (AA) entre duas Entidades de Aplicação (uma no servidor e outra no cliente). Num Sistema de Comunicação MMS devem existir tantos ambientes quantos forem os pares cliente/servidor. As AE-Is associadas (normalmente referidas como local e remota) devem possuir obrigatoriamente o mesmo AE-Tipo, pois necessitam das mesmas funções de comunicação.

Durante o estabelecimento de uma AA, o Serviço MMS *Initiate* é utilizado pelo AP Cliente, para que sejam negociadas todas as características de comunicação disponíveis (serviços, parâmetros, etc) com o AP remoto. O uso do serviço *Initiate* para o estabelecimento das associações explícitas segue o diagrama mostrado na figura 2.10 (a definição do LSAP destino deve ser feita previamente).

A figura 3.1(d) ilustra as associações estabelecidas no exemplo da aplicação anterior. Deve-se notar que estas associações são estabelecidas sempre entre duas AE-Is do mesmo Tipo-AE, fato que deve ser considerado na etapa de "Definição das Entidades de Aplicação".

Esta etapa é necessária apenas quando a comunicação entre duas AE-Is é orientada à associação (existência de uma Associação de Aplicação); caso contrário todas as características de comunicação são consideradas pré-definidas para ambas as AE-Is.

II. Especificação da Inicialização dos Servidores

Os Procedimentos que fazem parte desta etapa são dependentes do tipo do equipamento usado e da aplicação específica do usuário. Os procedimentos clássicos desta etapa são do tipo:

- definição e/ou carregamento de objetos MMS dinâmicos;

- verificação e/ou alteração de variáveis de estado;

- execução de códigos de inicialização dos dispositivos.

As tarefas definidas nesta etapa são geralmente desempenhadas pelo equipamento de supervisão, a fim de estabelecer o estado inicial do processo de produção (rotinas de inicialização de cada equipamento).

III. Especificação do Funcionamento da Aplicação

Nesta etapa, serão descritos os procedimentos MMS, com base nas operações que compõem o processo de produção da aplicação dada, definidos na etapa de "Descrição Funcional da Aplicação". Para cada operação, define-se a sequência de ações que devem ser executadas sobre os objetos (como criação/destruição e alteração de atributos) e consequentemente as sequências de primitivas MMS.

3.2.4 Etapa 4: Descrição Formal e Validação dos APs

A utilização da metodologia apresentada gera uma especificação contendo vários elementos envolvidos na comunicação. Antes de realizar a implementação, o usuário deve verificar a consistência desta especificação através de ferramentas baseadas em Técnicas de Descrição Formal. O modelo formal Rede de Petri ou linguagens de especificação como Estelle e Lotos podem ser utilizadas nesta etapa para formalização da especificação. Simuladores e/ou verificadores para estas linguagens facilitarão o entendimento do comportamento do sistema e a sua implementação.

3.2.5 Etapa 5: Implementação dos APs

A tarefa de implementação está diretamente associada ao ambiente de programação no qual o implementador irá codificar os processos de aplicação. As características de implementação dependem apenas das decisões do implementador e do ambiente utilizado, não sofrendo restrições da norma a este nível.

O uso de um *software* de comunicação baseado numa Interface padronizada como a MMSI (Manufacturing Message Specification Interface), definida para o ASE MMS em [MAP 88], permite ao usuário MMS definir alguns objetos MMS, liberando o implementador responsável pelos APs da concepção de objetos especificos de cada tipo de aplicação. Neste caso os códigos gerados para tais objetos são responsabilidade das funções da MMSI. Além disso, o uso desta interface torna o acesso aos serviços MMS mais transparente ao usuário [Willrich 91].

3.3 Uma Ferramenta de Simulação para Sistemas de Comunicação MMS

Para auxiliar o usuário na utilização da metodologia que acaba de ser descrita, propomos também neste trabalho uma ferramenta de simulação dedecada a sistemas de comunicação baseados no Padrão MMS. A especificação inicial desta ferramenta é dada a seguir e algumas de suas características testadas no capítulo seguinte.

Esta ferramenta está baseada no uso de uma linguagem de especificação formal. Escolheu-se a linguagem Estelle por apresentar as características de uma abordagem *top-down* (a mesma escolhida para a metodologia proposta), por ter uma sintaxe e semântica de fácil entendimento, e por ser adaptada a descrição de sistemas de comunicação, haja visto sua escolha pela ISO para esta função [ISO 87b]. As principais características da linguagem Estelle são apresentadas no anexo C. A Ferramenta de Simulação para Sistemas de Comunicação MMS utilizará o mecanismo de evolução e a interface homem_máquina de um ambiente de simulação genérico para especificação Estelle.

A descrição das várias entidades (APs, VMDs, Objetos MMS, AEs e Ambientes MMS) necessárias à especificação completa de uma determinada aplicação, ao longo dos vários refinamentos determinados pela metodologia proposta, serão apresentadas na forma de módulos Estelle agrupados numa biblioteca. O usuário construirá o sistema de comunicação para sua aplicação a partir da instanciação destes módulos, da definição de valores para os atributos genéricos que o parametrizam e das conexões entre as instâncias, segundo as necessidades do sistema de comunicação a ser construído.

É importante destacar que a ferramenta de simulação proposta apresenta características de flexibilidade e de facilidade na construção da aplicação. Tal fato, torna esta ferramenta valiosa para o implementador testar o comportamento de sua aplicação específica antes de implementá-la e determinar os serviços que lhe serão necessários.

A figura 3.2 apresenta um esquema dos blocos da Ferramenta de Simulação para Sistemas de Comunicação MMS.

A ferramenta de simulação proposta neste trabalho utiliza o ambiente de simulação genérico ESTIM descrito no anexo C, disponível no LCMI-UFSC.

3.3.1 Descrição da Biblioteca

A biblioteca deve conter um módulo para cada uma das diferentes entidades do sistema de comunicação. Este mapeamento de entidades de protocolo em módulos da linguagem de especificação deve seguir a especificação dada pelo Padrão MMS para cada tipo de dispositivo.

Além disso, estes módulos devem respeitar a forma de organização das entidades, Esta estruturação hierárquica pode ser alcançada através da relação entre módulos pai/filhos da linguagem Estelle.

No nível de abstração mais alto, a biblioteca contém três módulos Estelle: os módulos AP_Servidor e AP_Cliente, que encapsulam, respectivamente, as funcionalidades de um Servidor MMS e as de um Cliente MMS; e um módulo Camada_Enlace, responsável pela simulação da interface com a camada de Enlace do Mini-MAP por exemplo (LSAPs). A descrição de cada um destes módulos, bem como dos módulos-filhos que representam suas funcionalidades (entidades específicas), é dada a seguir:

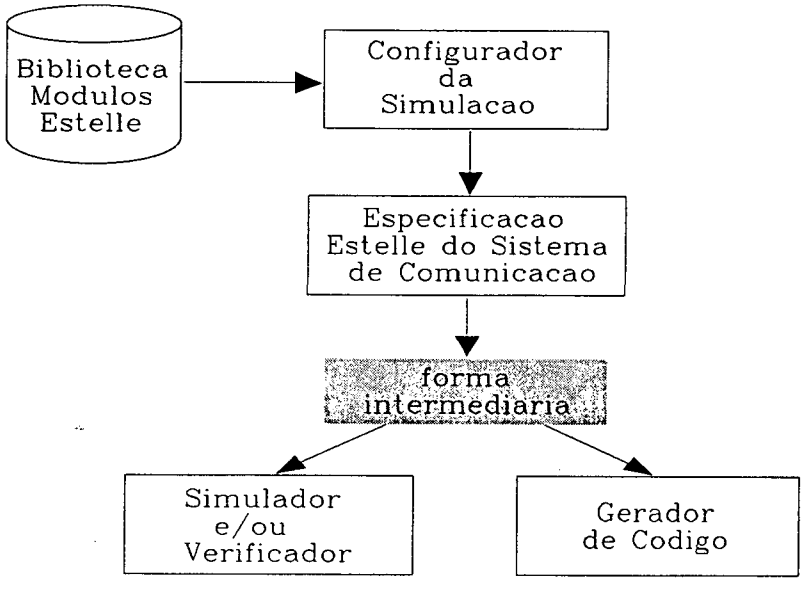


Fig. 3.2 Diagrama de Blocos da Ferramenta Proposta

I. Módulo AP_Servidor

Cada módulo AP_Servidor deve permitir a instanciação de tantos módulos AE e módulos VMD, quantos sejam necessários para cada Servidor MMS da aplicação específica do usuário.

Módulo AE

O módulo-filho AE contém a máquina de protocolo do provedor de serviços MMS para a arquitetura Mini-MAP (MMPM/Mini-MAP). O módulo é responsável também pela sintaxe abstrata usada na comunicação de cada dispositivo.

A proposta deste trabalho, de manter um Tipo-AE para cada contexto Padrão MMS genérico + *Companion Standard*, exigirá os seguintes modulos AEs, cada qual com seu conjunto próprio de funções de comunicação:

- **Módulo AE_CP**: que contém a sintaxe abstrata definida no padrão MMS genérico [ISO 88], mais a sintaxe abstrata definida no *Companion Standard* para Controladores Programáveis [IEC 89];

- **Módulo AE_NC**: que contém a sintaxe abstrata definida no padrão MMS genérico, mais a sintaxe abstrata definida no *Companion Standard* para Comandos Numéricos [ISO 89a];

- **Módulo AE_Robô**: que contém a sintaxe abstrata definida no padrão MMS genérico, mais a sintaxe abstrata definida no *Companion Standard* para Robôs [ISO 89b];

- **Módulo AE_SCP**: que contém a sintaxe abstrata definida no padrão MMS genérico, mais a sintaxe abstrata definida no *Companion Standard* para Sistemas de Controle de Processos [ISA 90].

Para manter estes módulos AEs da forma mais genérica e abrangente possível dentro da biblioteca, deve-se considerar durante a sua especificação, as classes de conformidade mais completas entre aquelas fornecidas em cada *Companion Standard.*

Módulo VMD

O módulo-filho VMD contém como variáveis os atributos MMS que listam todas as funcionalidades do dispositivo real modelado e deve conter em seu corpo os módulos Objeto_MMS e do módulo Função_Executiva com a finalidade de gerenciar o acesso aos módulos Objeto_MMS

Módulos Objeto_MMS

Os módulos Objeto_MMS (módulos-filho do Módulo VMD), modelam os atributos e a máquina de estados que rege o comportamento de cada objeto descritos nos *Companion Standards* (módulo Domínio, módulo Invoc_Prog, módulo Cond_Evento, módulo Gate, etc). Os módulos Objetos_MMS a serem instanciados em cada módulo VMD, são apenas aqueles efetivamente usados na aplicação do usuário e que foram definidos, para cada dispositivo, durante a fase de "Definição dos Objetos MMS nos VMD".

Módulo Função_Executiva

Este módulo deve controlar o acesso aos módulos Objeto MMS instanciados no VMD em questão (módulo-pai), roteando as primitivas de serviço para os

objetos de maneira coerente. Neste sentido, cada primitiva que chega a um módulo VMD deve ser classificada através das transições do módulo Função_Executiva e encaminhada ao módulo Objeto_MMS que representa o objeto de interesse do serviço. A existência de um VMD implica na própria existência de uma Função_Executiva, desta forma, este módulo será instanciado uma única vez para cada módulo VMD.

Um exemplo de estrutura de especificação Estelle montada para um Processo de Aplicação MMS servidor qualquer pode ser visto na figura 3.3.

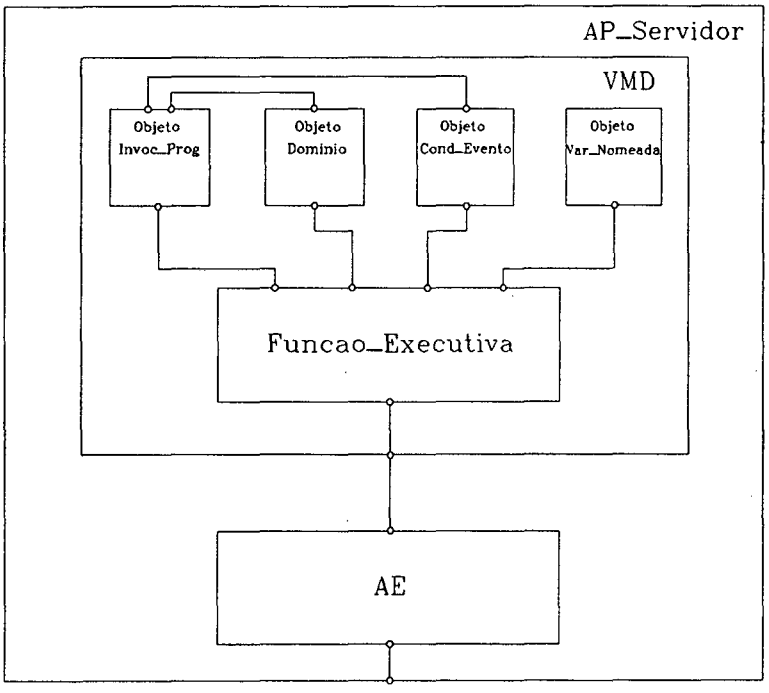


Fig. 3.3 Especificação Estelle de um AP_Servidor

II. Módulo AP_Cliente

Um módulo AP_Cliente deve permitir a instanciação de tantos módulos AE e módulos Programa_Aplicativo, quantos sejam necessários para cada Cliente MMS da aplicação específica do usuário. Um módulo-filho AE, neste caso, tem as mesmas funcionalidades que as descritas para o módulo AP_Servidor.

Módulo Programa_Aplicativo

Este módulo-filho do módulo AP Cliente deve conter a máquina de estados definida para supervisão e controle de um ou mais dos APs Servidores. O exemplo mais comum de módulo AP_Cliente é o que representa um Supervisor de Célula, onde o código do programa aplicativo deve supervisionar e controlar os demais dispositivos (AP_Servidores) para encaminhamento da produção.

A figura 3.4 mostra um exemplo de estrutura de especificação montada para um Processo de Aplicação MMS Cliente. A existência de dois módulos AE exemplifica a possível necessidade deste módulo AP_Cliente se comunicar com dois tipos de dispositivos diferentes.

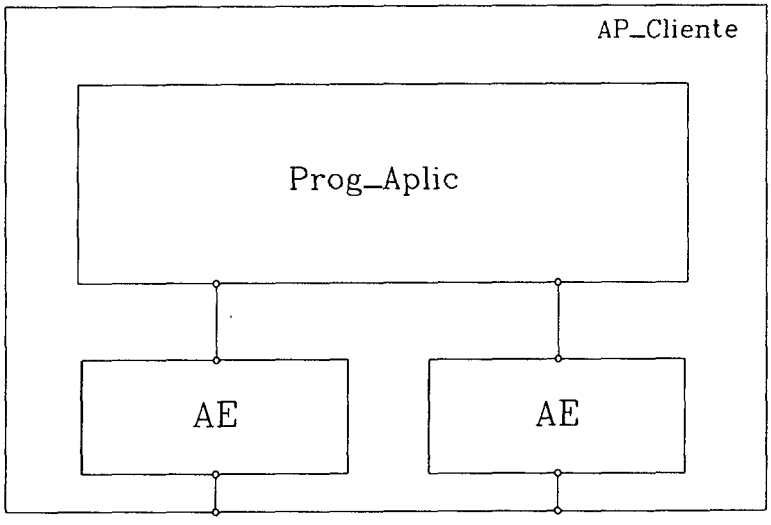


Fig. 3.4 Especificação Estelle de um AP_Cliente

III. Módulo Camada_Enlace

Para permitir nesta ferramenta de simulação a interação entre os Processos de Aplicação definidos para a aplicação do usuário, as funcionalidades da camada de enlace da arquitetura Mini-MAP serão encapsuladas em um módulo Camada_Enlace no qual os pontos de acesso aos serviços de enlace (LSAPs) fossem especificados como pontos de interação deste módulo.

Este módulo fornece unicamente as ligações lógicas entre os processos de aplicação da célula, garantindo apenas as funcionalidades de reconhecimento local imediato

(*L_Data_Ack_Status*) e transporte das primitivas de serviço MMS via primitivas de enlace (*L_Data_Ack*), até o Processo de Aplicação remoto.

3.3.2 O Configurador da Simulação

A ferramenta de simulação deverá oferecer, além da biblioteca de módulos Estelle, um conjunto de funções que permitirão a geração do código da especificação do sistema de comunicação para a aplicação específica do usuário. Este conjunto de funções deverá ser oferecido pelo Configurador da Simulação (ver figura 3.2).

O Configurador deverá garantir no mínimo as seguintes funcionalidades ao usuário:

- Carregamento dos módulos pais e filhos escolhidos da biblioteca para gerar a especificação global da aplicação;

- Instanciação de cada módulo escolhido pelo usuário;

- Definição e/ou alteração dos valores de atributos que parametrizam os módulos;

- Conexão lógica dos módulos escolhidos segundo as necessidades da aplicação.

A utilização da ferramenta segue basicamente as etapas da metodologia proposta no início deste capítulo.

3.4 Conclusão

Este capítulo apresentou uma sistematização das etapas que devem guiar um implementador na tarefa de construção de um sistema de comunicação baseado no padrão MMS, para uma aplicação fabril qualquer.

Estas etapas compiladas na forma de uma metodologia, culminaram na proposta de uma ferramenta de simulação baseada numa biblioteca de módulos Estelle reconfiguráveis, dedicados a simulação do uso do protocolo MMS em aplicações diversas da área de automação fabril. Esta facilitará o processo de concepção de um sistema de comunicação MMS e de escolha dos serviços mais apropriados para cada aplicação específica.

CAPÍTULO 4

UTILIZAÇÃO DO PADRÃO MMS NUMA CÉLULA FLEXÍVEL DE USINAGEM

4.1 Introdução

Este capítulo apresenta, a título de exemplo, a aplicação do Padrão MMS numa Célula Flexível de Usinagem (CFU) destinada à fabricação de peças prismáticas, de grande uso em um Sistema Flexível de Manufatura. Neste exemplo, a utilização do Padrão MMS para obtenção dos Processos de Aplicação será descrita segundo a metodologia apresentada no capítulo anterior.

4.2 Etapa 1: Descrição Funcional Informal da CFU

A descrição da CFU, apresentada a seguir na forma de uma especificação funcional informal, objetiva um conhecimento básico sobre os dispositivos que compõem a célula e das atividades requeridas durante o processo de produção.

4.2.1 Descrição dos Dispositivos

Para funcionar de maneira autônoma, uma célula deste tipo, deve comportar os seguintes elementos:

- um ou mais centros de usinagem;

- sistema de transporte e armazenamento de peças;

- dispositivo de manipulação e armazenamento de ferramentas;

- sistema de controle (computador supervisor da célula).

Neste trabalho será usada a configuração de célula do Projeto CIM da UFSC [CERTI 88] (figura 4.1) que é composta pelos seguintes equipamentos:

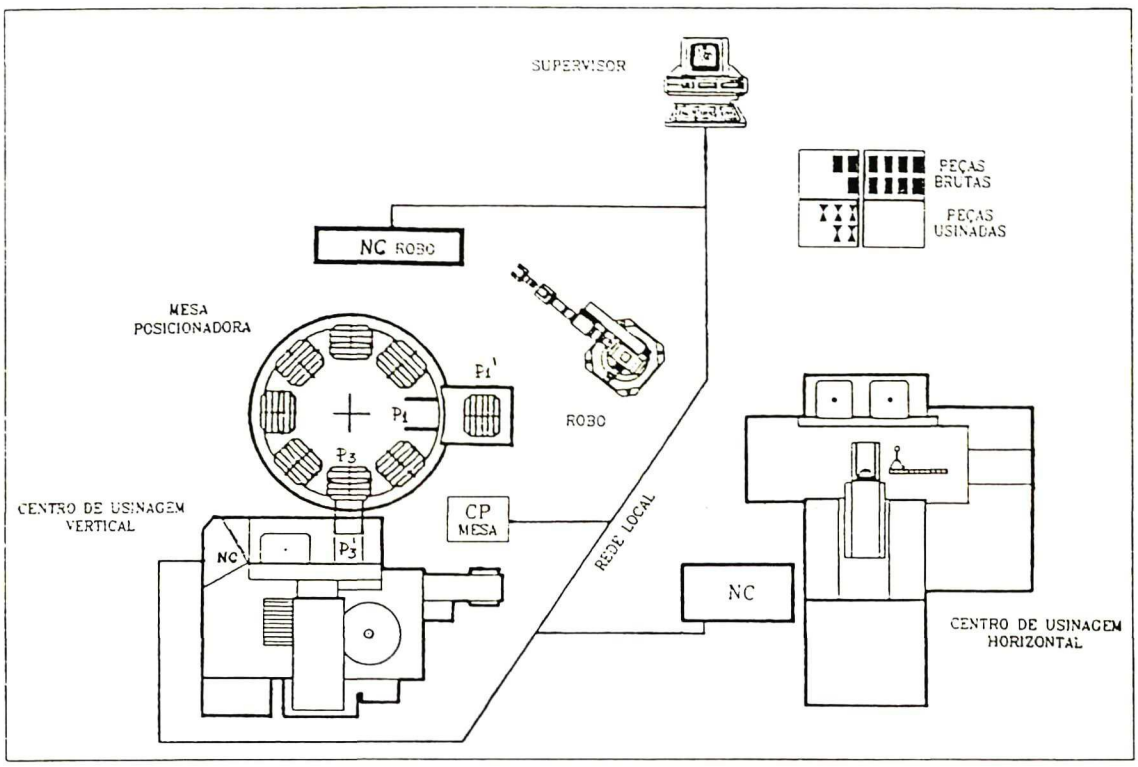


Fig. 4.1 Lay-out da CFU

I. Centros de Usinagem

A CFU possui dois centros de usinagem, um destinado à usinagem vertical e outro à usinagem horizontal. Cada centro de usinagem é acionado por um Comando Numérico (CN) que efetua o armazenamento e execução dos programas-peça (que definem o sequenciamento das operações de usinagem sobre uma peça). O CN também é responsável pelos processos de medição *in-process*, cujo objetivo é detectar erros na usinagem, aferir elementos de corte e detectar erros no posicionamento da peça.

Será assumido que o armazém de ferramentas possui capacidade para comportar todas as ferramentas necessárias à usinagem de ao menos um tipo de peça, e só poderá ser trocado manualmente. O operador deverá ser previamente informado sobre este fato, através da estação do operador local.

A colocação e retirada do *pallet* na posição de trabalho da máquina é feita pela própria máquina no início e no fim de cada programa-peça.

II. Mesa Posicionadora

Este dispositivo tem como função básica, armazenar peças que serão usinadas no centro vertical (vindos do armazém de peças brutas ou do centro de usinagem horizontal) e peças que já foram usinadas neste centro (aguardando um processamento no centro de usinagem horizontal, ou colocação no armazém de peças prontas).

A mesa é acionada por um Controlador Programável (CP), que executa os procedimentos de movimentação da mesa, que deverão ser ativados pelo supervisor. Estes procedimentos são definidos como:

- **Rot(Px,Py)**, rotaciona a posição origem "Px" da mesa até a posição destino "Py", onde "Py" pode assumir os valores P1 e P3 (figura 4.1). A saída do *pallet* para as posições de acesso pelo robô e entrada no centro de usinagem é executada automaticamente após a rotação;

- **P3in**, inserção do *pallet* situado na posição P3' (entrada do centro de usinagem);

- **P1in**, inserção do *pallet* situado na posição P1' (acesso pelo robô);

Cada uma destas funções básicas da mesa posicionadora é sucedida pelo envio de reconhecimento com a informação de sucesso ou fracasso, ao dispositivo remoto que a invocou (no caso, o supervisor).

Durante a permanência do *pallet* na posição P3', a mesa poderá realizar qualquer outro movimento. Esta funcionalidade possibilita a entrada/saída de peças da mesa durante uma etapa de usinagem vertical.

III. Robô

A função do robô é transportar peças entre os vários pontos da célula. Ele é acionado por um Comando Numérico (CNR), que armazenará o programa de transporte de peças (**TRANS**).

O supervisor comandará o robô através de uma chamada de procedimento de transporte, que conterá os parâmetros necessários para definição da trajetória e do tipo de peça a transportar, possibilitando proceder a manipulação de acordo com as características mecânicas de cada peça. As opções possíveis de trajetória são apresentadas na tabela 4.1.

IV. Supervisor

Trata-se de um microcomputador dotado de interfaces de comunicação que permite acessar tanto os níveis superiores da hierarquia fabril, quanto os demais dispositivos da célula. Desta forma o supervisor recebe as diretivas dos níveis superiores e as usa para controlar a produção. O supervisor deve armazenar as seguintes informações:

- regras de escalonamento (sequenciamento da produção);

- programas-peça em uso na produção de um lote de peças;

- informações sobre as ferramentas;

- programas para transporte e manipulação de peças.

O supervisor é responsável pela inicialização de todos os dispositivos programáveis e pelo encaminhamento da produção segundo as regras de escalonamento.

Funcao	Origem	Destino
TRANS(APB,P1,tpec)	armazem de pecas brutas	entrada/saida da mesa
TRANS(APB,H,tpec)	armazem de pecas brutas	pallet h
TRANS(P1,H,tpec)	entrada/saida da mesa	pallet h
TRANS(H,P1,tpec)	pallet h	entrada/saida da mesa
TRANS(P1,APA,tpec)	entrada/saida da mesa	armazem de pecas acabadas
TRANS(H,APA,tpec)	pallet h	armazem de pecas acabadas
TRANS(P1,Lx,tpec)	entrada/saida da mesa	armazem de pecas defeituosas
TRANS(H,Lx,tpec)	pallet h	armazem de pecas defeituosas

Tab. 4.1 Combinações válidas para trajetória do Robô

4.2.2 Descrição do Comportamento da Célula

O aspecto mais importante na descrição da CFU está relacionado com a interação de cada equipamento com os demais dentro da célula. Cada um dos equipamentos deve realizar uma ou mais funções, que sincronizadas, irão determinar o funcionamento da CFU durante o processo de produção.

O funcionamento da célula pode ser dividido em duas etapas principais. A primeira diz respeito à inicialização dos equipamentos e a segunda constitui o próprio processo de produção da célula.

A etapa de inicialização caracteriza os procedimentos básicos, tomados por parte do supervisor, para definição dos contextos iniciais de cada um dos equipamentos da célula. Durante esta etapa o supervisor deverá desempenhar ações do tipo:

- alterar o valor de variáveis de controle dos equipamentos visando o início do processo de produção;

- carregar os programas de controle do robô e da mesa posicionadora;

- fornecer os programas-peça e ferramentas para início de usinagem; entre outros.

Após ter sido inicializada, a célula estará pronta para começar o processo de produção. Nesta fase, o supervisor terá um completo conhecimento do estado da célula, visto que ele é o responsável pela ativação das ações da mesma. Desta forma, o supervisor poderá controlar todas as operações, tomando como base os eventos ocorridos e as especificações do escalonamento de produção.

O funcionamento da célula pode ser analisado através das tarefas básicas, executadas desde a escolha de uma determinada peça bruta até o armazenamento desta mesma peça já na forma de peça usinada. Estas tarefas básicas são descritas informalmente a seguir:

I. Colocação de Peça na Mesa Posicionadora

Toda peça que exigir uma usinagem vertical deverá ser colocada primeiramente na posição de entrada da mesa posicionadora, para posteriormente acessar o centro de usinagem vertical. O tipo da peça é definido pelo escalonamento da produção e a origem poderá ser tanto o armazém de peças brutas quanto a mesa de trabalho do centro de usinagem horizontal (no caso de uma peça previamente usinada neste centro e que necessite ser usinada também no centro de usinagem vertical).

II. Colocação de Peça no Centro Vertical

O supervisor, a partir da configuração de peças na mesa e das especificações de escalonamento, definirá qual peça bruta será colocada na posição P3' (posição de entrada/saída do centro de usinagem vertical).

III. Colocação de Peça no Centro Horizontal

Sempre que estiver livre, o centro de usinagem horizontal poderá ser carregado por ordem do supervisor. O tipo de peça será especificado pelo escalonamento e a origem poderá ser tanto o armazém de peças brutas quanto a mesa.

IV. Usinagem

O supervisor, a partir do escalonamento de produção, definirá que peça será usinada e de que forma. Após isto deverá prover todos os recursos necessários ao NC escolhido para que a partida do processo de usinagem possa ser dada (carregamento de peça, ferramentas, programa-peça, etc).

Há três situações com relação ao procedimento de usinagem que devem ser especificadas. A primeira situação envolve a primeira usinagem de determinado tipo de peça. Neste caso, deve-se carregar o programa e os dados necessários ao procedimento de usinagem deste tipo de peça. Como procedimento local, algumas informações acerca do processo de usinagem da peça devem ser enviadas à estação do operador, para que este proceda a colocação das ferramentas no armazém (quando necessário). Após a colocação das ferramentas, o operador deve informar (através da estação do operador) ao CN, que o processo de usinagem já pode começar.

Uma segunda situação ocorre durante a usinagem de peças do mesmo tipo, quando pode haver necessidade de troca de alguns parâmetros do programa-peça para fins de detalhamentos num mesmo tipo de peça, mudança da velocidade de corte e outras características de usinagem. Após o encerramento de cada processo de usinagem, o supervisor deverá ler os dados gerados durante os ciclos de medição das ferramentas. Estes dados servirão para manter o arquivo de informações da ferramentaria atualizado, e eventualmente requisitar a troca de ferramenta (através de mensagem à estação do operador).

A última situação refere-se a mudança do tipo de peça a usinar, onde o supervisor deve proceder nova etapa de carregamento de programa-peça e dos dados necessários ao procedimento de usinagem.

Durante toda a operação com os centros de usinagem, considera-se que todos os eventos significativos, como falhas no equipamento, serão informados ao supervisor.

V. Retirada de Peça Usinada do Centros de Usinagem

O supervisor, informado acerca de um fim de usinagem, deverá então, proceder a retirada da peça usinada da máquina. No caso do centro horizontal, a peça já usinada deverá ser retirada e levada ao seu local de destino. No caso do centro vertical a peça será depositada na posição vazia (sem *pallet*) da mesa.

VI. Retirada de Peça Usinada da Mesa Posicionadora

O supervisor definirá qual peça já usinada da mesa será retirada e levada ao local de destino. Esta tarefa também obedece ao escalonamento da produção.

4.2.3 Descrição Formal do Funcionamento da Célula

As etapas de descrição vistas nos ítens anteriores abordaram a especificação do comportamento da CFU de maneira informal. Para verificar a conformidade da especificação e também apresentá-la de forma mais concisa e correta, pode ser interessante fazer uso de ferramentas de especificação formal tais como Redes de Petri, Estelle e Lotos.

Neste trabalho, foi simulada a especificação Estelle mostrada na figura 4.2. Esta simulação permitiu conhecer o comportamento exigido da Célula Flexível de Usinagem e obter informações adicionais a respeito de suas necessidades em termos de comunicação.

A simulação baseou-se na utilização da ferramenta ESTIM [Saqui 90], um Simulador de Especificações Estelle cujas principais características são descritas no anexo C.

4.3 Etapa 2: Especificação dos Processos de Aplicação

Como descrito na metodologia apresentada no capítulo anterior, a etapa de especificação dos Processos de Aplicação se subdivide nos seguintes ítens:

4.3.1 Definição dos APs

A partir da análise da especificação funcional da CFU, serão definidos os APs necessários ao processo de produção. Por se tratar de uma aplicação inerentemente distribuída, cada equipamento da célula deve possuir ao menos um AP.

Considerando que os equipamentos não dispõem de multi-processamento, podemos assumir que cada equipamento deve ser modelado por um único AP.

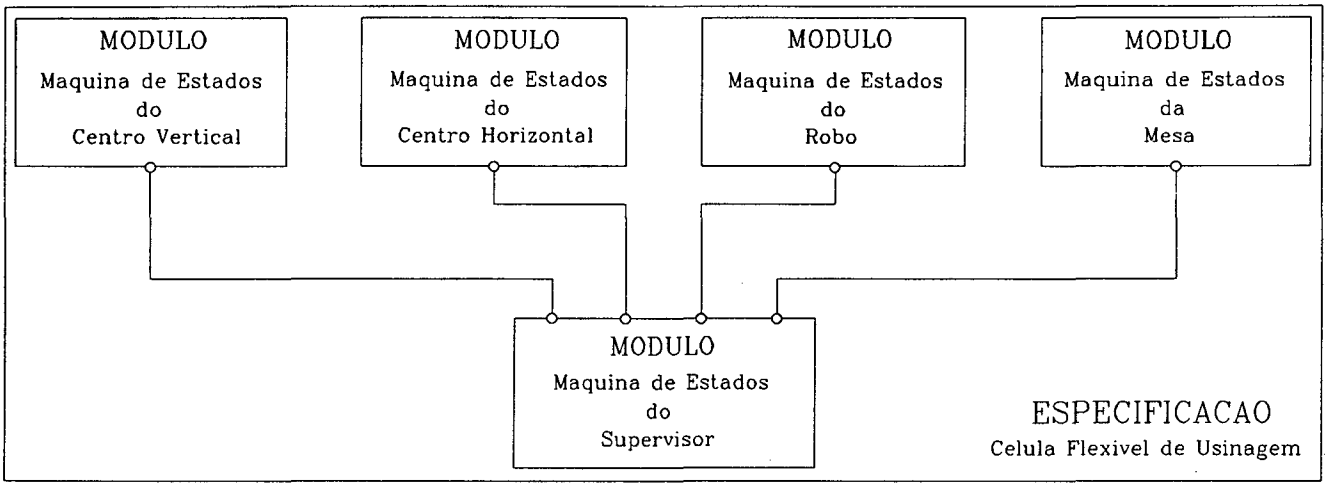


Fig. 4.2 Especificação Estelle do Comportamento exigido da CFU

4.3.2 Definição dos VMDs

Como já foi comentado anteriormente, apenas os APs que possuem caráter de servidor necessitam ser modelados por VMDs. Analisando a descrição da CFU podemos considerar que o AP_Supervisor é um AP Cliente dos demais equipamentos da célula (centros de usinagem, robô e a mesa posicionadora), que tem papel de servidores e que terão seus APs modelados por um VMD. A figura 4.3 mostra a distribuição dos APs e VMDs para a CFU.

4.3.3 Definição dos Objetos MMS contidos nos VMDs

Após a definição dos VMDs existentes na célula, cada uma das funcionalidades requeridas na descrição funcional dos equipamentos deve ser modelado por objetos MMS.

Este processo permitirá que os VMDs possam externar o comportamento referente a cada equipamento com relação ao sistema de comunicação.

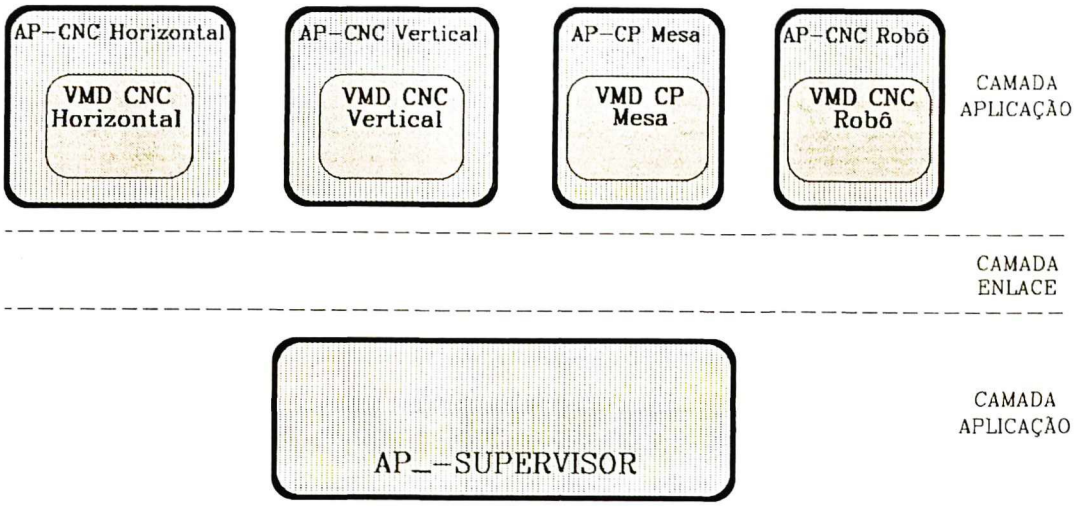


Fig. 4.3 VMDs no Ambiente da CFU

Como vimos, o modelamento dos Objetos MMS nos VMDs é específico de cada equipamento, e baseia-se nas considerações do Padrão MMS genérico e nas especificações adicionais dos *Companion Standards*.

I. Definição dos Objetos MMS dos Centros de Usinagem

Como visto no capítulo 2, o *Companion Standard* para máquinas CN classifica estes dispositivos de acordo com o tipo de operação executada em um ambiente de manufatura. Neste exemplo, são necessários os serviços de transferência de dados (dados sobre ferramentas) e de resposta a inspeções pedidas pelo supervisor. Isto permite definir que os CNs devem pertencer à subclasse *Full Unattended NC* [ISO 89a]. Nesta classe poderão ser criados e deletados remotamente (pelo supervisor) domínios do tipo *Program Machine* além de permitir também o controle da execução destes domínios através de condições de eventos.

Não há necessidade de enquadrar os CNs desta aplicação na classe *Distributed NC* porque a função de controle destes sistemas é assumida pelo supervisor da célula e não pelos próprios CNs como previsto nesta classe.

A seguir são apresentadas as principais considerações para definirmos os objetos necessários ao VMD dos centros de usinagem nesta aplicação (considerando a classe *Full Unattended NC*):

Objetos Domínio (Domain)

O *Companion Standard* define vários tipos de Domínio (através do atributo adicional *Domain Type*), cada tipo identifica o conteúdo do Domínio e especifica o formato dos dados associados. O conteúdo de um objeto Domínio consiste de programas executáveis, dados e outros objetos subordinados. Quando executado pelo CN, os dados contidos no Domínio são passados para uma estrutura interna apropriada; no caso de Domínio de programa são geradas ações de máquinas para executar alguma função útil. Os tipos de domínio utilizados no caso desta aplicação são:

- **Machine Program,** que contém uma sequência de passos de execução. O nome padrão para este tipo de Domínio é N_PRG_ seguido pelo nome do programa máquina (definido pelo usuário). Cada CN da célula conterá ao menos um Domínio deste tipo denominado **N_PRG_x** (onde "x" identifica o tipo da peça a ser usinada pelo programa).

- **Tool Data Table,** O nome padrão é N_TLD_ seguido pelo nome da tabela de ferramentas necessárias à usinagem de uma peça (definido pelo usuário). Nesta aplicação o Domínio será identificado pelo nome **N_TLD_x** (onde "x" identifica a tabela de ferramentas usadas para fabricação da peça do tipo "x"). Este Domínio servirá para a identificação do conjunto de ferramentas necessárias à usinagem de uma peça e o ajuste da ferramenta. Após a operação de usinagem pode-se realizar o carregamento (*upload*) deste Domínio para verificar o estado das ferramentas, e atualizar as informações das ferramentas.

- **Statistical Data,** que contém dados coletados dinamicamente. O nome padrão é N_SPD_ seguido pelo nome da aquisição definida pelo usuário. Cada CN conterá um objeto Domínio deste tipo, que será denotado pelo nome **N_SPD_Med** e armazenará as medidas obtidas na medição *in-process*.

Outros atributos adicionais dão ainda, informações acerca do tipo, tamanho em octetos, data e hora da criação e nome do arquivo fonte onde pode ser encontrado o Domínio.

Alguns objetos Variáveis Nomeadas (*Named Variable*) são específicos do Domínio *Machine Program* e aparecem na sua especificação com intuito de descrever características

de usinagem do programa-peça. Estes objetos especificam eixos, *offsets*, taxa de alimentação, velocidade de corte limite entre outros. Tais objetos não são listados aqui por questão de simplificação.

Todos os objetos Domínios deste VMD são considerados dinâmicos, devendo ser carregados pelo supervisor em tempo de inicialização ou manutenção de programa-peça.

Objetos Invocação de Programa (Program Invocation)

Os objetos Invocação de Programa serão usados neste caso, apenas para controlar remotamente a execução de um programa-peça. Existirão portanto, apenas objetos Invocação de Programa do tipo *Active Program*, que são usados para ativação de Domínios do tipo *Machine Program* [ISO 89a]. O controle de um programa-peça poderá ser feito então, utilizando os serviços MMS *Start*, *Stop*, *Resume*, *Kill* e *Reset* sobre o objeto Invocação de Programa. O nome padrão para este tipo de objeto é N_ACT_ seguido pelo nome do programa-peça (**N_ACT_x**).

O *Companion Standard* também define alguns atributos adicionais para o objeto Invocação de Programa. Estes atributos dizem respeito à sequencialização de programas-peça e à definição de sub-estados.

Objetos da Classe Evento (Event Condition)

Os seguintes objetos Condição de Evento são pré-definidos no VMD (estáticos), possibilitando que o supervisor receba notificações de eventos importantes ocorridos no equipamento durante o processo de produção.

N_LCL - entrada em LOCAL CONTROL.
N_RMT - entrada em REMOTE CONTROL.
N_IDL - entrada no estado IDLE.
N_RDY - entrada no estado READY.
N_NRY - entrada no estado NOT READY.
N_UVC - ocorrência da condição de sobre tensão.
N_ELO - ocorrência de sobre carga elétrica.
N_LHP - ocorrência de baixa pressão hidráulica.
N_LBF - falha no sistema de lubrificação.
N_LAP - ocorrência de baixa pressão de ar.
N_COF - ocorrência de falha no refrigerante.
N_SFV - ocorrência de violação de segurança.

N_CYS - *cycle start* ativado.
N_CYI - *cycle inhibit* ativado.
N_EFE - erro seguindo excessão.
N_PGS - uma ocorrência de parada de programa.
N_EOP - uma ocorrência de parada de fim de programa.
N_TLC - ocorrência de troca de ferramenta.
N_EOD - ocorrência de fim de dados do programa.
N_TWP - ultrapassagem do limite de corte da ferramenta.
N_PER - ocorrência de um erro de programa.
N_PFH - ocorrência de *feedhold* em um programa ativo.

O *Companion Standard* para máquinas CN não padroniza estes objetos, devendo portanto seguir a especificação contida no Padrão MMS genérico.

Para que as notificações de evento sejam enviadas ao destinatário remoto (neste caso, o supervisor), é necessário definir um objeto Registro de Evento (*Event Enrollment*)para cada um dos objetos Condição de Evento citados acima. Neste caso, nomes serão idênticos aos objetos Condição de Evento e suas estruturas também como definidas no Padrão MMS genérico.

Objeto Operador de Estação (Operator Station)

Deverá modelar a forma de comunicação do CN com o operador local através da console de programação da máquina (o nome padronizado é **N_Operator**). Os atributos *Input Buffer* e *List of Output Buffer* funcionam como memória temporária para os dados passados pelos Serviços MMS *Input* e *Output* respectivamente.

Objetos Machine Control

Este objeto adicionado pelo *Companion Standard* é uma representação abstrata das chaves lógicas providas para permitir a ativação remota (*true* ou *false*) de características da máquina. A estrutura de um objeto *Machine Control* deve ser mapeada sobre um objeto Variável Nomeada pré-definido no VMD [ISO 89a]. São objetos da classe *Machine Control*:

- **N_MachinePower**, definida para ligar/desligar remotamente a alimentação da máquina CN. Através de um pedido de *Write* sobre esta variável MMS, o usuário transfere o estado da máquina CN de *IDLE* para *READY* ou vice-versa.

- **N_ControlLocal,** ativa e desativa o modo remoto. Através de um pedido de *Write* sobre esta variável MMS, o usuário pode colocar a máquina em modo local ou remoto.

Objeto Store

Este objeto é adicionado pelo *Companion Standard* para representação abstrata das capacidades de armazenamento dos domínios e outros Objetos no VMD. O objeto adicional *Store* deve ser mapeado em um objeto Variável Nomeada estático com a seguinte estrutura para o parâmetro *Type Description* [ISO 89a]:

Local Store Name = (identificador no VMD)
Store type = (tipos possíveis de armazenamento)
Store Entries = (número total de objetos armazenáveis)
Store Capability = (número de octetos disponíveis)
Current Entries = (número atual de objetos)
Remaining Entries = (número de entradas restantes)
Remaining Capacity = (número de octetos restantes)

Objeto Actual Tool

Este objeto adicional contém informações acerca de uma ferramenta específica. Segundo o *Companion Standard*, o Objeto *Actual Tool* deve ser mapeado em um objeto MMS Variável Nomeada estático, com a seguinte estrutura para o parâmetro *Type Description* [ISO 89a]:

Tool ID = (identificador único da ferramenta no VMD)
Next Tool = (próxima ferramenta no armazém)
Tool Type = (tipo da ferramenta)
Insert Type = (define as operações possíveis na peça)
Nominal Length = (designa o comprimento da ferramenta)
Nominal Diameter = (designa o diâmetro da ferramenta)
List of Tool Edges = (lista dos elementos de corte)
Home Matrix ID = (posição do armazém de ferramentas)
Home Pocket ID = (posição da ferramenta no armazém)
Tool State = (condição operaconal atual da ferramenta)
Tool Location = (referencia o objeto que contém ou podeconter esta ferramenta)

Objeto Tool Matrix

Este objeto contém informações acerca de um armazém de ferramentas e segundo o *Companion Standard* deve ser mapeado em um objeto MMS Variável Nomeada estático, com a seguinte estrutura para o parâmetro *Type Description* [ISO 89a]:

Matrix ID = (identificador único do armazém no VMD)

Number of Pockets = (número de posições possíveis)
Spacing of Pockets = (medida grosseira do máximo diâmetro de ferramenta suportável)
List of Actual Tool Objects = (lista das ferramentas encontradas neste armazém)
List of Machine Paths = (lista das máquinas capazes de usar as ferramentas deste armazém)

Objeto VMD

Finalmente pode-se definir a forma padrão para o objeto VMD dos centros de usinagem. Este modelo apresenta alguns atributos adicionais que são definidos apenas no *Companion Standard* para máquinas CN.

Object: VMD
Vendor Name = (específico de implementação)
Model Name = (específico de implementação)
Revision = (específico de implementação)
Logical Status = {STATE_CHANGES_ALLOWED,
NO_STATE_CHANGES_ALLOWED,
LIMITED_SERVICES_SUPPORTED}
List of Capabilities = (específico de implementação)
Physical Status = {OPERATIONAL, PARTIALLY_OPERATIONAL,
INOPERABLE, NEEDS_COMMISSIONING}
List of Program Invocations = [N_ACT_x,]
List of Domains = [N_PRG_x, N_TLD_x, N_SPD_Med]
List of Transaction Objects = (criado nos pedidos de serviços confirmados)
List of Upload State Machine = EMPTY
Lists of Other VMD-Specific Objects = [
Operator Station: [N_Operator]
Named Variable: [N_MachinePower, N_Store,
N_ControlLocal, N_ToolMatrix, N_Tool]
Event Condition: [N_LCL, N_RMT, N_IDL, N_RDY,
N_NRY, N_UVC, N_ELO, N_LHP, N_LBF, N_LAP,
N_COF, N_SFV, N_CYS, N_CYI, N_EFE, N_PGS,
N_EOP, N_TLC, N_EOD, N_TWP, N_PER, N_PFH]
Event Enrollment: [N_LCL, N_RMT, N_IDL, N_RDY,
N_NRY, N_UVC, N_ELO, N_LHP, N_LBF, N_LAP,
N_COF, N_SFV, N_CYS, N_CYI, N_EFE, N_PGS,
N_EOP, N_TLC, N_EOD, N_TWP, N_PER, N_PFH]]
NC State = {REMOTE_CONTROL, LOCAL_CONTROL, IDLE, READY,
RUNNING, HOLD, NOT_READY}
Current State = (detalhe das condições operacionais)
List of Tool Matrices = [N_ToolMatrix]
List of Machine Paths = EMPTY
List of Parking Locations = EMPTY
List of Machine Axes = EMPTY
List of Machine Controls = [N_MachinePower,
N_ControlLocal]
List of Executable Programs = [N_ACT_x]
List of Parts = EMPTY
List of Pallets = EMPTY
List of Local Store = [N_Store]
List of Accessories = (específico de implementação)

Uma representação gráfica do VMD para um dos Centros de Usinagem é apresentada na figura 4.4.

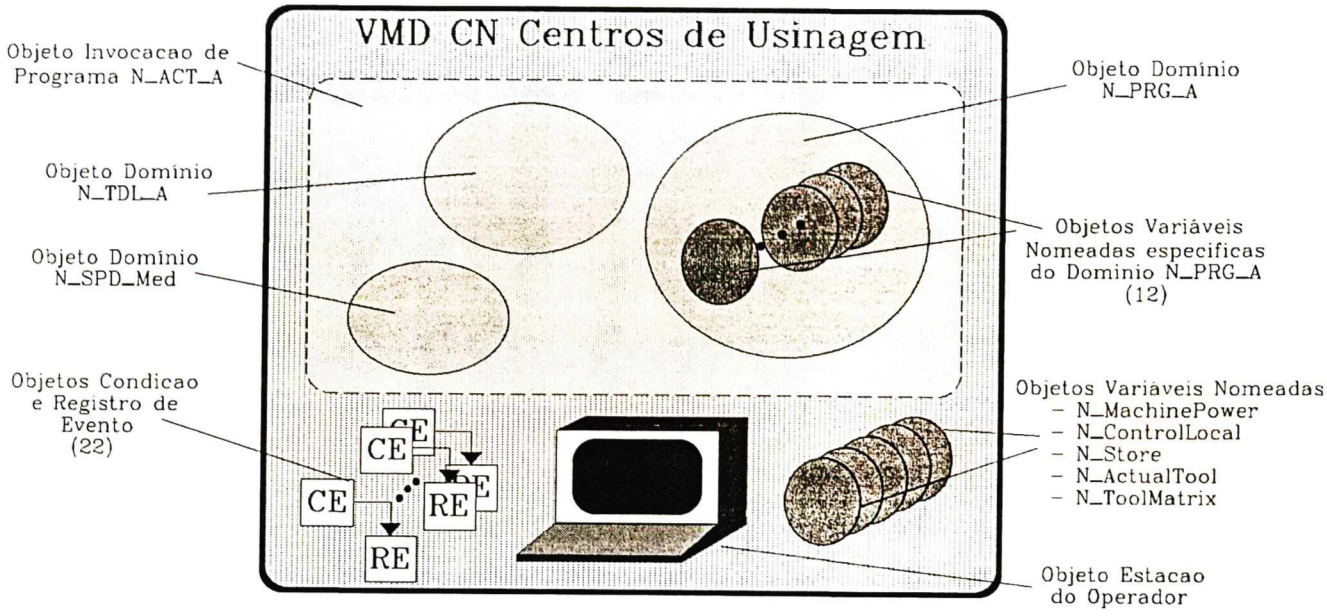


Fig. 4.4 VMD dos Centros de Usinagem

II. Definição dos Objetos MMS no VMD do Robô

O *Companion Standard* [ISO 89b] classifica a atuação do robô nesta aplicação como: "*Robot Server, Single Client*", onde o robô age como servidor de um único cliente (Supervisor da célula). São definidos vários objetos e atributos adicionais ao padrão MMS básico, com a finalidade de tornar visível as atividades desempenhadas pelo robô. Em especial, é introduzido o conceito de interrelacionamento entre Invocações de Programa, característica importante para este tipo de equipamento.

Objetos Domínio

Existem neste VMD a necessidade de quatro objetos Domínio, responsáveis pelo modelo das características físicas e operacionais do robô.

O primeiro representa o recurso braço do robô e é padronizado com o nome **R_ARM** (*Robot Arm*). Este recurso é composto pela garra de manipulação, pelos servomecanismos (um para cada grau de liberdade) e pelo sistema de planejamento de rota, responsável pela conversão de uma trajetória desejada em comandos para os

servomecanismos. Cada um destes componentes adiciona um número de atributos específicos ao objeto Domínio R_ARM de forma a modelar por completo a cinemática e o controle do braço do robô.

Alguns objetos Variáveis Nomeadas são específicos do Domínio R_ARM e aparecem na sua especificação com intuito de descrever características de controle do braço do robô. Estes objetos especificam modo de operação (local ou remoto), relação entre sistemas de coordenadas, calibração das junções, velocidade de movimento entre outros. Tais objetos não são listados aqui por questão de simplificação.

Os demais Domínios modelam os subsistemas que não fazem parte do braço do robô e são chamados pelo *Companion Standard* de objetos *Auxiliary Devices* (todos mapeados em objetos Domínio). São eles os Domínios **R_SAFE** (*Robot Safeguard*), associado aos dispositivos de proteção; **R_CAL** (*Robot Calibration*), que contém os procedimentos de calibração das diversas junções do braço do robô; e o último **R_TRANS**, que deve ser definido pelo usuário para conter o código do programa que controla o movimento do robô na célula. Note-se que este programa de controle de movimento deve implementar todas as características da função "Trans", que definimos na etapa de descrição funcional, devendo ser carregado pelo supervisor durante o processo de inicialização da aplicação.

Objetos Invocação de Programa

O modelo do objeto Invocação de Programa descrito no *Companion Standard* [ISO 89b] adiciona o conceito de interrelacionamento entre dois ou mais destes objetos. Esta funcionalidade consiste em referenciar uma Invocação de Programa como controladora e outra como controlada. O serviço adicional *Select* permite ligar uma Invocação de Programa controladora a uma Invocação de Programa controlada.

Geralmente duas Invocações de Programa são modeladas como tendo canais de comunicação interprocessos por onde passam mensagens, proporcionando a troca de dados e a sincronização entre processos. As indicações de pedido de serviço *Start*, *Stop*, *Resume* e *Reset*, recebidas pela Invocação controladora são passadas para a controlada, possivelmente com parâmetros alterados. As respostas de serviço com resultado vindas da Invocação de Programa controlada são recebidas pela Invocação de Programa controladora e incorporada a seus resultados. O *Companion Standard* define alguns atributos adicionais a fim de arbitrar esta relação entre Invocações de Programa controladora e controlada.

O VMD do Robô necessita de três objetos Invocação de Programa, o primeiro está permanentemente associado ao Domínio R_ARM e tem por finalidade o controle direto do braço do robô. Outro objeto Invocação de Programa, está associado ao Domínio R_CAL e controla a execução do procedimento de calibração das junções. A última Invocação de Programa deve ser carregada dinamicamente pelo usuário, para controlar a execução dos procedimentos de movimentação. Ela está associada não somente ao Domínio R_TRANS mas também ao Domínio R_SAFE, garantindo assim a segurança do equipamento durante um procedimento de movimentação.

O objeto Invocação de Programa **TRANS**, por ser responsável pelos procedimentos de transporte, deverá relacionar-se com o objeto Invocação de Programa **R_ARM**. Neste sentido, a Invocação de Programa TRANS deverá controlar a Invocação de Programa R_ARM de forma que os movimentos do robô correspondam aos movimentos programados no procedimento de transporte (TRANS).

Objetos Tipos Nomeados

São padronizados quatro Tipos Nomeados para suportar estruturas de dados específicas do robô. Estes objetos são considerados estáticos.

R_PIS - tipo para coordenadas de posição no espaço;
R_OIS - tipo para orientação no espaço;
R_PSE - concatenação de R_PIS e R_OIS, usado como tipo para todos os sistemas de coordenadas;
R_EEF - tipo garra do robô;

Objetos Variáveis Nomeadas

Estes objetos são pré-definidos no robô e utilizados para verificação do estado e controle deste dispositivo:

R_VPWR - indica o valor do atributo *Any Physical Resource Power On* do VMD e assume o valor *true* se a energia está aplicada.
R_VUOM - indica a unidade de medida, assume *true* para milímetros e *false* para polegadas.
R_VCAL - contém o valor do atributo *All Phisical Resources Calibrated* do VMD.
R_VLOCAL - contém o valor do atributo *Local Control* do VMD.

Objetos da Classe Evento

O *Companion Standard* do robô padroniza, numa configuração mínima, a existência de quatro objetos Condição de Evento para a sinalização de eventos no dispositivo robô:

R_RVS - será pré-definido no VMD com a finalidade de indicar o fim da execução de um procedimento de movimentação do robô (saída do estado *RUNNING* do VMD).

R_SIV - será pré-definido no VMD para indicar a ocorrência de um problema de segurança (transição de *false* para *true* do atributo *Safety Interlocks Violated* do VMD).

R_RLC - será pré-definido no VMD para indicar a ocorrência de mudança no modo de operação (transição do atributo *Local Control* do VMD).

R_ARM - será usada como monitor do Objeto Invocação de Programa R_ARM (saída do estado *RUNNING*) e será pré-definido no VMD.

O *Companion Standard* do robô padroniza ainda, dois objetos Ação de Evento:

R_ARM - que indica que os valores dos atributos do Objeto Invocação de Programa R_ARM devem ser enviados na notificação (através do Serviço MMS *Get_Program_Invocation_Attributes*), quando esta invocação deixar o estado *RUNNING*.

R_STC - que provê uma notificação de evento (através do uso do Serviço MMS *Status*) quando qualquer um das condições de evento definidas anteriormente ocorre (com exceção de R_ARM).

Apesar de não existir padronização para nenhum objeto Registro de Evento, o usuário desta aplicação deverá criar um objeto desta classe para cada condição de evento, a fim de identificar a Associação de Aplicação pela qual será enviada a notificação do evento.

Objeto VMD

A forma padrão para o objeto VMD do robô fica sendo então a seguinte:

Object: VMD
Vendor Name = (específico de implementação)
Model Name = (específico de implementação)
Revision = (específico de implementação)
Logical Status = {STATE_CHANGES_ALLOWED,
NO_STATE_CHANGES_ALLOWED,
LIMITED_SERVICES_SUPPORTED}
List of Capabilities = (específico de implementação)
Physical Status = {OPERATIONAL, PARTIALLY_OPERATIONAL,
INOPERABLE, NEEDS_COMISSIONING}
List of Program Invocations = [TRANS, R_ARM, R_CAL]
List of Domains = [TRANS, R_ARM, R_CAL, R_SAFE]
List of Transaction Objects = (criado nos pedidos de
serviços confirmados)
List of Upload State Machine = EMPTY (não está previsto
upload nesta aplicação)
Lists of Other VMD-Specific Objects = [
Named Type: [R_PIS,R_OIS,R_PSE,R_EEF]

Event Action: [R_STC,R_ARM]
Variable Named: [R_VPWR,R_VUOM,R_VAL,R_VLOCAL]
Event Condition: [R_RVS,R_SIV,R_RLC,R_ARM,CAL]
Event Enrollment: [R_RVS,R_SIV,R_RLC,R_ARM,CAL]]
Safety Interlocks Violed = {TRUE, FALSE}
Robot VMD State = {ROBOT_IDLE, ROBOT_EXECUTING,
ROBOT_READY, ROBOT_LOADED,
ROBOT_PAUSED,
MANUAL_INTERVENTION_REQUIRED}
Any Phisical Resourse Power On = {TRUE, FALSE}
All Phisical Resourse Calibrated = {TRUE, FALSE})
Local Control = {TRUE, FALSE}
Unit of Mesure = MILLIMETERS

O atributo *adicional Robot VMD State* expressa o estado atual do robô, e depende do estado das Invocações de Programa que controlam o movimento do braço do robô (R_TRANS e R_ARM). A figura 4.5 mostra uma representação gráfica do VMD do robô.

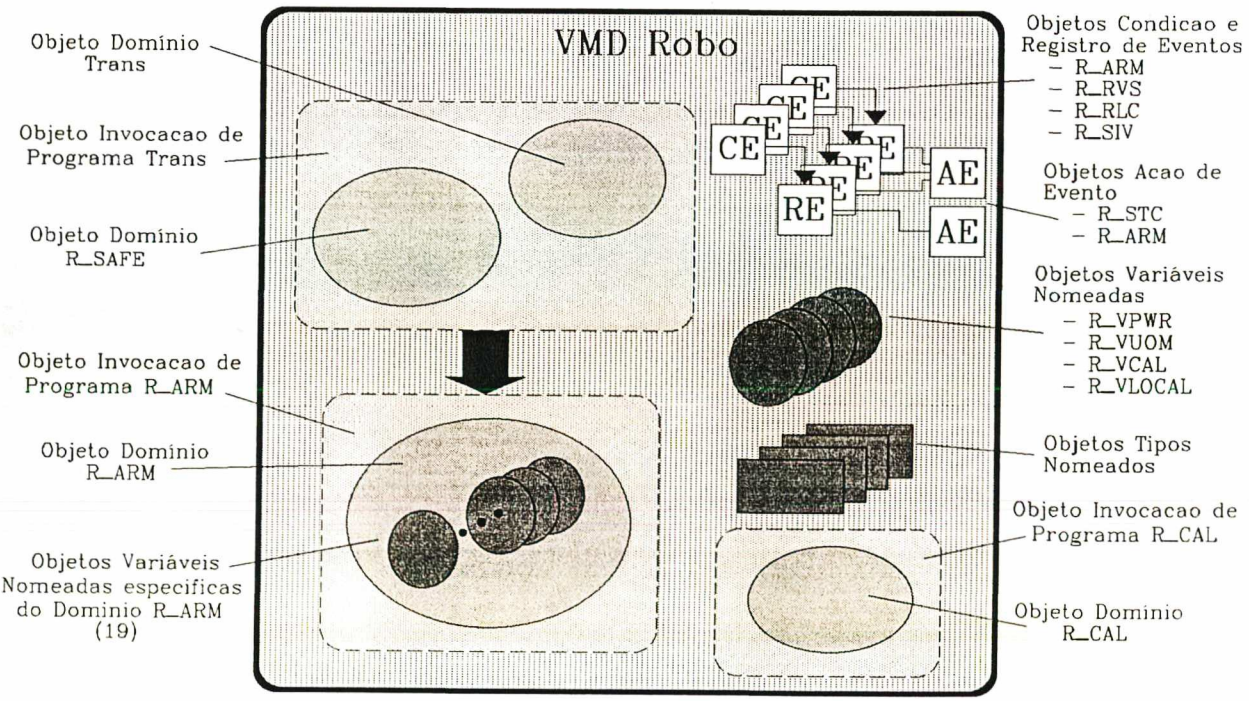


Fig. 4.5 VMD do Robô

III. Definição dos Objetos MMS da Mesa Posicionadora

As operações da mesa devem ser efetuadas a partir de chamada de procedimentos remotos (classe *Interlocked Control* definido no *Companion Standard* para CPs [IEC 89]).

Nesta configuração, um cliente pede ao servidor a execução de uma operação de aplicação, que após executá-la, informa ao cliente o resultado da operação.

Há dois aspectos importantes quanto ao uso da classe *Interlocked Control*; a sincronização dos programas aplicação cliente e servidor, e a troca de dados entre eles (que ocorre no ponto de sincronização). O *Companion Standard* para CPs define o objeto adicional *Gate* e o serviço *Data_Exchange* para tratar destes aspectos.

Objetos Domínio e Invocação de Programa

As funções básicas de movimentação da mesa estarão estaticamente alocadas na memória do controlador programável, e serão mapeadas em um objeto Domínio estático. Desta forma, o supervisor ativará a mesa através de um determinado objeto Invocação de Programa. O fato do objeto Domínio estar alocado estaticamente, se deve à característica dedicada de suas funções, que não necessitam ser alteradas durante o processo de produção. Segundo o *Companion Standard* os objetos Invocação de Programa são obrigatoriamente dinâmicos.

O atributo *List of Subordinate Objects* contém as referências para os objetos *Gate* que por sua vez implementarão as chamadas de procedimento remoto. Cada um dos comandos da mesa será considerado uma subrotina do domínio **CP_MESA**, acessada individualmente por meio do respectivo objeto *Gate*.

Objeto Gate

Cada operação básica da mesa será representada por um objeto *Gate*, que por sua vez será subordinado ao objeto Domínio CP_MESA.

O pedido de serviço *Data_Exchange* ativará um dos objetos Gate e conterá os parâmetros de posição origem e posição destino, conforme definição na descrição funcional. No final da execução será retornada a resposta do serviço *Data_Exchange*, indicando o sucesso ou fracasso da operação.

Durante a implementação, o objeto *Gate* e o serviço *Data_Exchange* deverão ser mapeados nos blocos funcionais *SEND* e *RCV*, que são especificados pelos Controladores Programáveis para execução da função de *Interlock control*.

Objeto VMD

O objeto VMD-CP da mesa posicionadora assume, então a seguinte estrutura:

Object: VMD
Vendor Name = (específico de implementação)
Model Name = (específico de implementação)
Revision = (específico de implementação)
Logical Status = {STATE_CHANGES_ALLOWED,
NO_STATE_CHANGES-ALLOWED,
LIMITED_SERVICES_SUPPORTED}
List of Capabilities = (específico de implementação)
Physical Status = {OPERATIONAL, PARTIALLY_OPERATIONAL,
INOPERABLE, NEEDS_COMMISSIONING}
List of Program Invocations = [CP_MESA]
List of Domains = [CP_MESA]
List of Transaction Objects = (criado nos pedidos de
serviços confirmados)
List of Upload State Machine = EMPTY
Lists of Other VMD-Specific Objects = EMPTY
PC Status = (descreve o estado do CP e de seus sub-sistemas)

A figura 4.6 mostra a representação gráfica do VMD do CP da mesa posicionadora.

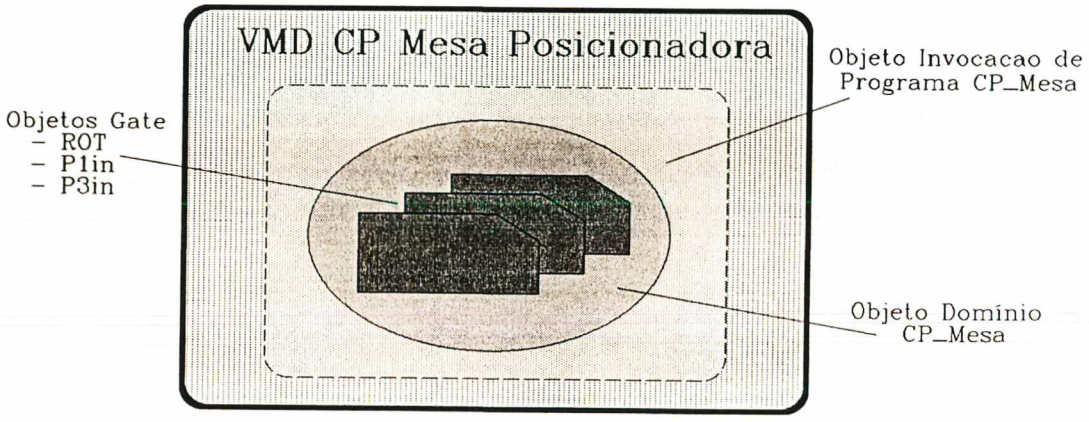


Fig. 4.6 VMD da Mesa Posicionadora

4.3.4 Definição das Entidades de Aplicação (AEs)

As características de comunicação para cada um dos dispositivos da célula são modeladas pelas AEs instanciadas em cada AP da célula flexível. Considerou-se que as comunicações são baseados apenas em associação explícita.

A figura 4.7 mostra as instâncias de AE a serem ativadas em cada um dos APs da CFU. Deve-se notar pontos seguintes:

- As AE-Is local e remota, devem obrigatoriamente possuir o mesmo AE-Tipo;

- A utilização de uma arquitetura Mini-MAP, acarreta na alocação prévia dos LSAPs para cada instância de AE;

- Os APs cliente e servidor deverão possuir tantos AE-Tipos quantos forem os diferentes dispositivos remotos com os quais necessitam estabelecer comunicação.

- Algumas AE-Is estão ligadas a um VMD e outras por serem usadas pelo modo cliente não necessitam estar ligadas logicamente a este tipo de modelo.

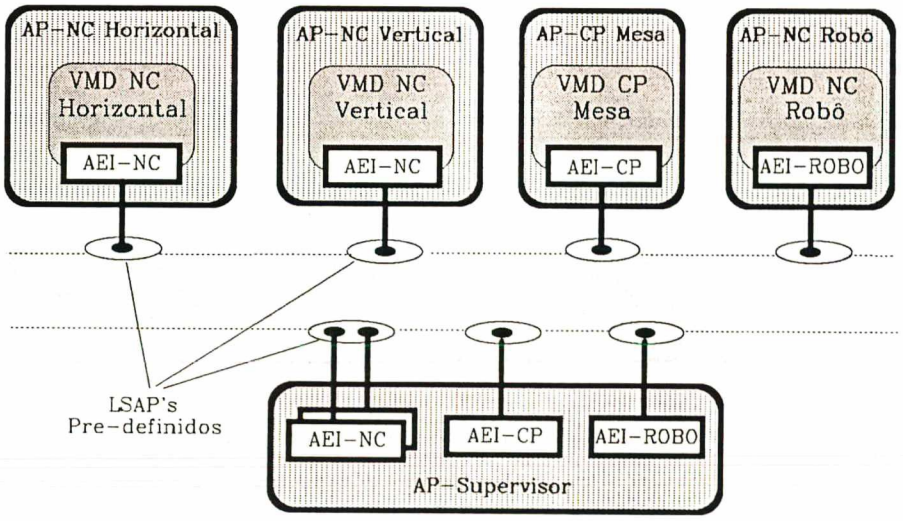


Fig. 4.7 AE-Is no Ambiente da CFU

4.4 Etapa 3: Definição das Interações entre os APs

Nesta etapa, é necessário definir a forma de utilização dos Serviços MMS para as etapas de inicialização do sistema de comunicação, inicialização dos dispositivos e funcionamento normal da aplicação.

4.4.1 Estabelecimento dos Ambientes MMS

Durante a inicialização do sistema de comunição, deverão ser estabelecidas as associações de aplicação entre cada equipamento e o supervisor. O ambiente de comunicação completo da CFU, incluindo as Associações Explícitas estabelecidas, é apresentado na figura 4.8.

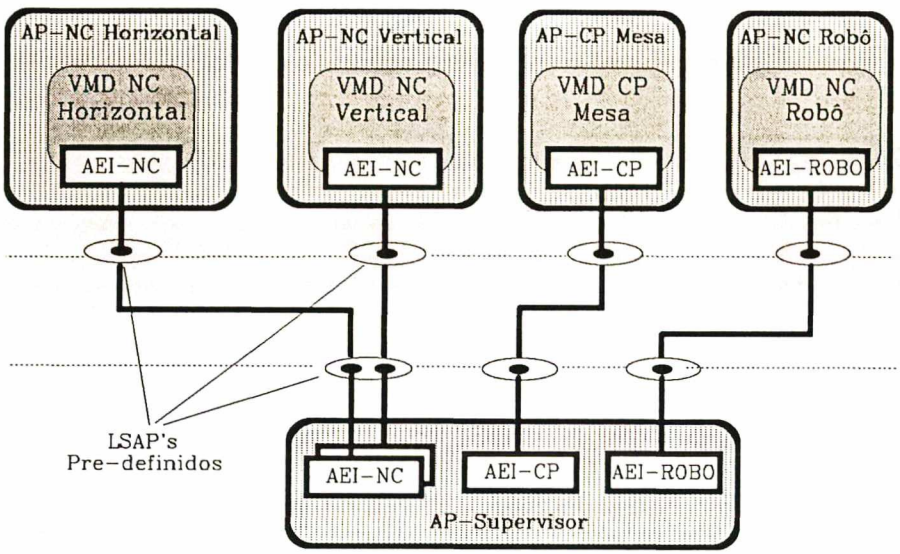


Fig. 4.8 Ambiente de Comunicação Completo da CFU

Estas associações servirão como suporte lógico para as demais trocas de informação tais como inicializações, transmissões de comando, carregamento de programas, e supervisão das etapas do processo de produção.

Após esta etapa, todos os APs estarão ligados através de Ambientes MMS, e desta forma já podem fazer uso dos Serviços MMS para os procedimentos de comunicação na célula.

4.4.2 Inicialização dos Dispositivos Servidores

Este procedimento é individual para cada dispositivo e inicia com a verificação de operacionalidade (através do serviço MMS *status*).

A seguir serão descritos os procedimentos básicos, por parte do supervisor, para inicialização de cada equipamento. Foi considerado que o nome de todos os objetos pré-

definidos no dispositivo já são do conhecimento do usuário cliente, sendo assim não é necessário verificar sua existência.

I. NCs dos Centros de usinagem

- Criar todos os objetos Registro de Evento necessários, através do serviço MMS *Define_Event_Enrollment.*

- Atribuir à variável *N_ControlLocal* o valor *false* (serviço MMS *Write*), para colocar o NC em modo de operação remoto.

- Atribuir à variável *N_MachinePower* o valor *true* (serviço MMS *Write*), para energizar o equipamento.

- Carregar o Domínio N_SPD_Med responsável pela armazenamento dos dados da medição *in-process*. Estes objetos serão compartilhados por todos os programas-peça. O carregamento destes domínios é feito através dos serviços de sequência de carregamento (*Initiate_Down_Load_Sequence*, *Down_Load Segment* e *Terminate_Down_Load_Sequence*).

II. CP da Mesa posicionadora

- Criar a invocação de programa CP_MESA responsável pela inicialização da mesa posicionadora. Neste caso usa-se o serviço *Create_Program_Invocation*

- Executar o procedimento CP_MESA para colocar a mesa posicionadora em operação (à espera das chamadas remotas de procedimentos). Isto é feito através de um serviço *Start* sobre o objeto Invocação de Programa CP_MESA.

III. NC do Robô

- Criar todos os objetos Registro de Evento necessários.

- Atribuir à variável *R_VLOCAL* o valor *false*, para colocar o NC do robô em modo de operação remota.

- Atribuir à variável *R_VUOM* o valor *true*, indicando que a unidade de medida usada será centímetros.

- Ativar o procedimento de auto-calibração usando o serviço *Start* sobre o objeto Invocação de Programa R_CAL.

- Carregar o Domínio contendo o código do programa de transporte TRANS.

- Criar a Invocação de Programa TRANS.

- Relacionar a Invocação de Programa TRANS com a Invocação de Programa R_ARM para permitir que o procedimento de transporte tenha acesso às rotinas de movimentação do braço do robô. Esta relação é conseguida através do serviço *Select*.

4.4.3 Funcionamento normal da Aplicação

Após ter sido inicializada, a célula estará pronta para entrar na fase de produção. Nesta fase, o supervisor poderá controlar todas as operações, tomando como base os eventos ocorridos e as especificações do escalonamento de produção.

O sequenciamento das primitivas de comunicação, que implementam as operações do processo de produção, é apresentado de maneira isolada, como se a operação fosse única sem levar em conta as possibilidades de entrelaçamento entre as primitivas de várias operações, quando de um funcionamento global.

A seguir serão apresentados os procedimentos tomados em cada operação, tomando como base as funções executadas por cada um dos dispositivos que trabalham no modo cliente, já descritas anteriormente.

I. Colocação de Peça Bruta na Mesa Posicionadora

A partir da definição de uma posição vazia (x) da mesa e da origem (y) da peça a ser carregada nele, o supervisor gera uma sequência de primitivas de serviços para executar a operação de carregamento de uma peça bruta na mesa posicionadora. Esta sequência de primitivas pode ser vista no diagrama de tempo mostrado na figura 4.9.

O supervisor enviará inicialmente um pedido ao VMD da mesa para que esta rotacione a posição vazia (x) até a posição de entrada/saída (P1'). O uso do serviço *Data_Exchange* garante o reconhecimento da operação ao final de mesma. Após o sucesso desta operação, é a vez do robô ser ativado (através da Invocação de Programa R_TRANS) para transportar a peça da posição de origem até a entrada da mesa. Como parâmetro do serviço *Start* serão enviadas a posição de origem, a posição destino (mesa posicionadora) e o tipo da peça. O supervisor será informado quanto ao sucesso ou não do procedimento de transporte através de uma notificação de evento (R_RVS) enviada pelo

VMD do robô. Por fim, o supervisor volta a enviar uma primitiva de serviço *Data_Exchange*, desta vez contendo o pedido de inserção do *pallet* recém carregado.

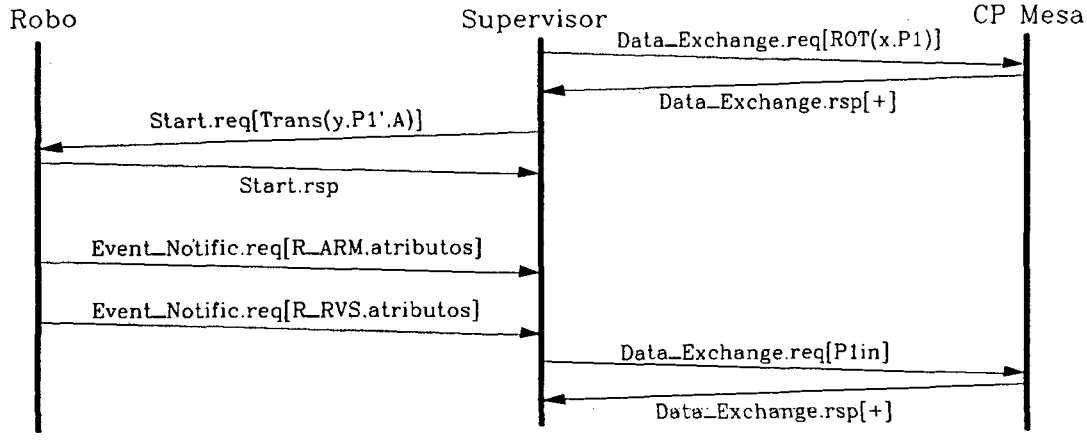


Fig. 4.9 Colocação de Peça na Mesa

II. Retirada de Peça Usinada da Mesa Posicionadora

O supervisor definirá qual peça usinada da mesa (posição x) será retirada e levada ao local de destino (y). Esta decisão deve obedecer aos critérios do escalonamento imposto ao supervisor. A sequência de primitivas usada nesta operação é idêntica àquela da operação anterior.

III. Colocação de Peça Bruta no Centro Vertical

O supervisor, a partir da configuração de peças na mesa e das especificações de escalonamento, definirá qual peça bruta contida na mesa posicionadora (posição x) será colocada na posição P3 (posição de entrada do centro de usinagem vertical). O uso das primitivas de serviço MMS para executar esta tarefa pode ser visto no diagrama de tempo da figura 4.10.

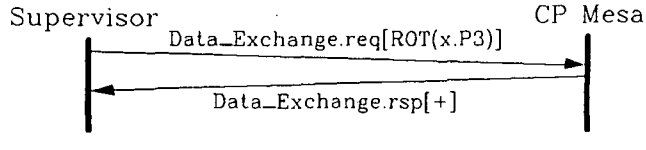


Fig. 4.10 Colocação de Peça
no Centro Vertical

O supervisor terá basicamente que enviar uma primitiva de serviço *Data_Exchange* contendo o pedido de rotação da mesa Px até a posição P3. Ao final desta operação a peça estará posicionada na entrada do centro de usinagem, pronta para ser usinada.

IV. Retirada de Peça Usinada do Centro Vertical

Neste caso, o supervisor deverá ser informado acerca da ocorrência de um evento de fim de usinagem (N_EOP), acusado pelo VMD do centro de usinagem vertical. Após este evento, o supervisor, sabendo a posição da mesa referente ao *pallet* em usinagem (posição Px), deverá enviar à mesa o pedido de rotação, a fim de colocar a posição Px diante da entrada do centro de usinagem e inserir o *pallet* na mesa. O uso das primitivas de serviço MMS para executar esta tarefa é visto no diagrama de tempo da figura 4.11.

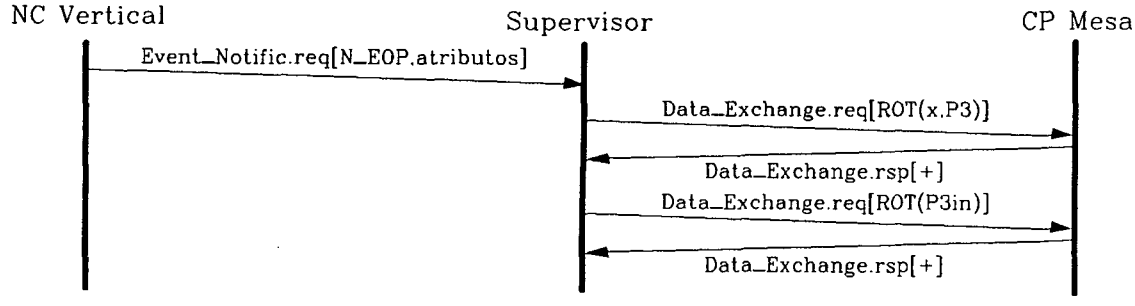


Fig. 4.11 Retirada de Peça
do Centro Vertical

V. Colocação de Peça Bruta no Centro Horizontal

A origem da peça poderá ser tanto o armazém de peças brutas quanto a mesa posicionadora. O supervisor, neste caso, precisará apenas enviar uma primitiva de serviço ao VMD do robô, ativando o procedimento de transporte. Como parâmetro do serviço *Start* serão enviados a posição de origem, a posição destino (centro horizontal) e o tipo da peça. A confirmação será feita por notificação de evento. O uso das primitivas de serviço para executar esta tarefa são vistas no diagrama de tempo da figura 4.12.

Caso a peça tenha como origem a mesa posicionadora, uma operação de saída de peça da mesa posicionadora deverá ter sido feita previamente.

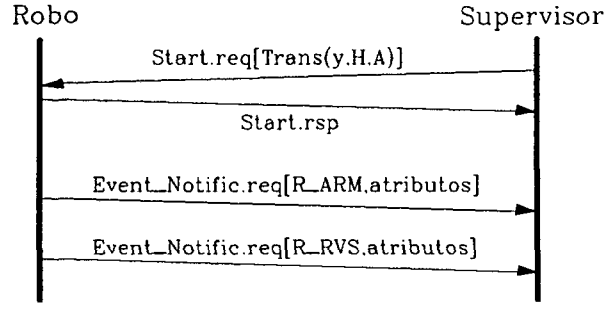


Fig. 4.12 Colocação de Peça
no Centro Horizontal

VI. Retirada de Peça Usinada do Centro Horizontal

O supervisor deverá ser informado acerca da ocorrência de um evento de fim de usinagem (N_EOP), acusado pelo VMD do centro de usinagem horizontal. Após este evento, o supervisor poderá proceder a retirada da peça usinada, enviando uma primitiva de serviço ao VMD do robô, a fim de ativar o procedimento de transporte. Como parâmetro do serviço *Start* será enviada a posição de origem (centro horizontal), a posição destino e o tipo da peça. O uso das primitivas de serviço MMS para executar esta tarefa é visto no diagrama de tempo da figura 4.13.

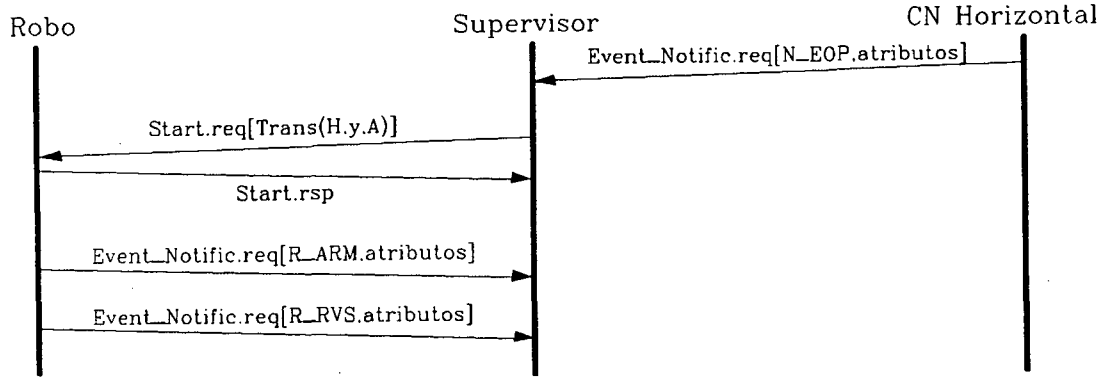


Fig. 4.13 Retirada de Peça
do Centro Horizontal

Caso a peça tenha como destino a mesa posicionadora, este procedimento não será ativado, ficando a peça a espera de uma operação de colocação de peça na mesa posicionadora.

VII. Usinagem

O supervisor definirá que peça será usinada, a partir das especificações do escalonamento. O procedimento de usinagem requer que todos os recursos tais como peça bruta, ferramentas adequadas e programa-peça sejam providos previamente. Neste sentido existem, como já foi visto, três situações possíveis:

a. Primeira usinagem de um tipo de peça

Neste caso, é necessário carregar todos os domínios de programas e dados necessários ao procedimento de usinagem de um tipo de peça (a título de exemplo, será considerada uma peça do tipo "A"), que são:

- Objeto Domínio N_PRG_A;

- Objeto Domínio N_TLD_A;

- Objeto Invocação Programa N_ACT_A.

Os objetos Domínio são fornecidos ao VMD do centro de usinagem através dos serviços de sequência de carregamento (*Initiate_Down_Load_Sequence*, *Down Load_Segmente* e *Terminate_Down_Load_Sequence*). O objeto Invocação de Programa é provido através do serviço *Create_Program_Invocation*.

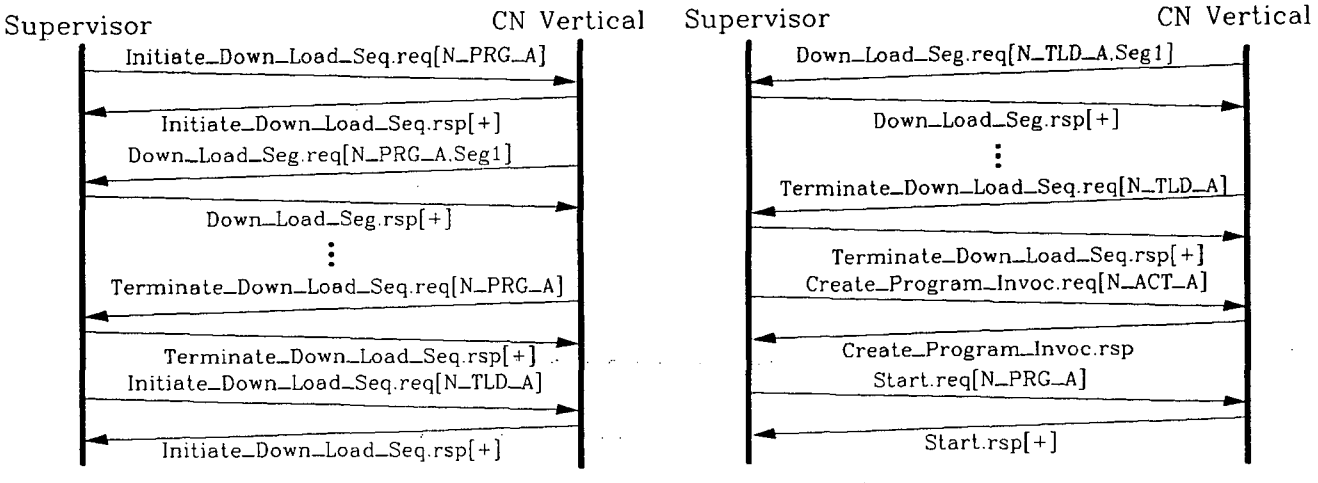


Fig. 4.14 Primeira Usinagem

O uso das primitivas de serviço para executar esta operação é visto no diagrama de tempo da figura 4.14.

Como procedimento local, algumas informações contidas no Domínio N_TLD_A devem ser enviadas à estação de operador, para que este proceda a colocação das ferramentas no armazém. Após a colocação das ferramentas, o operador deve informar este fato (através da estação de operador) ao NC, indicando que este pode continuar o processo de usinagem.

Durante toda a operação com os centros de usinagem, todos os eventos significativos (como falhas no equipamento) serão informados ao supervisor.

b. Usinagem de peça do mesmo tipo

Após o encerramento de um processo de usinagem, o supervisor deverá ler o Domínio N_TLD_A (feito através dos serviços *Initiate_Up_Load_Sequence*, *Up Load_Segment* e *Terminate_Up_Load_Sequence*), cujos dados podem ter sido atualizados durante os ciclos de medição das ferramentas. Estes dados deverão manter o arquivo de informações das ferramentas atualizado, além de indicar a possível necessidade de troca de alguma ferramenta (diagrama de tempo da figura 4.15).

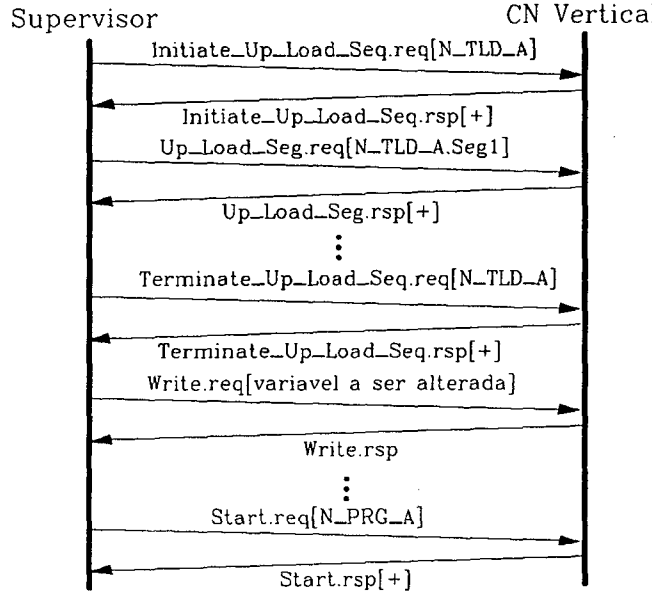


Fig. 4.15 Próxima Usinagem
(Peça do Mesmo Tipo)

Neste caso, normalmente não há necessidade de troca do programa-peça armazenado e a troca de parâmetros do programa (detalhamento de um tipo de peça, mudança da velocidade de corte, etc), pode ser feita através de escrita em variáveis nomeadas (serviço *write*).

c. Mudança do tipo de peça a usinar

Caso os objetos necessários ao procedimento de usinagem da peça anterior sejam ainda necessários e haja espaço para definir os novos objetos, estes poderão coexistir, caso contrário, os primeiros devem ser apagados (utilizando o serviço *Delete_Domain*). Antes de ser apagado, porém, o Domínio N_TLD_ atual deverá ser lido como no caso anterior.

O uso das primitivas de serviço para executar a tarefa de usinagem neste caso é visto no diagrama de tempo da figura 4.16.

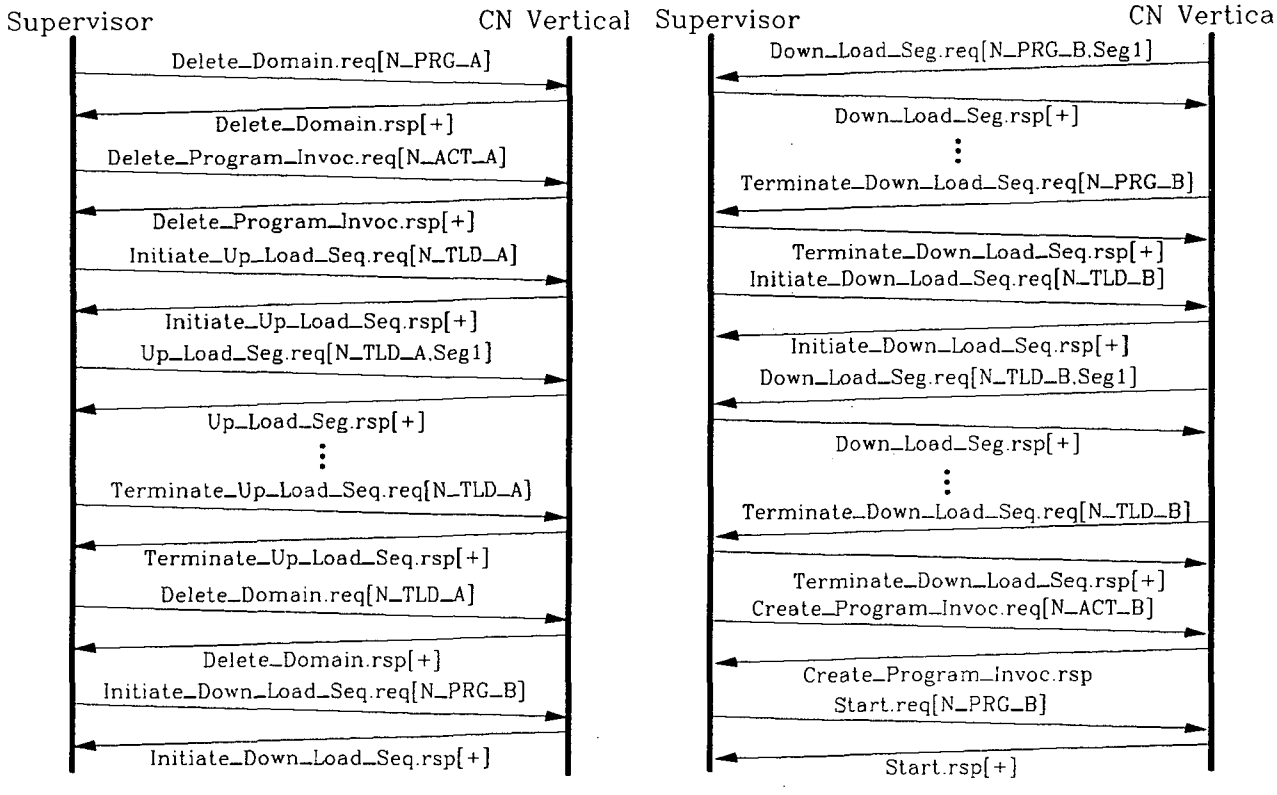


Fig. 4.16 Próxima Usinagem
(Tipo de Peça Diferente)

4.5 Etapa 4: Descrição Formal e Validação dos APs

Antes de partir para a implementação dos elementos que compõem o sistema de comunicação, deve-se certificar que este sistema está livre de erros e que atende às funcionalidades exigidas pela aplicação do usuário.

Para a especificação formal do conjunto de elementos de protocolo do sistema de comunicação da CFU, utilizou-se a linguagem de especificação Estelle. A estrutura da especificação comporta, na forma de módulos, os elementos definidos pelo Padrão MMS para a comunicação neste tipo de aplicação (as decisões tomadas no mapeamento de cada um destes elementos na linguagem de especificação Estelle são apresentadas no anexo D).

Durante a inicialização estática, são instanciados os módulos AP_Servidor que representam os Processos de Aplicação MMS, definidos para cada equipamento da célula; e o módulo AP_Cliente, que representa o Processo de Aplicação MMS, executado no Supervisor. Também será instanciado um módulo Camada_Enlace, cuja finalidade será a de encapsular as funcionalidades das camadas inferiores da arquitetura Mini-MAP. A figura 4.17 apresenta a estrutura da especificação em seu nível de abstração mais alto.

Os módulos AP_Servidor, são encarregados de instanciar os módulos AE e os módulos VMD para cada dispositivo da célula flexível. O número e as especificações dos módulos AE e VMD para cada dispositivo são baseados nos resultados obtidos com o uso da metodologia (figura 4.8).

O módulo AP_Cliente é responsável pela instanciação dos módulos AE e do módulo Prog_Aplic, que por sua vez é responsável pelo gerenciamento das operações definidas para o funcionamento da célula flexível.

O Módulo Camada_Enlace apenas encapsula as funcionalidades da camada de enlace da arquitetura Mini-MAP, fornecendo os pontos de acesso aos serviços de enlace (LSAPs) para cada um dos módulos AP.

O objetivo desta etapa do trabalho foi o de validar a utilização do protocolo MMS para a comunicação dentro da Célula Flexível de Usinagem, verificando se os elementos de protocolo escolhidos atendem a todas as funcionalidades requeridas pela aplicação. Esta verificação do Sistema de Comunicação foi realizada a partir da simulação de sua especificação Estelle, utilizando o Simulador ESTIM [Saqui 91].

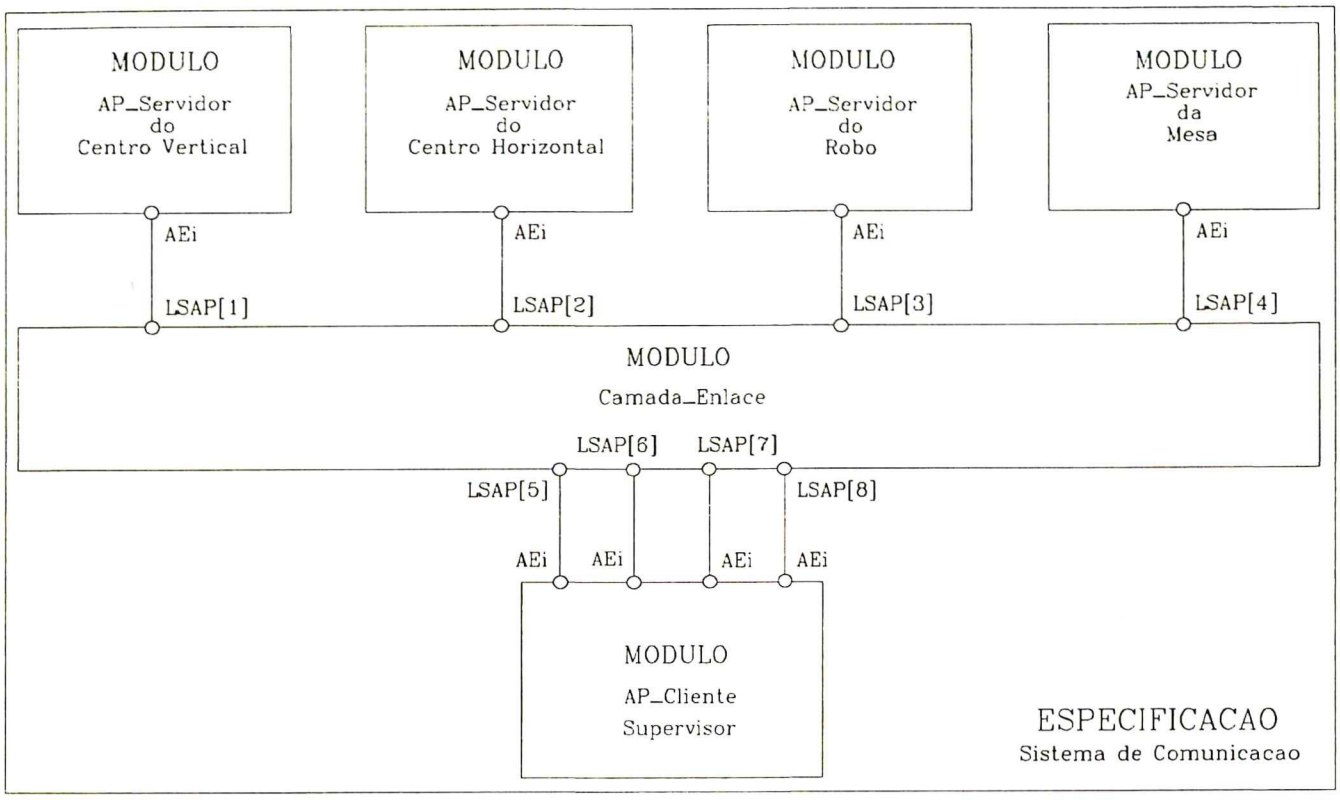


Fig. 4.17 Especificação Estelle do Sistema
de Comunicação da CFU

Devido às restrições de memória do ambiente de simulação, a verificação da especificação foi realizada sobre parte dos dispositivos integrantes da célula. Desta forma, decidiu-se verificar a comunicação entre o Supervisor, o Centro de Usinagem Horizontal e o Robô, cobrindo assim parte significativa da célula. A figura 4.18 mostra a estrutura da especificação Estelle simulada (os cenários usados durante a simulação podem ser vistos no anexo E).

A simulação, mesmo parcial, permitiu verificar o funcionamento adequado das entidades que compõem o sistema de comunicação a ser implementado. Em particular no que diz respeito:

- à escolha e utilização dos serviços MMS para desempenhar os aspéctos de comunicação necessários às operações básicas da célula;

- ao modelamento dos dispositivos existentes na célula, no que diz respeito a interação mútua;

- à sincronização entre os objetos que fazem parte de um VMD.

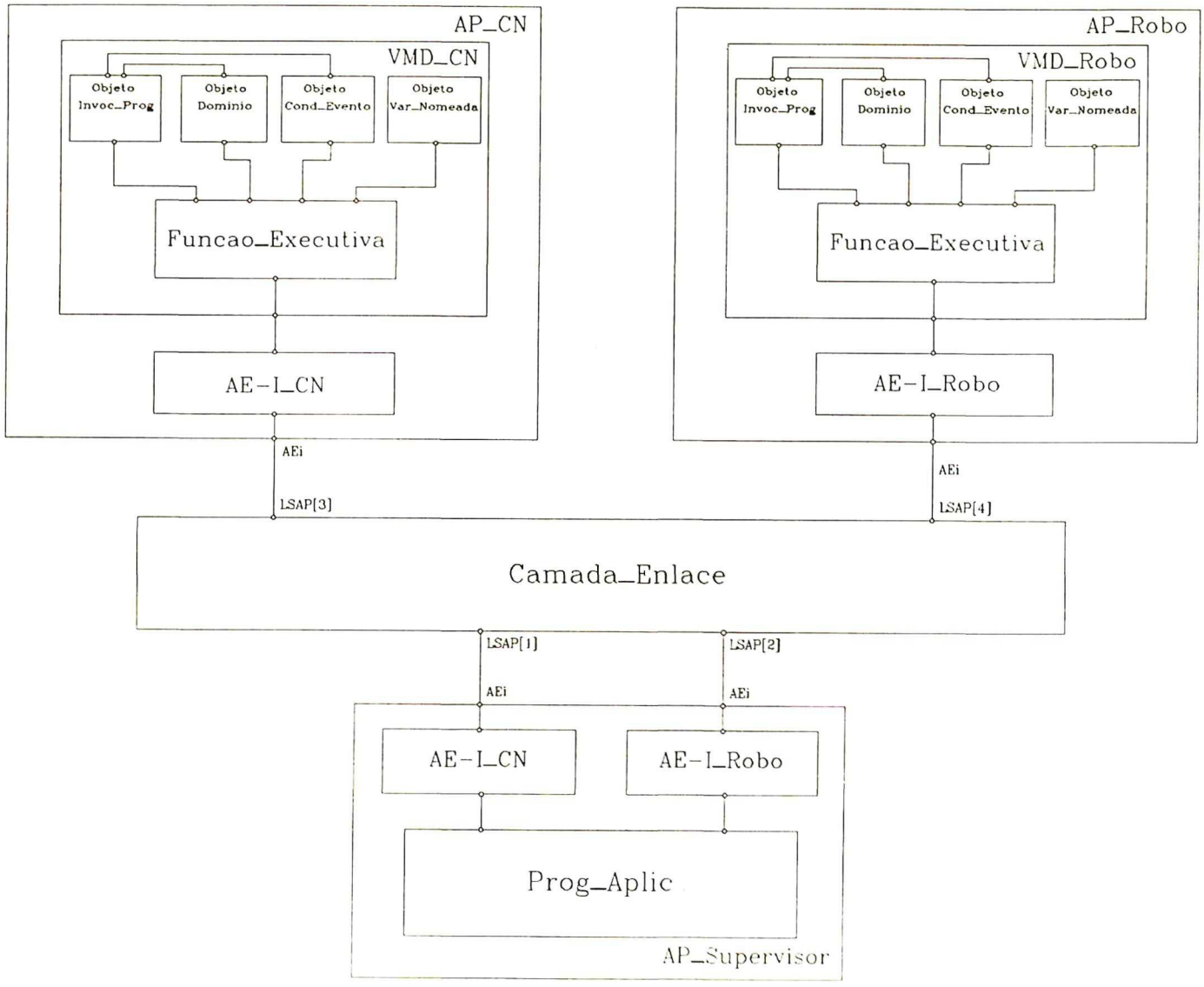


Fig. 4.18 Parte da Especificação Estelle do Sistema de Comunicação da CFU usada na Simulação

4.6 Etapa 5: Implementação dos APs

A partir da certeza de uma especificação correta, pode-se então passar à fase de implementação do sistema de comunicação da Célula Flexível de Usinagem.

Este trabalho, porém, não chegou a desenvolver tal etapa, limitando-se à especificação formal e à validação desta especificação.

4.7 Conclusão

Neste capítulo tomou-se como base a metodologia proposta no capítulo anterior para aplicar o Padrão MMS na comunicação existente entre os dispositivos que compõem uma célula flexível destinada à usinagem de peças prismáticas.

O comportamento esperado da célula flexível foi especificado e validado, permitindo uma descrição mais detalhada e correta das tarefas a serem desempenhadas pela célula. Durante a simulação do modelo comportamental da célula, tarefas como as de usinagem, movimentação da mesa e do robô puderam ser observadas em sincronismo dentro de uma sessão de produção

As definições dos objetos e serviços MMS usados para cumprimento das tarefas atribuidas à célula também foram motivo de validação. Desta vez, parte do sistema de comunicação da célula foi simulado com o objetivo de validar o uso das primitivas de serviços e dos objetos MMS escolhidos.

CAPÍTULO 5

CONCLUSÃO

Este trabalho apresentou uma metodologia capaz de guiar um implementador na tarefa de construção de um ambiente de comunicação baseado no padrão MMS, para uma aplicação fabril qualquer. A partir da especificação funcional da aplicação e dos requisitos de comunicação, esta metodologia define os Processos de Aplicação (APs) seguindo uma abordagem de refinamentos sucessivos e as interações entre estes APs através dos Serviços MMS. A validação, etapa que também faz parte da metodologia, permite a análise da especificação do sistema de comunicação, antes de passar à fase de implementação. Desta forma, a metodologia permite implementar o sistema de comunicação destas aplicações de forma correta.

A fim de automatizar as fases de especificação formal e de validação, foi proposta ainda, uma ferramenta de simulação baseada numa biblioteca de módulos Estelle reconfiguráveis, dedicados à simulação da utilização dos Serviços MMS em tais aplicações. Esta ferramenta é de especial importância na fase de escolha dos Serviços MMS usados no sistema de comunicação, visto que através dela o usuário pode testar o sistema de comunicação e escolher o mais adequado à aplicação dada.

As operações da célula foram simuladas a partir de uma especificação Estelle, permitindo entender o comportamento da célula e inferir a respeito dos requisitos de comunicação nesta.

O comportamento esperado da célula flexível foi especificado e validado através de simulação, permitindo uma descrição mais detalhada e correta das operações a serem desempenhadas pela célula. Durante a simulação do modelo comportamental da célula, tarefas como as de usinagem, movimentação da mesa e do robô puderam ser observadas em conjunto dentro de um processo de produção.

A especificação formal e a simulação dos elementos que formam o sistema de comunicação definido para esta aplicação também foram mostrados. Devido às restrições de memória do ambiente de simulação, a verificação da especificação foi realizada sobre

parte dos dispositivos integrantes da célula. Desta forma, foi somente analisada a comunicação entre o Supervisor, o Centro de Usinagem Horizontal e o Robô, cobrindo assim parte significativa da célula.

A simulação, mesmo parcial, permitiu verificar o funcionamento adequado das entidades que compõem o sistema de comunicação especificado, em particular no que diz respeito ao mapeamento dos objetos e serviços MMS que atendem aos requisitos de comunicação do exemplo proposto.

As perspectivas de continuação deste trabalho apontam para o aprimoramento e implementação da ferramenta de simulação proposta, bem como seu teste e a aplicação em outras aplicações fabris.

BIBLIOGRAFIA

[Azema 85] Azema, P. Papapanagiotakis, G. "Protocol Analysis by using Predicate Nets". 5th Int. Worshop on Protocol Specification, Verification and Testing. France. June, 1985.

[Bochmann 89] Bochmann G.V. "Protocol Specification for OSI". Computer Networks and ISDN Systems, Nº 18. North-Holland. 1989.

[Budkowski 87] Budkowski, S. & Dembinski, P. "An Introduction to Estelle: A Specification Language for Distributed Systéms". Computer Networks and ISDN Systems, Nº 14. North-Holland. 1987.

[CERTI 88] PC-073. "Laboratório CIM - UFSC/CERTI". Fundação Centro Regional de Tecnologia em Informática da Santa Catarina. Agosto, 1988.

[Courtiat 88] Courtiat, J-P & Diaz, M. & Mazzola, V.B. "Estelle*: a Powerful Dialect of Estelle for OSI Protocol Description". In Proceedings of the 8th IFIP Symposium on Protocol Specification, Testing and Verification. Atlantic City. June, 1988.

[Courtiat 91] Courtiat, J-P. "Description Formelle de Protocolles et Services OSI en Estelle et Estelle* Expérience et Méthodologie". Article Invité dans CFIP'91. Pau. Septembre, 1991.

[Fernandez 88] Fernandez, J-C. "ALDEBARAN: un système de vérification par réduction de processus communicantes": Thése de Doctorat, Université Joseph Fourier. Grenoble. May, 1988.

[IEC 89] IEC/SC65A/WG6/TF7. "Programmable Controller Message Specification (Companion Standard to ISO 9506/1-2)". June, 1989

[ISA 90] ISA-dS72.02. "Companion Standard for Process Control". Instrument Society of America. January, 1990.

[ISO 84] ISO 7498. "Information Processing Systems - Opens System Interconnection-Basic Reference Model". 1984

[ISO 87a] ISO 8824. "Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)". 1987.

[ISO 87b] ISO 9074. "Estelle - A formal Description Technique Based on an Extended State Transition Model". 1987.

[ISO 88] ISO 9506/1-2. "Manufacturing Message Specification". Implementation Release. 1988.

BIBLIOGRAFIA

[Azema 85] Azema, P. Papapanagiotakis, G. "Protocol Analysis by using Predicate Nets". 5th Int. Worshop on Protocol Specification, Verification and Testing. France. June, 1985.

[Bochmann 89] Bochmann G.V. "Protocol Specification for OSI". Computer Networks and ISDN Systems, Nº 18. North-Holland. 1989.

[Budkowski 87] Budkowski, S. & Dembinski, P. "An Introduction to Estelle: A Specification Language for Distributed Systéms". Computer Networks and ISDN Systems, Nº 14. North-Holland. 1987.

[CERTI 88] PC-073. "Laboratório CIM - UFSC/CERTI". Fundação Centro Regional de Tecnologia em Informática da Santa Catarina. Agosto, 1988.

[Courtiat 88] Courtiat, J-P & Diaz, M. & Mazzola, V.B. "Estelle*: a Powerful Dialect of Estelle for OSI Protocol Description". In Proceedings of the 8th IFIP Symposium on Protocol Specification, Testing and Verification. Atlantic City. June, 1988.

[Courtiat 91] Courtiat, J-P. "Description Formelle de Protocolles et Services OSI en Estelle et Estelle* Expérience et Méthodologie". Article Invité dans CFIP'91. Pau. Septembre, 1991.

[Fernandez 88] Fernandez, J-C. "ALDEBARAN: un système de vérification par réduction de processus communicantes": Thése de Doctorat, Université Joseph Fourier. Grenoble. May, 1988.

[IEC 89] IEC/SC65A/WG6/TF7. "Programmable Controller Message Specification (Companion Standard to ISO 9506/1-2)". June, 1989

[ISA 90] ISA-dS72.02. "Companion Standard for Process Control". Instrument Society of America. January, 1990.

[ISO 84] ISO 7498. "Information Processing Systems - Opens System Interconnection-Basic Reference Model". 1984

[ISO 87a] ISO 8824. "Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)". 1987.

[ISO 87b] ISO 9074. "Estelle - A formal Description Technique Based on an Extended State Transition Model". 1987.

[ISO 88] ISO 9506/1-2. "Manufacturing Message Specification". Implementation Release. 1988.

[ISO 89a] ISO 9506/4. "Numerical Control of Machines - Numerical Control Message Specification (Companion Standard to ISO 9506/1-2)". April, 1989.

[ISO 89b] ISO 9506/3. "Robots for Manufacturing Environment - Robot Specific Message System (Robot Companion Standard for MMS)". March, 1989.

[Leite 87] Leite,J.R.E & Mendes,M.J. "Protocolos de Aplicação em Redes Locais de Computadores na Automação Industrial (Protocolos MAP/TOP)". 5º Simpósio Brasileiro de Redes de Computadores. São Paulo. 1987.

[Leite 88] Leite,J.R.E. "Aspéctos de Implementação do Protocolo Aplicativo RS-511 do MAP (Protocolo para Automação Industrial)". 3º Congresso Nacional de Automação Industrial. 1988.

[MAP 88] MAP 3.0. "Manufacturing Automation Protocol Specification". Implementation release subject to errata change. 1988.

[McGuffin 87] McGuffin, L.J. et alli. "MAP/TOP IN CIM Distributed Computing". IEEE Network Vol.2 Nº 3. May, 1988.

[Mendes 89] Mendes,M.J."Comunicação fabril e o Projeto MAP/ TOP". IV EBAI, Janeiro, 1989.

[Pinho 88] Pinho, A.N. "Um Estudo da Especificação da Menssagem da Manufatura (MMS) do Protocolo de Comunicação para Ambientes Industriais MAP". Dissertação submetida à Universidade Federal de Santa Catarina para obtenção do grau de Mestre em Engenharia Elétrica. Florianópolis. 1988.

[Saqui 90a] Saqui-Sannes, P. "Prototypage d'un Environnement de Validation de Protocoles : Apllication à l'Approche Estelle". Ph.D. dissertation, Toulouse, April, 1990.

[Saqui 90b] Saqui-Sannes, P. "The ESTIM User Manual for release 4.0 on SUN3 & SUN4 machines. November, 1990.

[Silveira 91] Silveira, J.L. & Willrich, R. & Farines, J-M & Fraga, J.S. "Um Estudo da Utilização e da Implementação do Padrão MMS". 1º Seminário sobre Redes de Comunicação Industrial. São Paulo. Setembro, 1990.

[TOP 88] TOP 3.0. "Technical Office Protocol". Implementation release subject to errata changes. 1988

[Willrich 91] Willrich, R. "Uma Proposta de um Modelo de Implementação de Serviços de Apoio a Aplicações Industriais Segundo o Padrão MMS". Dissertação submetida à Universidade Federal de Santa Catarina para obtenção do grau de Mestre em Engenharia Elétrica. Florianópolis. Março, 1991.

ANEXO A

SERVIÇOS MMS

Os serviços MMS são separados em dez unidades funcionais, cada uma das quais contendo um certo número de serviços que efetuam procedimentos afins. Estas unidades funcionais são listadas abaixo:

- **Gerenciamento de Contexto**: Os Serviços MMS desta unidade permitem a dois Processos de Aplicação negociar, liberar e abortar uma conexão lógica. Além de rejeitar e cancelar comunicações com um usuário MMS.

- **Suporte de VMD**: Estes serviços fornecem a capacidade de requisitar informações acerca de um VMD remoto (lista de nomes, renomeação, etc).

- **Gerenciamento de Domínio**: Permitem que o usuário cliente possa manipular domínios definidos pelo usuário servidor. As sequências *DownLoad* (carregamento) transferem um donínio do cliente para o servidor e a sequência *UpLoad* (arquivamento) na direção oposta.

- **Gerenciamento de Invocação de Programa**: Permitem ao cliente MMS dirigir a execução de programas.

- **Acesso de Variável**: Dão ao usuário MMS cliente a capacidade de acessar variáveis tipadas do VMD'. São definidos cinco classes adicionais de objetos: variável não-nomeada, nomeadas, de acesso distribuído, listas de variáveis nomeadas, tipos nomeados.

- **Gerenciamento de Semáforo**: Permitem a manipulação de um objeto semáforo de forma a controlar, sincronizar e coordenar recursos compartilhados em diversos usuários MMS.

- **Comunicação com o Operador**: Estes serviços fornecem meios para entrada e saida de dados da Estação do Operador.

- **Gerenciamento de Eventos**: Permitem que o usuário cliente defina, gerencie e obtenha informações acerca de um objeto evento em um VMD.

- **Gerenciamento de Jornal**: Permitem o armazenamento e a busca de informações cronológicas em objetos jornal.

- **Gerenciamento de Arquivo**: Permitem a manipulação (abertura, leitura , renomeação, remoção, etc) de arquivos contidos em dispositivos *Filestore* externos ou em servidores especializados.

A figura A.1 e A.2 mostram os Serviços MMS contidos em cada uma destas unidades funcionais.

UNIDADES FUNCIONAIS	SERVICOS MMS	PRIMITIVAS POSSIVEIS
Gerenciamento de Contexto	Initiate	req/ind/rsp/cnf
	Conclude	req/ind/rsp/cnf
	Abort	req/ind
	Cancel	req/ind/rsp/cnf
	Reject	req/ind/rsp/cnf
Suporte de VMD	Status	req/ind/rsp/cnf
	Unsolicited_status	req/ind
	Get_name_list	req/ind/rsp/cnf
	Identify	req/ind/rsp/cnf
	Rename	req/ind/rsp/cnf
Gerenciamento de Dominio	Initiate_down_load_sequence	req/ind/rsp/cnf
	Down_load_segment	req/ind/rsp/cnf
	Terminate_down_load_sequence	req/ind/rsp/cnf
	Initiate_up_load_sequence	req/ind/rsp/cnf
	Upload_segment	req/ind/rsp/cnf
	Terminate_upload_sequence	req/ind/rsp/cnf
	Request_domain_down_load	req/ind/rsp/cnf
	Request_domain_upload	req/ind/rsp/cnf
	Load_domain_content	req/ind/rsp/cnf
	Store_domain_content	req/ind/rsp/cnf
	Delete_domain	req/ind/rsp/cnf
	Get_domain_attribute	req/ind/rsp/cnf
	Domain_file	req/ind/rsp/cnf
Gerenciamento de Invocacao de Programa	Create_program_invocation	req/ind/rsp/cnf
	Delete_program_invocation	req/ind/rsp/cnf
	Start	req/ind/rsp/cnf
	Stop	req/ind/rsp/cnf
	Resume	req/ind/rsp/cnf
	Reset	req/ind/rsp/cnf
	Kill	req/ind/rsp/cnf
	Get_program_invocation_attributes	req/ind/rsp/cnf
Acesso da Variavel	Read	req/ind/rsp/cnf
	Write	req/ind/rsp/cnf
	Information_report	req/ind/rsp/cnf
	Get_variable_access_attributes	req/ind/rsp/cnf
	Delete_named_variable	req/ind/rsp/cnf
	Define_scattered_access_attributes	req/ind/rsp/cnf
	Delete_variable_access	req/ind/rsp/cnf
	Define_name_variable_list	req/ind/rsp/cnf
	Get_named_variable_list_attributes	req/ind/rsp/cnf
	Delete_named_variable_list	req/ind/rsp/cnf
	Define_named_type	req/ind/rsp/cnf
	Get_named_type_attributes	req/ind/rsp/cnf
	Delete_named_type	req/ind/rsp/cnf

Fig. A.1 Serviços MMS

UNIDADES FUNCIONAIS	SERVICOS MMS	PRIMITIVAS POSSIVEIS
Gerenciamento de Semaforo	Take_control	req/ind/rsp/cnf
	Relinquish_control	req/ind/rsp/cnf
	Define_semaphore	req/ind/rsp/cnf
	Delete_semaphore	req/ind/rsp/cnf
	Report_semaphore_status	req/ind/rsp/cnf
	Report_pool_semaphore_status	req/ind/rsp/cnf
	Report_semaphore_entry_status	req/ind/rsp/cnf
Comunicacao de Operadores	Input	req/ind/rsp/cnf
	Output	req/ind/rsp/cnf
Gerenciamento de Eventos	Define_event_condition	req/ind/rsp/cnf
	Delete_event_condition	req/ind/rsp/cnf
	Get_event_condition_attribute	req/ind/rsp/cnf
	Report_event_condition_status	req/ind/rsp/cnf
	Alter_event_condition_monitorining	req/ind/rsp/cnf
	Triger_event	req/ind/rsp/cnf
	Define_event_action	req/ind/rsp/cnf
	Delete_event_action	req/ind/rsp/cnf
	Get_event_action_attributes	req/ind/rsp/cnf
	Report_event_action_status	req/ind/rsp/cnf
	Define_event_enrollment	req/ind/rsp/cnf
	Delete_event_enrollment	req/ind/rsp/cnf
	Get_event_enrollment	req/ind/rsp/cnf
	Report_event_enrollment	req/ind/rsp/cnf
	Alter_event_enrollment	req/ind/rsp/cnf
	Event_notification	req/ind/rsp/cnf
	Acknowledge_event_notification	req/ind/rsp/cnf
	Get_alarm_summary	req/ind/rsp/cnf
	Get_alarm_enrollment_summary	req/ind/rsp/cnf
	Attach_to_event_condition_modifier	req/ind/rsp/cnf
Gerenciamento de Jornal	Read_journal	req/ind/rsp/cnf
	Write_journal	req/ind/rsp/cnf
	Initialize_journal	req/ind/rsp/cnf
	Report_journal_status	req/ind/rsp/cnf
Gerenciamento de Arquivo	File_open	req/ind/rsp/cnf
	File_read	req/ind/rsp/cnf
	File_close	req/ind/rsp/cnf
	File_rename	req/ind/rsp/cnf
	File_delete	req/ind/rsp/cnf
	File_direction	req/ind/rsp/cnf

Fig. A.2 Serviços MMS (Continuação)

ANEXO B

OBJETOS MMS USADOS NESTE TRABALHO

Neste anexo serão apresentados os objetos MMS utilizados durante este trabalho. Cada objeto é tratado dentro da classe de Serviços MMS a que está ligado.

Para cada objeto é fornecido primeiramente, uma pequena descrição informal. Esta descrição é superficial e visa apenas mostrar qual a finalidade do objeto dentro do modelo MMS. Após a descrição informal é apresentado o modelo contendo o conjunto de Atributos do objeto segundo o exemplo de definição sintática dado no capítulo 2 deste trabalho. Cada atributo também é descrito informalmente na sequência.

Deve ser observado que os objetos analisados aqui seguem o Padrão MMS genérico, detalhes adicionais descritos nos *Companion Standards* não são mostrados neste anexo.

B.1 Objeto VMD

Como já foi dito, o VMD representa (à luz do MMS) os recursos e funcionalidades de um dispositivo real do ambiente de manufatura. Dentro da representação do tipo Objetos, cada informação, recurso, capacidade ou *status* é modelado através de informações contidas nos atributos do objeto VMD. A descrição deste objeto é a seguinte:

Object: VMD
Key Attribute: *Executive Function*
Attribute: *Vendor Name*
Attribute: *Model Name*
Attribute: *Revision*
Attribute: *Logical Status* {STATE_CHANGES_ALLOWED,
NO_STATE_CHANGES_ALLOWED,
LIMITED_SERVICES_SUPPORTED}
Attribute: *List of Capabilities*
Attribute: *Phisical Status* {OPERATIONAL, INOPERABLE,
PARTIALLY_OPERATIONAL,
NEEDS_COMMISSIONING}
Attribute: *List of Program Invocations*
Attribute: *List of Domains*

Attribute: *List of Transaction Objects*
Attribute: *List of Upload State Machines* (ULSM)
Attribute: *List of Other VMD-Specific Objects*

Onde cada um dos atributos pode ser descritos da seguinte forma:

- Executive Function, a existência de uma função executiva corresponde à existência do VMD;

- Vendor Name, identifica o vendedor do sistema que suporta este VMD;

- Model Name, identifica o modelo do sistema que suporta este VMD, fornecido pelo vendedor;

- Revision, identifica o nível de revisão do sistema que suporta este VMD, fornecido pelo implementador;

- Logical Status, indica que serviços disponíveis no VMD estão aptos a serem executados. Existem 3 níveis de funcionalidades disponíveis através do MMS que são descritas por este atributo.

- List of Capabilities, capacidade é um recurso (físico ou lógico) ou um conjunto de recursos definidos localmente que pode ser identificados por um *string* de caracteres. A definição e gerenciamento destas capacidades estão fora do escopo do padrão MMS, no entanto assume-se que a existência e o status destas capacidades são do conhecimento da Função Executiva;

- Phisical Status, um ou mais atributos que descreve o estado associado a cada capacidade;

- List of Program Invocations, contém referências aos objetos Invocação de Programa associados a este VMD;

- List of Domains, contém referências aos objetos Domínio associados a este VMD;

- List of Transaction Objects, contém referências aos objetos Transação associados a este VMD;

- List of Upload State Machines, contém referências as máquinas de estados de *Upload* associados a este VMD. Estas máquinas de estados são criadas durante uma sequência de *UpLoad*;

- List of Other VMD-Specific Objects, contém referências a outros objetos MMS específicos (subordinados) deste VMD.

B.2 Objeto Transação

Quando um VMD recebe uma primitiva de indicação de um serviço confirmado qualquer, um objeto de Transação deve ser criado, afim de acompanhar e governar a execução deste serviço. A descrição deste objeto é dada a seguir:

Object: *Transaction*
Key attribute: *Invoke_ID*
Attribute: *List of Pre-execution Modifiers*
Attribute: *Current Modifier Reference*
Attribute: *Confirmed Service Request*
Attribute: *List of Post-execution Modifiers*
Attribute: Cancelable {TRUE,FALSE}

Onde cada um dos atributos pode ser descrito da seguinte forma:

- Invoke_ID, identifica a invocação, sendo único para todos os pedidos pendentes;

- List of Pre-execution Modifiers, identifica modificadores que devem ser satisfeitos antes da execução *Confirmed Service Request* começar;

- Current Modifier Reference, referencia um objeto *Semaphore Entry* ou *Event Enrollment* que controla a execução do modificador corrente;

- Confirmed Service Request, identificador e argumento do serviço pendente;

- List of Post-execution Modifiers, referencia os objetos *Semaphore Entry* que esta invocação de serviço possue devido ao modificador *Attach to Semaphore* processado;

- Cancelable, indica se o serviço MMS *Cancel* pode ou não ser usado com sucesso (inicialmente *true*, pode ser tornado igual a *false* durante a execução do serviço).

A seguir serão descritos as classes de objetos MMS utilizadas neste trabalho.

B.3 Objetos da Classe Serviços de Gerenciamento de Domínio

Domínios representam subconjuntos de potencialidades (instruções e/ou dados) que são usados para um propósito específico.

Um domínio pode vir a existir de três maneiras distintas; ele pode ser criado explicitamente no início de um processo de *Download* (execução dos serviços MMS *Initite_Down_Load_Sequence*, *Down_Load_Segment* e *Terminate_Down_Load_Sequence*); pode ser criado como parte da execução de uma invocação de programa; ou ainda por outro meio local ao VMD. Um domínio pode também ser pré-definido em um sistema, existindo antes mesmo de se estabelecer um contexto MMS (domínio estático).

O modelo do objeto Domínio contém uma série de atributos e é mostrados abaixo.

Object: *Domain*
Key Attribute: *Domain Name*
Attribute: *List of Capabilities*
Attribute: *State* {LOADING, COMPLETE, INCOMPLETE, READY, IN_USE}
Constraint: *State* = {LOADING, COMPLETE, INCOMPLETE}
Attribute: *Assigned Application Association*
Attribute: *MMS Deletable* {TRUE, FALSE}
Attribute: *Sharable* {TRUE, FALSE}
Attribute: *Domain Content*
Attribute: *List of Subordinate Objects*
Constraint: *State* = IN_USE
Attribute: *List of Program Invocation References*
Attribute: *Upload In Process*
Attribute: *Additional Detail*

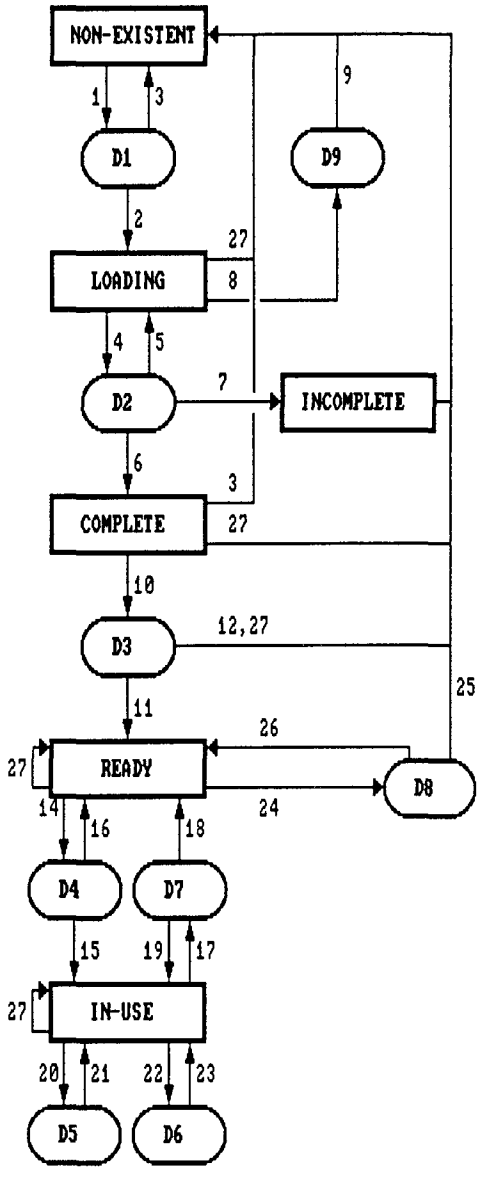
Onde cada um dos atributos pode ser descritos da seguinte forma:

- Domain Name, este atributo identificará unicamente o domínio dentro de um VMD. Ele sempre terá o escopo específico de VMD (pode ser referenciado por qualquer cliente MMS);

- List of Capabilities, que lista os parâmetros específicos de implementação necessários para particionar os recursos do VMD (alocação de memória, entradas e saídas, etc) para este domínio;

- State, especifica o estado em que o domínio se encontra. As opções possíveis podem ser vistas na máquina de estados da figura B.1;

- Assigned Application Association, durante o processo de *Download* de um domínio, o domínio é dependente da associação de aplicação sobre a qual foi criado. Se esta é perdida antes que o domínio seja colocado em um estado *READY*, o domínio será automaticamente deletado;



TRANSICOES:

1. InitiateDownLoadSequence.ind
2. InitiateDownLoadSequence.rsp (+)
3. InitiateDownLoadSequence.rsp (-)
4. DownLoadSequence.req
5. DownLoadSequence.cnf (+) [more_follows = true]
6. DownLoadSequence.cnf (+) [more_follows = false]
7. DownLoadSequence.cnf (-)
8. TerminateDownLoadSequence.ind
9. TerminateDownLoadSequence.rsp (+) ou (-)
10. TerminateDownLoadSequence.req [discart = false]
11. TerminateDownLoadSequence.cnf (+)
12. TerminateDownLoadSequence.cnf (-)
13. TerminateDownLoadSequence.req [discart = true]
14. CreateProgramInvocation.ind [program_invocation_count = 0]
15. CreateProgramInvocation.rsp (+)
16. CreateProgramInvocation.rsp (-)
17. DeleteProgramInvocation.ind [program_invocation_count = 1]
18. DeleteProgramInvocation.rsp (+)
19. DeleteProgramInvocation.rsp (-)
20. CreateProgramInvocation.ind [program_invocation_count > 0]
21. CreateProgramInvocation.rsp (+) ou (-)
22. DeleteProgramInvocation.ind [program_invocation_count > 1]
23. DeleteProgramInvocation.rsp (+) ou (-)
24. DeleteDomain.ind
25. DeleteDomain.rsp (+)
26. DeleteDomain.rsp (-)
27. Abort.ind

Fig. B.1 Máquina de Estados de Domínio

- MMS Deletable, define se este objeto pode ser deletado, usando um serviço *Delete_Domain*, ou não;

- Sharable, especifica se este domínio pode ser usado em mais de uma Invocação de Programa ao mesmo tempo;

- Domain Content, informação contida nos dados de carga, que é matéria dos serviços *Down_Load_Segment* e *Load_Domain_Content*. A natureza da informação contida é específica de implementação e não é descrita no padrão MMS;

- List of Subordinate Objects, que lista as referências aos objetos nomeados, que são partes do atributo *domain Content*;

- List of Program Invocation References, que lista as referências para invocações de programa que usam atualmente este domínio;

- UpLoad In Progress, especifica o número de sequências *UpLoad* correntemente ativadas para este domínio;

- Additional Detail, que contém zero ou mais atributos cujo significado sintaxe são definidas pelo *Companion Standard* apropriado.

B.4 Objetos da Classe Serviços de Gerenciamento de Invocação de Programa

Invocações de Programas podem ser dinâmicas (criadas e destruídas do sistema ou pelos serviços MMS ou por uma ação local) ou estáticas (pré-definidas dentro do VMD).

O modelo do objeto Invocação Programa de contém uma série de atributos que são mostrados abaixo:

Object: *Program Invocation*
Key Attribute: *Program Invocation Name*
Attribute: *State*: {IDLE, STARTING, RUNNING, STOPPING, STOPPED, RESETTING, RESUMING, UNRUNNABLE}
Attribute: *List of Domain references*
Attribute: *MMS Deletable* {TRUE,FALSE}
Attribute: *Reusable* {TRUE,FALSE}
Attribute: *Monitor* {TRUE,FALSE}
Constraint: *Monitor* = TRUE
Attribute: *Event Condition*
Attribute: *Event Action*
Attribute: *Event Enrollment*
Attribute: *Start Argument* (inicialmente vazio)
Attribute: *Addicional Detail*

Onde cada um dos atributos pode ser descritos da seguinte forma:

- Program Invocation Name, é o identificador da invocação de programa;

- State, indica o estado da invocação de programa. O diagrama de estado deste atributo pode ser visto na figura B.2 (*Companion Standards* podem aumentar esta tabela de estados);

- List of Domain references, é uma lista de referências para domínios que compõem esta invocação de programa;

- MMS Deletable, indica se a invocação de programa pode ser deletada usando serviços MMS ou não (*true, false*);

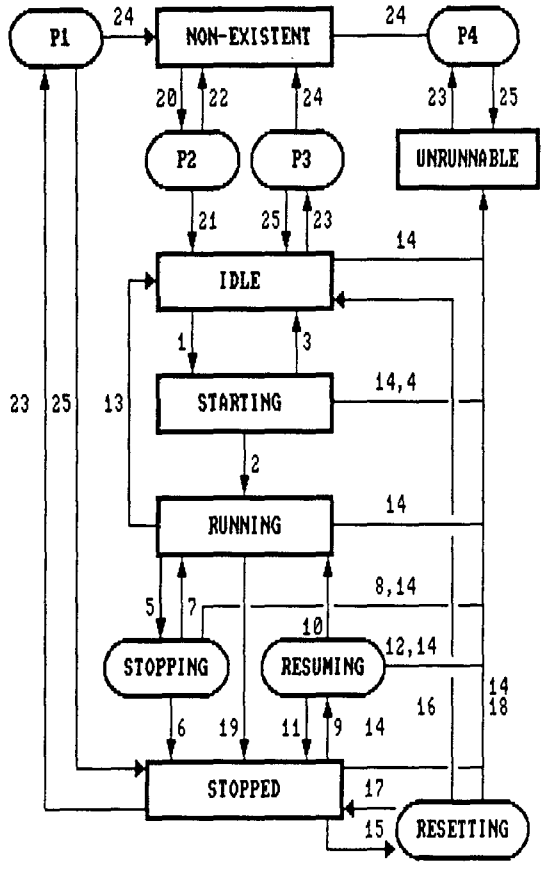
- Reusable, indica se a invocação de programa retornará ao estado *IDLE* após o término de sua execução ou não (*true, false*). Caso contrário, irá ao estado UNRUNNABLE;

- Monitor, indica se a monitoração de programa está em efeito para esta invocação de programa ou não. Uma invocação de programa monitorada usa as facilidades de Gerenciamento de Eventos para informar ao Usuário MMS solicitante sempre que a invocação de programa sair do estado *RUNNING*. Se este atributo possuir o valor *true* são criados os objetos abaixo:

    - Event Condition, é uma condição de evento cujo nome é igual ao da invocação de programa e seus atributos serão:
        - *Event Condition Class = MONITORED*;
        - *Enabled = TRUE*;
        - *Monitored Variable Reference* = (invocação de programa estar no estado *RUNNING*);
        - *Event Action*, é uma ação de evento cujo nome é o mesmo que o da invocação de programa e seu atributo *Confirmed Service Request* contém o serviço MMS *Get_Program_Invocation_Attributes*;

    - *Event Enrollment*, é um registro de evento cujo nome é o mesmo que o da invocação de programa e cujos atributos são:
        - *Enrollment Class = NOTIFICATION*;
        - *Event Condition Transitions = ACTIVE-to-IDLE*;
        - *Assigned Application Association* é a associação sobre a qual a invocação de programa foi criada;
        - *Duration* é determinada pelo parâmetro do serviço *Create_Program_Invocation*;

- Start Argument, retém o valor do parâmetro passado para a invocação de programa no pedido mais recente do serviço *Start*;
- Addicional Detail, permite atributos adicionais a serem incluídos pelos *Companion Standards*.



TRANSICOES:

1. Start.ind
2. Start.rsp (+)
3. Start.rsp (-) [nao destrutivel]
4. Start.rsp (-) [destrutivel]
5. Stop.ind
6. Stop.rsp (+)
7. Stop.rsp (-) [nao destrutivel]
8. Stop.rsp (-) [destrutivel]
9. Resume.ind
10. Resume.rsp (+)
11. Resume.rsp (-) [nao destrutivel]
12. Resume.rsp (-) [destrutivel]
13. (fim de programa) [reusable = true]
14. Kill.rsp (+)
15. Reset.ind
16. Reset.rsp (+) [reusable = true]
17. Reset.rsp (-) [nao destrutivel]
18. Reset.rsp (-) [destrutivel]
19. (programa parado)
20. CreateProgramInvocation.ind
21. CreateProgramInvocation.rsp (+)
22. CreateProgramInvocation.rsp (-)
23. DeleteProgramInvocation.ind
24. DeleteProgramInvocation.rsp (+)
25. DeleteProgramInvocation.rsp (-)

Fig. B.2 Máquina de Estados de Inv. Programa

B.5 Objetos da Classe Serviços de Acesso a Variáveis

Os serviços de acesso a variáveis permite que um usuário MMS cliente tenha acesso a variáveis tipadas através do VMD. Uma variável MMS é um elemento abstrato do VMD que é capaz de prover (quando lida), ou aceitar (quando escrita), um valor de um certo

tipo. Variáveis MMS não podem ser consideradas como variáveis no conceito usual, mas sim um caminho de acesso para uma variável real do dispositivo.

Um VMD que permite o acesso a uma ou mais de suas variáveis reais usando serviços de acesso a variável, provê o mapeamento entre as variáveis reais e um ou mais objetos de acesso a variável. Este mapeamento é feito pelas funções *V_GET* e *V_PUT* que obteem e atualizam o valor da variável real do dispositivo.

O MMS define dois objetos que descrevem o mapeamento entre uma variável MMS e uma variável real no VMD, são os objetos *Unnamed Variable* e *Named Variable.*

Existem também objetos que descrevem o acesso a múltiplas variáveis que são os objetos *Scattered Access* e *Named Variable List.* Além destes há o objeto *Named Type,* que proporciona a atribuição de um nome para uma descrição de tipo MMS. Neste trabalho são descritos apenas os objetos *Unnamed Variable, Named Variable* e *Named Type.*

B.5.1 Objeto Variável Não Nomeada

Este objeto descreve o mapeamento de uma única variável MMS não nomeada para uma variável real existente num endereço fixo e conhecido dentro do VMD. Este objeto nunca é criado ou destruído (sempre estático).

Os atributos deste objeto são especificados a seguir:

Object: *Unnamed Variable*
Key Attribute: *Address*
Attribute: *MMS Deletable* {FALSE}
Attribute: *Access Method* {PUBLIC}
Attribute: *Type Description*

Onde cada um dos atributos pode ser descritos da seguinte forma:

- Address, fornece a localização da variável real no sistema que suporta o VMD;

- MMS Deletable, sempre será especificado como *false*;

- Access Method, será sempre *PUBLIC*;

- Type Description, indica o tipo abstrato da variável real fundamental, quando vista usando os serviços MMS.

B.5.2 Objeto Variável Nomeada

Descreve o mapeamento entre uma variável MMS e uma variável real definida de aplicação no VMD.

Este objeto pode ser usado para descrever uma variável que não tenha um endereço conhecido (variável calculada) ou ainda quando o endereço não for considerado como *PUBLIC*. Pode também ser usada para descrever um conjunto de um ou mais elementos de dados relacionados com a aplicação que são acessados usando um único nome com uma única operação, ou para apresentar uma descrição mais explícita do tipo para um conjunto de um ou mais objetos Variáveis Não Nomeados, ou os dois.

Os atributos deste objeto são especificados a seguir:

Object: *Named Variable*
Key Attribute: *Variable Name*
Attribute: *MMS Deletable*
Attribute: *Type Description*
Attribute: *Access Method* {PUBLIC, ...}
Constraint: *Access Method* = PUBLIC
Attribute: *Address*

Onde cada um dos atributos pode ser descritos da seguinte forma:

- Variable Name, identifica o objeto *Named Variable*, podendo ser definido no escopo:
  - específico de VMD, podendo ser referenciado por todos os cliente;
  - específico de Domínio, que existe durante a vida do domínio ou se explicitamente deletado;
  - específico de Associação de Aplicação, que pode ser referenciado apenas sobre a Associação de Aplicação;

- MMS Deletable, indica se este objeto pode ser deletado, utilizando o serviço *Delete_Variable_Access*, ou não;

- Type Description, indica o tipo abstrato da variável real fundamental, quando vista usando os serviços MMS;

- Access Method, provê informações necessárias pelo VMD a fim de alocar a variável real. Este atributo pode conter qualquer valor que seja apropriado para o VMD, incluindo atributos adicionais que não são especificados por este padrão. Se o método de acesso é declarável ou possível de ser obtido usando

serviços MMS, então este atributo terá o valor *PUBLIC* e o atributo Address será disponível para o cliente;

- Address, existe apenas quando o atributo *Access Method* é *PUBLIC*, provê a localização da variável descrita pelo atributo *Type Description*, em endereços contíguos.

B.5.3 Objeto Tipo Nomeado

Proporciona a atribuição de um nome a uma descrição de tipo MMS.

Os atributos do objeto *Named Type* são especificados a seguir:

Object: *Named Type*
Key Attribute: *Type Name*
Attribute: *MMS Deletable*
Attribute: *Type Description*

- MMS Deletable, indica se este objeto pode ser deletado, utilizando o serviço *Delete_Variable_Access*, ou não;

- Type Description, indica o tipo abstrato da variável real fundamental, quando vista usando os serviços MMS;

B.6 Objetos da Classe Serviços de Gerenciamento de Evento

Esta classe provê facilidades que permitem um usuário MMS cliente definir e gerenciar objetos de evento no VMD e para obter notificações das ocorrências de eventos.

São incluidos três objetos:

- Objeto Event Condition, que modela a parte da informação de estado que define a detecção do evento. Este objeto também inclui informações que assistem na detecção de alarmes;

- Objeto Event Action, que modela a parte da informação de estado que define os serviços MMS a serem executados na ocorrência do evento;

- Objeto Event Enrollment (opcional), é utilizado para ligar um objeto condição de evento a um objeto ação de evento. Além disso permite definir notificações a serem enviadas a um cliente quando da ocorrência de um evento.

B.6.1 Objeto Condição de Evento

Este objeto modela os aspectos visíveis ao MMS de uma condição de evento.

A seguir são descritos os atributos deste objeto:

Object: *Event Condition*
Key attribute: *Event Condition Name*
Attribute: *MMS Deletable* {TRUE,FALSE}
Attribute: *Event Condition Class* {NETWORK_TRIGGERED, MONITORED}
Attribute: *State*
Attribute: *Priority*
Attribute: *Severity*
Attribute: *Aditional Detail*
Attribute: *List Of Event Enrollment Reference*
Constraint: *Event Condition Class* = MONITORED
Attribute: *Enabled*
Attribute: *Alarm Summary Reports*
Attribute: *Monitored Variable Reference*
Attribute: *Evaluation Interval*
Attribute: *Time Of Last Transition To Active*
Attribute: *Time Of Last Transition To Idle*

- Event Condition Name, identifica este objeto;

- MMS Deletable, indica se este objeto pode ser deletado, utilizando o serviço Delete_Event_Condition, ou não;

- Event Condition Class, indica a classe da condição de evento. Existem duas classes de Condições de Eventos:
  - *Network-triggered*, define um evento que ocorre devido a um pedido explícito de um cliente, utilizando o serviço *Trigger_Event*;
  - *Monitored*, é uma representação virtual de um aspecto da atividade do VMD que, devido ao critério do projeto da aplicação, representa uma ocorrência significante no processamento do VMD. Eventos associados com este tipo são detectados por ação autônoma do VMD;

- State, manterá o estado corrente deste objeto, possuindo as seguintes condições:
  - Caso seja uma condição de evento da classe *Network_Triggered* este atributo terá sempre o valor *DISABLED*;
  - Caso seja uma condição de evento da classe *Monitored* dependerá do valor do atributo *Enabled*, se *false* seu valor será *DISABLED*; se *true* e o valor do atributo *monitored Variable Reference* não for *UNSPECIFIED* ou

UNDEFINED, então o valor deste atributo será *IDLE* se o valor da variável referenciada pelo atributo *monitored Variable Reference* seja *false*; ou *ACTIVE*, se a variável referenciada pelo atributo *Monitored Variable Reference* seja *true*;

- Priority, indicará a importância da condição de evento relativa a outras Condições de Eventos definidas no VMD;

- Severity, indica o efeito do evento no processo que está sendo controlado;

- Aditional Detail, manterá informações de estado específico da classe do dispositivo. A semântica é especificada pelos *Companion Standards*;

- List Of Event Enrollment Reference, conterá uma lista de referências para Registros de Eventos que especifica uma ou mais transições do estado da condição de evento no atributo *Event Condition Transitions* do objeto *Event Enrollment*;

- Enabled, existe apenas nas Condições de Eventos Monitoradas, especificando se a troca no valor da variável monitorada referenciada na condição de evento causará o processamento de um registro de evento ou não;

- Alarm Summary Reports, existe apenas nas Condições de Eventos Monitoradas. Se *true*, especifica que a condição de evento será incluida nos sumário de alarmes sem relação com o estado do registro de eventos; se *false*, não será incluida nos sumários de alarme, a menos que no último registro de evento especifica um *Alarm Acknowledgment Rule* diferente de *NONE*;

- Monitored Variable Reference, existe apenas nas Condições de Eventos Monitoradas. Para um objeto *Event Condition* definido no MMS, este atributo referência um objeto Variável (nomeada ou não) do tipo *boolean*. Para um objeto definido localmente, este atributo terá o valor *UNSPECIFIED*, indicando que a transição do evento não é determinada pelo valor de um objeto visível MMS;

- Evaluation Interval, especifica o tempo máximo, em milisegundos, entre avaliações do estado da condição de evento;

- Time Of Last Transition To Active, existe apenas nas Condições de Evento Monitoradas. Registra o instante da última transição detectada do estado da condição de evento para *ACTIVE*;

- Time Of Last Transition To Idle, existe apenas nas Condições de Evento Monitoradas. Registra o instante da última transição detectada do estado da condição de evento para *IDLE*;

B.6.2 Objeto Ação de Evento

É um serviço MMS confirmado que será executado sempre que uma especificada transição de um estado da condição de evento é detectada. Os atributos deste objeto são vistos abaixo:

Object: *Event Action*
Key attribute: *Event Action Name*
Attribute: *MMS Deletable*
Attribute: *Confirmed Service Request*
Attribute: *List Of Modifier*
Attribute: *List Of Event Enrollment Reference*
Attribute: *Additional Detail*

- Event Action Name, identifica este objeto;

- MMS Deletable, indica se este objeto pode ser deletado, pelo serviço *Delete_Event_Action*, ou não;

- Confirmed Service Request, indica o procedimento do serviço que será executado pelo VMD sempre que ocorrer um evento parao qual um registro especifique esta ação de evento. O resultado da execução do procedimento de serviço pedido com o argumento provido neste atributo será incluído no pedido do serviço *Event_Notification*, que será gerado como resultado do processamento da transição de evento;

- List Of Modifier, contém a lista de modificadores, se existir, que se aplicará para toda execução da ação de evento;

- List Of Event Enrollment Reference, manterá uma lista de referências aos objetos Registros de Evento que estão correntemente referenciando esta ação de evento;

- Additional Detail, manterá informações de estado específicas da classe do dispositivo. A semântica é especificada pelos *Companion Standards*;

B.6.3 Objeto Resgistro de Evento

Representa o pedido vindo do usuário MMS cliente para que seja notificado da ocorrência de uma transição de uma condição de evento ou para espera de execução de um serviço MMS confirmado até a ocorrência da transição de uma condição de evento, utilizando o modificador *Attach to Event Condition*. Os atributos deste objeto são vistos abaixo:

Object: *Event Enrollment*
Key attribute: *Event Enrollment Name*
Attribute: *MMS Deletable* {TRUE,FALSE}
Attribute: *Enrollment Class* {MODIFIER, NOTIFICATION}
Attribute: *Event Condition Reference*
Attribute: *Event Condition Transitions*
Attribute: *Application Association Local Tag*
Attribute: *Aditional Detail*
Constraint: *Enrollment Class* = MODIFIER
Attribute: *Invoke ID*
Attribute: *Remaining Acceptable Delay*
Constraint: *Enrollment Class* = NOTIFICATION
Attribute: *Notification Lost*
Attribute: *Event Action Reference*
Attribute: *Duration*
Attribute: *Client Application*
Attribute: *Alarm Acknowledgment Rule*
Attribute: *Time Active Acknowledged*
Attribute: *Time Idle Acknowledged*
Attribute: *State*

- Event Enrollment Name, identifica este objeto;

- MMS Deletable, indica se este objeto pode ser deletado, pelo serviço *Delete_Event_Enrollment*, ou não;

- Enrollment Class,são definidos dois valores:
  - *MODIFIER*, é um registro de evento temporário (executado num instante e deletado logo após), criado no recebimento de uma indicação de serviço especificando um serviço MMS confirmado que utiliza o modificador de serviço *Attach To Event Condition*;
  - *NOTIFICATION*, é um registro de evento definido explicitamente ou predefinido. Este pede que pedidos de *Event_Notification* sejam emitidos na ocorrência de algumas das transições da condição de evento indicada;

- Event Condition Reference, referencia a condição de evento que causará a execução deste registro de evento;

- Event Condition Transitions, quando as referências do registro de evento tem uma Condição de Evento Monitorada, este atributo contém um conjunto de transições de condição de evento ao qual este registro de evento invocará o serviço *Event_Notification*. Consiste de um conjunto composto de membros escolhidos entre:
  - *DISABLED-to-ACTIVE*;
  - *DISABLED-to-IDLE*;
  - *IDLE-to-ACTIVE*;
  - *IDLE-to-DISABLED*;
  - *ACTIVE-to-IDLE*;
  - *ACTIVE-to-DISABLED*;
  - *ANY-to-DELETED.*

- Application Association Local Tag, identifica localmente a Associação de Aplicação e a sintaxe abstrata que será usada pelo *Event_Notification*;

- Aditional Detail, manterá informações de estado específica da classe do dispositivo. A semântica é especificada pelos *Companion Standards*;

- Invoke ID, existe apenas nos Registros de Eventos *MODIFIER*. Contém o *invoke ID* do objeto Transação do serviço modificado;

- Remaining Acceptable Delay, exite apenas nos Registros de Eventos *MODIFIER*. Contém ou o valor *FOREVER* ou o tempo em segundos, no qual usuário MMS pedinte do registro de evento *MODIFIER* deseja esperar até que a condição de evento ocorra;

- Notification Lost, exite apenas nos Registros de Eventos da classe *Notification*. Este será *true* durante o período de tempo que a invocação do serviço *Event_Notification* para eventos associados com este registro de evento tenha sido suspendido devido a limitações de recursos no VMD, ou se a notificação não pode ser enviada devido a inabilidade de estabelecer uma Associação de Aplicação para registros de Duração *PERMANENT*. Do contrário, será *false*;

- Event Action Reference, exite apenas nos Registros de Eventos da classe *Notification*. Pode ser *UNSPECIFIED*, indicando que será enviado apenas o pedido de *Event_Notification*, ou este pode conter uma referência a ação de evento, indicando que o registro de evento pede um *Event_Notification* contendo o resultado da execução da ação de evento especificada;

- Duration, exite apenas nos Registros de Eventos da classe *Notification*. Indica a duração do registro de evento e pode ter dois valores:

    - *CURRENT*, indica que o registro de evento é especificado para a vida da Associação de Aplicação sobre a qual este objeto foi criado;
    - *PERMANENT*, é especificado durante toda a vida do VMD, a menos que explicitamente deletado;

- Client Application, exite apenas nos Registros de Eventos da classe *Notification*. Contém o identificador do cliente registrado;

- Alarm Acknowledgement Rule, exite apenas nos Registros de Eventos da classe *Notification* que referencia uma condição de evento *MONITORED*. Este indica o nível de reconhecimento requerido para o serviço *Event_Notification*. O valor deste atributo também determina se este registro de evento será incluído no sumários de registro de Alarmes ou não;

- Time Active Acknowledgment, exite apenas nos Registros de Eventos *Notification* que referencia uma condição de evento *MONITORED*. Registra o instante no qual um reconhecimento da transição de evento mais recentemente detectada da condição de evento para o estado *Active* foi recebido;

- Time Idle Acknowledgment, exite apenas nos Registros de Eventos *Notification* que referencia uma condição de evento *MONITORED*. Registra o instante no qual um reconhecimento da transição de evento mais recentemente detectada da condição de evento para o estado *IDLE* foi recebido;

- State, mantém o estado atual deste objeto. Num instante este objeto tem um estado maior, que é ou *NON_EXISTENT* ou igual ao estado do objeto Condição de Evento referenciado. Dependendo do valor do atributo *Alarm Acknowledgment Rule* deste objeto, o objeto Registro de Evento terá estados menores;

B.7 Objetos da Classe Serviços de Gerenciamento de Semáforo

O gerenciamento de semáforos permite sincronização, controle e coordenação de recursos compartilhados entre usuários MMS. Um semáforo é referenciado por um nome específico de VMD ou de domínio; podendo ser pré-definido, ou definido quando da criação de um domínio, ou pelo uso do serviço *Define_Semaphore*.

O MMS proporciona a utilização de dois tipos de semáforos:

- Semáforo *Token*, que permite um ou múltiplos possuidores;

- Semáforo *Pool*, permite uma alocação explícita ou dinâmica de fixas nomeadas.

Os serviços MMS permitem apenas a criação de semáforos tipo *TOKEN*, pois um semáforo do tipo *POOL* requer uma ligação entre entidades físicas ou lógicas no dispositivo real com a ficha nomeada no VMD.

Nesta classe existem dois objetos, o objeto Semáforo e o objeto *Semaphore Entry*, este último é criado sempre que um usuário MMS pede o controle do objeto Semáforo.

B.7.1 Objeto Semáforo

Sua estrutura é a seguinte:

Object: *Semaphore*
Key attribute: *Semaprore Name*
Attribute: *MMS Deletable* {TRUE, FALSE}
Attribute: *Class* {TOKEN, POOL}
Constraint: *Class* = TOKEN
Attribute: *Number Of Tokens*
Attribute: *Number Of Owners Tokens*
Constraint: *Class* = POOL
Attribute: *List Of Named Tokens*
Attribute: *List Of Named Tokens States*
Attribute: *List Of Owners*
Attribute: *List Of Requesters*

- Semaprore Name, identifica o semáforo;

- MMS Deletable, indica se este objeto pode ser deletado, utilizando o serviço *Delete_Semaphore*, ou não;

- Class, especifica a classe do semáforo;

- Number Of Tokens, define o número máximo de possuidores permitidos pelo semáforo *TOKEN*;

- Number Of Owners Tokens, contém o número de fichas atualmente possuidas por este semáforo *TOKEN*;

- List Of Named Tokens , define o nome das fichas controladas pelo semáforo *POLL*;

- List Of Named Tokens States, contém o estado (*FREE* ou *OWNED*) para cada ficha nomeada controlada pelo semáforo *POLL*;

- List Of Owners, contém uma lista de referências para objetos *Semaphore Entry* possuindo o semáforo;

- List Of Requesters, contém uma lista de referências para objetos *Semaphore Entry* esperando a obtenção do controle do semáforo;

B.7.2 **Objeto** Semaphore Entry

Object: *Semaphore Entry*
Key attribute: *Entry ID*
Attribute: *Entry Class* {SIMPLE, MODIFIER}
Attribute: *Semaphore Name*
Attribute: *Requester Application Reference*
Attribute: *Application Association Local Tag*
Attribute: *Invoke ID*
Attribute: *Named Token*
Attribute: *Priority*
Attribute: *Remaining Acquisition Delay*
Attribute: *Remaining Control Time Out*
Attribute: *Abort On Time Out* {TRUE, FALSE}
Constraint: *Abort On Time Out* = FALSE
Attribute: *Event Condition Reference*
Attribute: *Relinquish If Connection Lost* {TRUE, FALSE}
Attribute: *Entry State* {QUEUES, OWNER, HUNG}

- Entry ID, identifica este objeto e será único para todos os objetos *Semaphore Entry* relacionados a um semáforo;

- Entry Class, contém o valor *MODIFIER* se este objeto foi criado por um serviço *Modified* por um modificador *Attach To Semaphore*, caso contrário terá o valor *SIMPLE*.

- Semaphore Name, indica o semáforo pedido ou possuido pela *Semaphore Entry*;

- Requester Application Reference, identifica o processo de aplicação cujo pedido criou este objeto;

- Application Association Local Tag, identifica a Associação de Aplicação sobre a qual este objeto foi criado;

- Invoke ID, identifica o objeto Transação no VMD relacionado ao pedido;

- Named Token, é usado apenas se o semáforo pedido é um semáforo *POLL* e contém uma ficha nomeada pedida ou controlada;

- Priority, especifica a prioridade que este objeto tem comparada com outros objetos *Semaphore Entry*, enquanto na lista de espera;

- Remaining Acquisition Delay, contém ou o tempo máximo de espera para a posse do semáforo, além do qual é cancelado, ou o valor *FOREVER*;

- Remaining Control Time Out, contém ou o tempo máximo de posse do semáforo, ou o valor *FOREVER*;

- Abort On Time Out, especifica se ou não ocorrerá o aborto da Associação de Aplicação se ocorrer o *Time Out* de controle;

- Event Condition Reference, é uma referência a uma Condição de Evento, cujo nome é o mesmo do semáforo referenciado e cujos demais atributos são:
  - *Event Condition Class* = MONITORED;
  - *Enabled* = TRUE;
  - *Monitored Variable Reference* é a condição que em um dos *Semaphore Entry* deste semáforo tenha ocorrido um *Remaining Control Time Out*;

- Relinquish If Connection Lost, especifica se ou não a entrada de semáforo liberará o semáforo se a Associação de Aplicação identificada por *Application Association Local Tag* for perdida;

- Entry State, possui o valor *QUEUED* enquanto este objeto está na lista de espera, *OWNER* enquanto este estiver na lista dos possuidores e a Associação de Aplicação estiver mantida, e *HUNG* enquanto este estiver na lista de possuidores e a Associação de Aplicação foi perdida.

B.8 Objetos da Classe Serviços de Comunicação com o Operador

Esta classe proporciona mecanismos para comunicação com uma estação operador que permite listar dados, entrada de dados, ou ambos.

Os serviços de comunicação com o operador usa um objeto *Operator Station*, que é especificado a seguir:

Object: *Operator Station*
Key attribute: *Operator Station Name*
Attribute: *Station Type* {ENTRY, DISPLAY, ENTRY_DISPLAY}
Constraint: *Station Type* = ENTRY
Attribute: *Input Buffer*
Attribute: *State* {IDLE, WAITING_FOR_INPUT_STRING, INPUT_BUFFER_FILLED}
Constraint: *Station Type* = DISPLAY
Attribute: *List of Output Buffer*
Attribute: *State* {IDLE, OUTPUT_BUFFERS_FILLED}
Constraint: *Station Type* = ENTRY_DISPLAY
Attribute: *Input Buffer*
Attribute: *List of Output Buffer*
Attribute: *State* {IDLE, OUTPUT_BUFFERS_FILLED, DISPLAY_LIST_OF_PROMPT_DATA, WAITING_FOR_INPUT_STRING, INPUT_BUFFER_FILLED}

- Operator Station Name, identifica este objeto;

- Station Type, indica o tipo de estação;

- Input Buffer, contém o valor do parâmetro *Input String* da resposta do serviço *Input*. Terão este atributo os objetos do tipo *ENTRY* ou *ENTRY_DISPLAY*.

- List of Output Buffer, contém ou o valor do parâmetro *List of Output Data* do pedido do serviço *Output*, ou o valor do parâmetro *List of Prompt Data* do pedido de serviço *Input*. Apenas objetos do tipo *DISPLAY* e *ENTRY_DISPLAY* terão este atributo;

- State, contém o estado deste objeto, como visto na figura B.3.

B.9 Objetos da Classe Serviços de Gerenciamento de Jornal

O propósito desta classe é prover uma facilidade para registrar e buscar informações ordenadas cronologicamente de eventos de interesse, conteúdo de variáveis de interesse na conjunção com eventos, ou os dois, além de textos que podem ser usados para proporcionar, por exemplo, anotações ou observações do operador.

O objeto Jornal que pode ser predefinido no servidor MMS, ou pode ser criado através do serviço *Create_Journal*, podendo ser específico de VMD ou de Associação de Aplicação. Seus atributos são:

Object: *Journal*
Key attribute: *Journal Name*
Attribute: *MMS Deletable*
Attribute: *List Of Entry Reference*

- Journal Name, identifica este objeto;

- MMS Deletable, indica se este objeto pode ser deletado, pelo serviço *Delete_Journal*, ou não;

- List Of Entry Reference, consiste de uma lista de referências a entradas de jornal reais onde informações são mantidas.

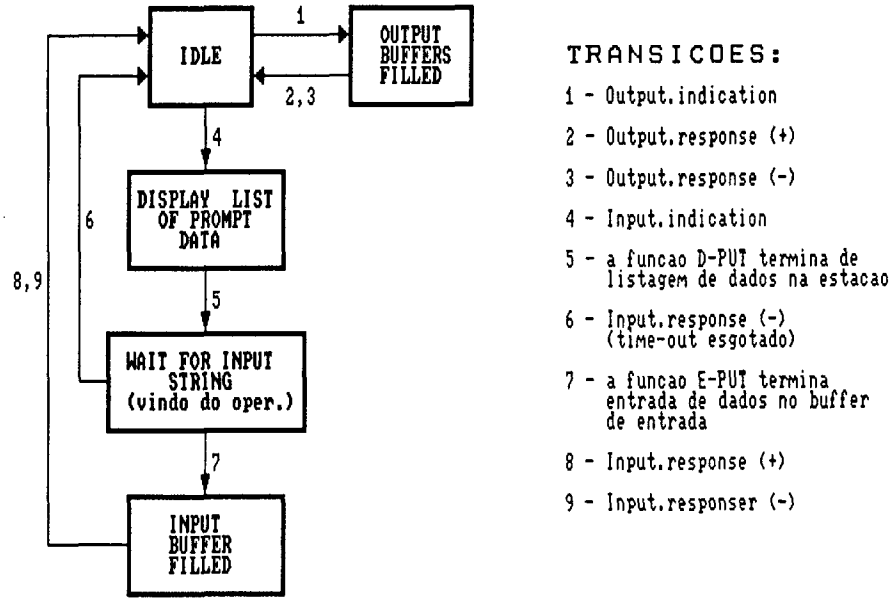


Fig. B.3 Máquina de Estados do Operador de Estação

ANEXO C

A LINGUAGEM DE ESPECIFICAÇÃO ESTELLE

Somente uma abordagem fundamentada sobre uma Técnica de Descrição Formal (TDF), permite conceber um sistema de maneira completa (com todas as funcionalidades requeridas) e isento de erros e ambiguidades. Outrossim, somente uma Técnica de Descrição Formal pode oferecer uma referência comum para a utilização de um mesmo protocolo sobre diferentes estações (geralmente heterogêneos) de uma rede [Courtiat 91]. Desta forma, pode-se afirmar que o uso de formalismo na especificação é especialmente importante quando relacionada à implementação de protocolos de comunicação.

Este anexo introduz alguns conceitos acerca de Estelle, uma linguagem para especificação de sistemas distribuídos com aplicação em sistemas de comunicação (protocolos e serviços). As características do dialeto Estelle* que adiciona o mecanismo de *rendez-vous* também são destacadas neste anexo. Por fim é apresentada a ferramenta ESTIM, desenvolvida para simular especificações escritas na linguagem Estelle*.

C.1 A Linguagem de Especificação Estelle

Em Estelle, uma especificação é modelada como um conjunto de máquinas de estado finito, que incorporam definições de tipo, expressões e comandos da linguagem de programação Pascal. Além destas características, Estelle fornece comandos próprios, destinados à criação da estrutura global da especificação.

Uma especificação Estelle consiste de uma estrutura hierárquica formada por **módulos**. A comunicação entre instâncias de módulos é feita a partir da definição de **pontos de interação** em ambas as instâncias. Dois pontos de interação conectados fornecem um elo de comunicação onde uma mensagem pode ser enviada de uma instância de módulo à outra.

A especificação Estelle é considerada dinâmica, na medida em que a estrutura formada pelas instâncias de modulos bem como a configuração das ligacões entre os pontos de interação destas instâncias, podem ser modificadas durante a execução.

I. Módulos Estelle

O comportamento interno de um módulo é descrito em termos de um "autômato de estados comunicante" cuja ação das **transições** é dada na forma de comandos da linguagem Pascal, salvo algumas restrições e extensões. Um módulo que possui ao menos uma transição é dito **ativo.**

Dentro da hierarquia fornecida por uma especificação Estelle, um módulo (denominado **módulo-pai)** pode ser substruturado em módulos pertencentes a um nível imediatamente inferior (denominados **módulos-filho**). A linguagem Estelle fornece uma relação de prioridade entre módulos pai/filhos, pela qual todas as transições dos módulos-filho ficam inibidas se uma das transições do seu módulo-pai está sensibilizada. A figura C.1 mostra a estrutura de uma especificação Estelle, nela, os módulos D e E estão num mesmo nível e são considerados módulos filhos do módulo B [Budkowski 87].

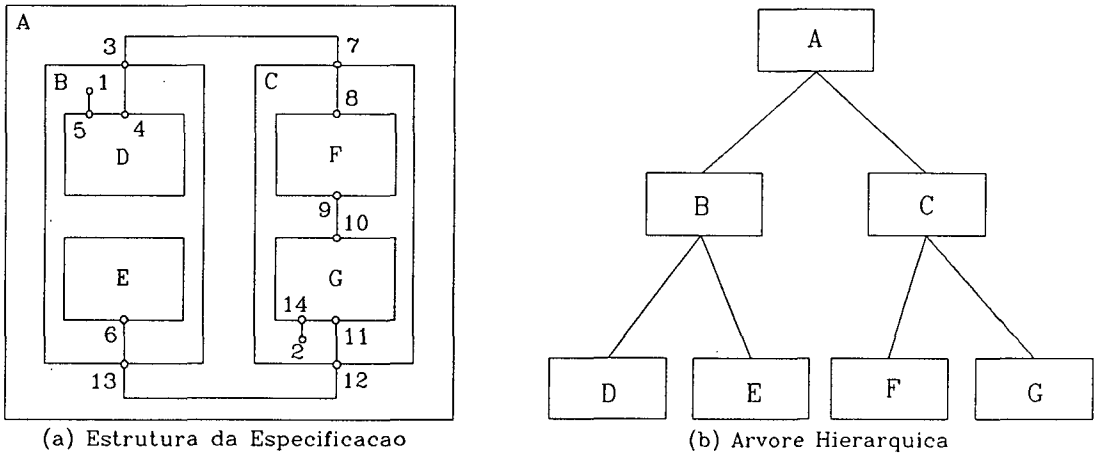


Fig. C.1 Estrutura de uma Especificação Estelle

Além da relação de prioridade pai/filhos, o comportamento interno de uma especificação Estelle é afetado pelo atributo que determina o tipo de cada módulo que compõe a especificação. Uma especificação Estelle é formada por um ou mais subsistemas (módulos do tipo SystemProcess ou SystemActivity), cada um destes podendo ser substruturado como um conjunto de módulos do tipo *Process* ou *Activity*.

Módulos atribuídos como *Process* ou *SystemProcess*, podem ser substruturados em módulos do tipo *Process* ou *Activity*; já os módulos atribuídos como *Activity* ou *SystemActivity*, só podem ser substruturados em módulos do tipo *Activity*.

Quando um módulo subsistema é atribuído como *SystemProcess* e seus módulos-filho como *Process*, todas as transições sensibilizadas fora do conflito pai/filho são selecionadas. Quando um módulo subsistema é atribuído como *SystemActivity* e seus módulos-filho como *Activity*, somente uma das transições sensibilizadas de um dos módulos-filho é selecionada. Um comportamento misto pode ser conseguido com o uso dos dois atributos, já que um subsistema do tipo *SystemProcess* pode ser substruturado como módulos do tipo *Process* e *Activity*.

Desta forma, o comportamento de uma especificação Estelle, que é definido como o conjunto de transições selecionadas a cada momento, depende da prioridade existente entre módulos pai/filhos e da maneira como os módulos de cada subsistema são atribuidos.

O paralelismo dentro de uma especificação Estelle é dado sob duas formas: o paralelismo dito **assíncrono**, que ocorre entre transições de diferentes subsistemas; e o paralelismo dito **síncrono**, que ocorre entre transições de diferentes módulos de um mesmo subsistema.

II. Pontos de Interação

Cada módulo da especificação Estelle pode ter um ou mais pontos de interação, os quais serviram para a troca de mensagem (envio/recepção) com outros módulos da especificação. Existem dois tipos de pontos de interação: internos e externos.

Ligações entre pontos de interação de módulos de um mesmo nível hierárquico são denominadas connect, ligações entre pontos de interação de um módulo-pai e um módulo-filho são denominadas attach. Na figura C.1, as ligações (3,4) e (11,12) são exemplos de *attach* e as ligações (3,7) e (9,10) são exemplos de *connect*.

A linha que liga dois módulos, por meio de um *connect* e zero ou mais *attaches*, é denominada link. Dois módulos Estelle ligados desta forma podem trocar mensagens em qualquer uma das direções, caracterizando uma comunicação fim-a-fim. Estelle utiliza o conceito de canal tipado (channel) para definir o tipo das mensagens que podem ser enviadas em cada *link*. Na figura C.1, a ligação dos pontos de interação 4,3,7,8 formam um *link* entre os módulos D e G [Budkowski 87].

Ao contrário do mecanismo de *Rendez-vous*, na comunicação por troca de mensagem fornecida pela linguagem Estelle, um módulo pode enviar uma mensagem a qualquer tempo. A mensagem, uma vez enviada, sempre é aceita e colocada numa fila FIFO associada ao ponto de interação do módulo receptor.

III. Compartilhamento de Variáveis

A linguagem Estelle fornece uma segunda forma de comunicação entre os módulos de uma especificação, onde uma ou mais informações podem ser trocadas através de variáveis compartilhadas. Tais variáveis só podem ser compartilhadas por um módulo-pai e seus módulos filhos.

A exclusão mútua no acesso da variável compartilhada é feita automaticamente através da relação de prioridade existente entre o módulo-pai e seus módulos-filho.

C.2 A Linguagem de Especificação Estelle*

Estelle* [Courtiat 88] é um dialeto da linguagem de especificação Estelle, onde se adiciona um mecanismo de Rendez-vous com a finalidade de exprimir de forma mais abstrata, as interações entre entidades de camadas adjacentes de um protocolo de comunicação.

As diferenças básicas entre a linguagem Estelle e Estelle* podem ser vistas na figura C.2.

	Estelle ISO	Estelle*
Attributo System	systemactivity systemprocess	systemactivity —
Atributo Module	activity process	activity —
Disciplina de Fila	individual queue common queue —	individual queue — no queue
Escala de tempo	—	irrelevante
Prioridades	clausula "priority" prioridade pai/filho	nao permitida desabilitada
Clausula rendezvous	— —	ip?interacao ip!interacao

Fig. C.2 Estelle* vs Estelle [Saqui 90]

O mecanismo de *rendez-vous* é associado às cláusulas IP?interaction e IP!interaction, que representam respectivamente, uma demanda de sincronização no receptor e no emissor. O ponto de interação deve ser necessariamente externo e desprovido de fila ([Courtiat 88] e [Courtiat 91]).

A fim de manter coerente o princípio de estruturação de Estelle, duas transições pertencentes a duas instâncias de módulos distintos não podem ser sincronizadas por *rendez-vous*, a menos que os módulos pertençam ao mesmo subsistema do tipo *SystemActivity*. Esta restrição é consequência do assincronismo total que deve existir entre subsistemas distintos.

C.3 O Simulador ESTIM

ESTIM (*Estelle Simulator based on an Interpretative Machine*) é uma ferramenta de validação capaz de simular a especificação formal de um sistema escrita em linguagem Estelle*. Pode-se dizer que ESTIM torna uma especificação Estelle* executável [Saqui 90a]. A figura C.3 mostra uma visão do ambiente ESTIM.

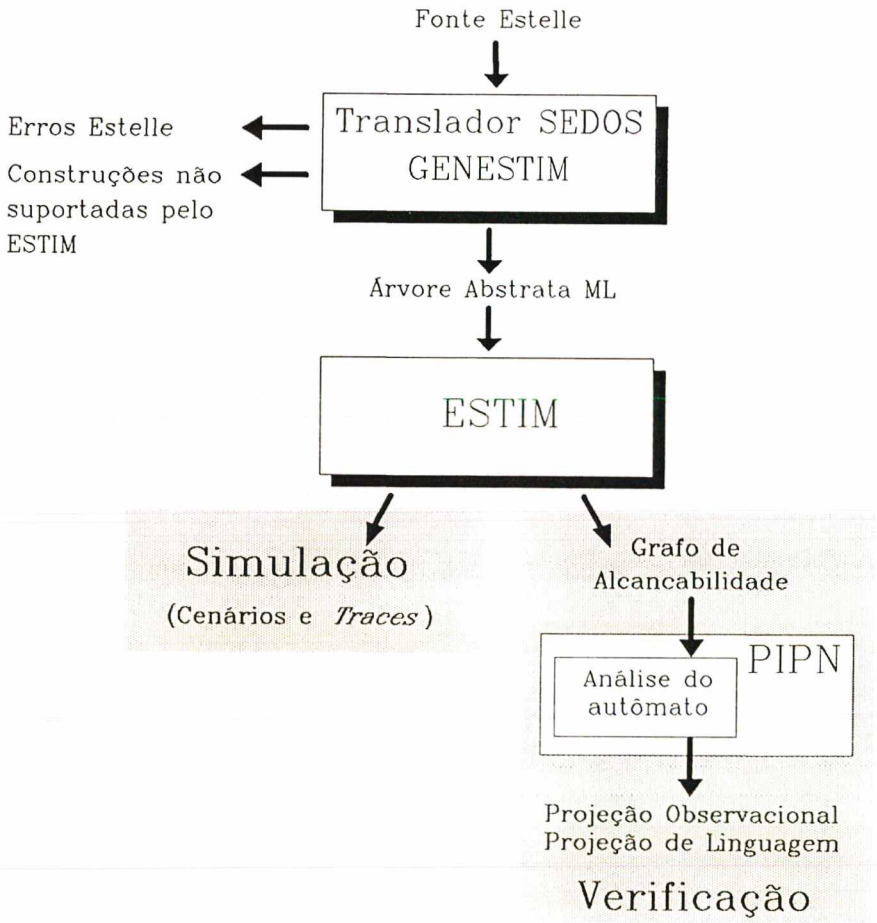


Fig. C.3 Ambiente ESTIM [Courtiat 91]

A simulação é uma estratégia de validação que explora alguns caminhos particulares do grafo de alcançabilidade. Durante a simulação, ESTIM age de maneira interativa com o usuário, fornecendo em cada estado, o conjunto de transições selecionadas (disparáveis) [Saqui 90b].

A verificação é outra estratégia de validação, onde examina-se todo o grafo de alcançabilidade. Neste caso, a ferramenta ESTIM fornece interfaces para as ferramentas de validação PIPN [Azema85] e ALDEBARAN [Fernandez 88].

A ferramenta ESTIM é capaz de manipular apenas um subsistema atribuído como *SystemActivity* e subestruturado como módulos do tipo *Activity*. Dentro deste subsistema único a comunicação pode ser assíncrona, através da fila FIFO de cada ponto de interação; e síncrona, através da abordagem de *rendez-vous* de Estelle*.

ANEXO D

OBJETOS MMS MAPEADOS EM ESTELLE

Neste anexo serão apresentados os critérios utilizados na especificação de cada um dos elementos do protocolo MMS em linguagem Estelle.

D.1 Mapeamento em Módulos Estelle

A opção pelo modelamento das entidades de protocolo através de módulos Estelle decorre da semelhança desta estrutura com àquela especificada no Padrão MMS através de objetos.

D.1.1 Mapeamento dos APs

Cada Processo de Aplicação foi descrito através de um Módulo Estelle contendo em seu corpo, os módulos-filho que descrevem cada uma das entidades (AEs, VMDs e Objetos MMS).

No caso do Processo de Aplicação cliente, a representação dos recursos através de VMD e Objetos MMS são substituidos pela máquina de estados que modela o cliente da aplicação (supervisor da célula), que contém as transições responsáveis pelo seu processamento. A figura D.1 mostra a máquina de estado que modela o comportamento do Processo de Aplicação supervisor da Célula Flexível de Usinagem.

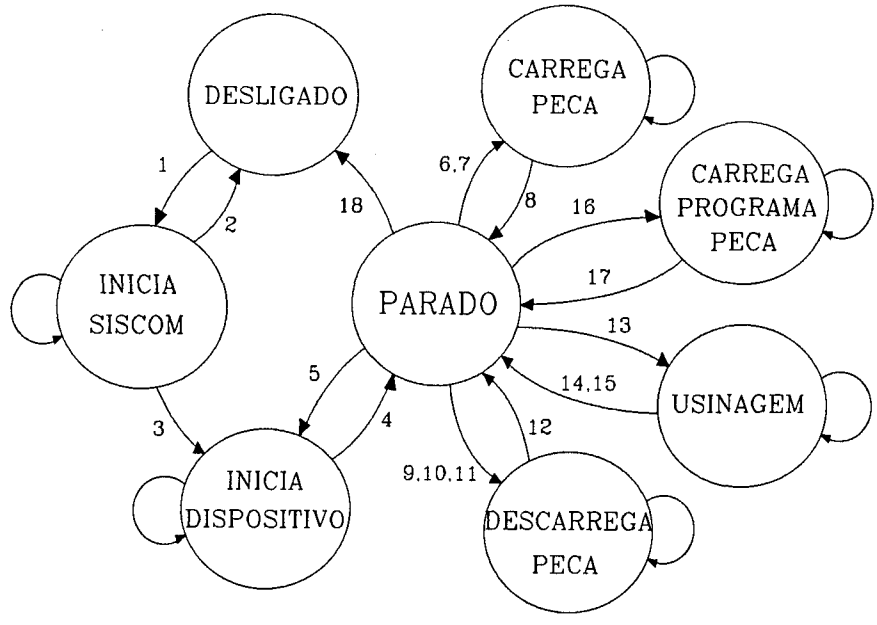
As variáveis deste módulo armazenam informações acerca do estado do supervisor e dos recursos manipulados por este, tais como peças, armazéns de peças, máquinas, etc.

D.1.2 Mapeamento das AEs

Cada Entidade de Aplicação foi descrita como um módulo contendo as transições que descrevem a máquina de estados do provedor MMS Mini-MAP (figura 2.10)

As máquinas de transação (serviços pendentes) são identificadas através de funções definidas no módulo. Estas funções manipulam uma variável indexada denominada

Transac[x] (onde "x" identifica a máquina de transação). Cada uma das variáveis contém o estado corrente da máquina do provedor para um determinado serviço pendente.



TRANSICOES PRINCIPAIS:

1. Conecta_Dispositivos
2. Erro_de_Conexao
3. Inicia_Dispositivos
4. Fim_da_Inicializacao
5. Inicia_Novo_Dispositivo
6. Carregar_Peca_Tipo_A
7. Carregar_Peca_Tipo_B
8. Confirma_Carga_CN
9. Transporta_Peca_Armazem_A
10. Transporta_Peca_Armazem_B
11. Transporta_Peca_Lixo
12. Confirma_Descarga_CN
13. Usinar_Peca
14. Termina_Usinagem
15. Aborta_Usinagem
16. Troca_Dominio_Prog_Peca
17. Cria_Nova_Invocacao_Prog
18. Fim_da_Aplicacao

Fig. D.1 Máquina de Estados do AP Supervisor

Outras variáveis deste módulo dão informações sobre o estado do usuário e se este está ou não correntemente associado a alguma AE remota.

D.1.3 Mapeamento dos VMDs

Cada VMD da aplicação foi descrito como um módulo Estelle contendo em seu interior os módulos-filhos que representam a Função Executiva e cada um dos objetos MMS necessários ao módulo AP servidor a que pertence o VMD. Os atributos do Objeto VMD são descritos como variáveis do módulo e podem ser alterados durante o processamento do VMD (objetos instanciados, nome de objetos, etc).

D.1.4 Mapeamento das Funções Executivas

O módulo Estelle que descreve a FE contém a máquina de estados cujas transições são encarregadas de tratar todos os problemas de instanciação e acesso aos objetos MMS contidos no Módulo Estelle VMD.

O módulo FE recebe, do módulo AE, as mensagens Estelle contendo as primitivas de Serviço MMS e envia as diretivas cabíveis ao Módulo Objeto MMS de interesse (não raramente, mais do que um).

As principais funcionalidades do módulo FE consideradas no exemplo da CFU são:

- Inicialização de contexto;
- Gerenciamento de Domínios;
- Gerenciamento de Invocações de Programa;
- Gerenciamento de Eventos;
- Gerenciamento de Variáveis;

A máquina de estados do módulo FE é mostrada na figura D.2.

D.1.5 Mapeamento dos Objetos MMS

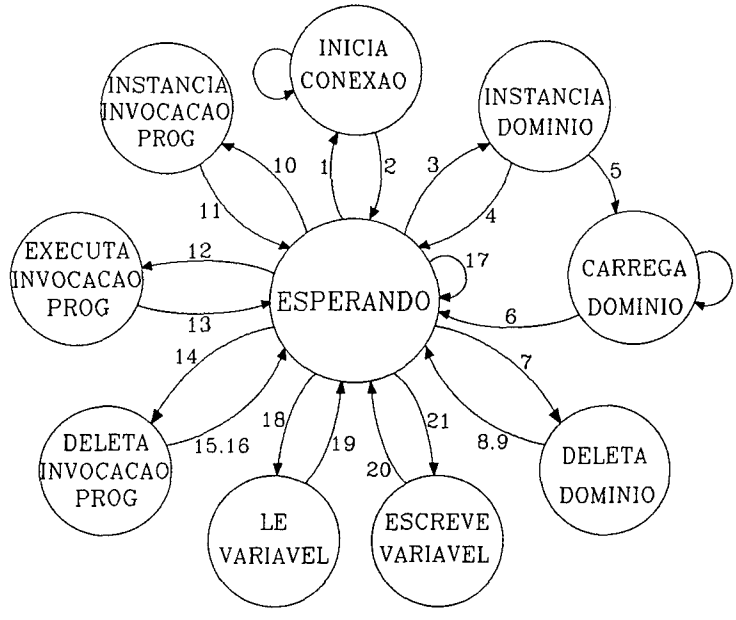
Cada Objeto MMS presente num VMD é descrito como um de seus módulos-filho e geralmente contém a máquina de estados que rege o comportamento do objeto (anexo B), juntamente com algumas variáveis de módulo que são associadas aos Atributos MMS especificados para o objeto.

No caso das relações entre módulos referentes aos objetos Domínio, Invocação de Programa e Condição de Evento, foram usadas transições dotadas de mecanismo *rendez-vouz* provido na linguagem Estelle*.

D.1.6 Mapeamento da Camada de Enlace

A simulação da camada de Enlace foi especificada como um módulo Estelle contendo um pequeno número de transições responsáveis pela transferência das PDUs de Enlace de um Processo de Aplicação a outro.

Não foram consideradas as possibilidades de erros de enlace e nem tão pouco o problema de endereçamento dos LSAPs.



TRANSICOES PRINCIPAIS:

1. Pedido_Conexao
2. Confirma_Inicializacao
3. Cria_Dominio
4. Erro_na_Criacao
5. Inicia_Seq_Carregamento
6. Termina_Seq_Carregamento
7. Deleta_Dominio
8. Aceita_Del_Dominio
9. Recusa_Del_Dominio
10. Cria_Invocacao_Prog
11. Confirma_Invocacao_Prog
12. Executa_Invocacao_Prog
13. Inicia_Execucao
14. Deleta_Dominio
15. Aceita_Del_Invocacao_Prog
16. Recusa_Del_Invocacao_Prog
17. Envia_Notificacao_Evento
18. Escreve_Variavel
19. Confirma_Escrita
20. Le_Variavel
21. Confirma_Leitura

Fig. D.2 Máquina de Estados do Módulo FE

D.2 Mapeamento em Canais Estelle

As trocas de mensagens entre os Processos de Aplicação e entre as entidades que os compõem, foi especificada através do mecanismo de interações fornecido pela linguagem Estelle.

Foram especificados três canais principais, cada qual com seu conjunto de interações e parâmetros associados. O Canal **CHN_ENLACE** especifica as mensagens tracadas entre Entidades de Aplicação e a Camada de Enlace. O Canal **CHN_MMS** especifica as mensagens trocadas entre Entidades de Aplicação e a Função Executiva do VMD, no caso de um AP servidor; e entre Entidades de Aplicação e o programa de aplicação do usuário no caso de um AP cliente. É neste canal que são especificados os

Serviços MMS e seus atributos. Por fim, o canal **CHN_FE** especifica as diretivas passadas pela Função executiva para controle dos Objetos MMS contidos no VMD.

Três outros canais foram especificados como *Rendez-vous* para permitir instanciação dinâmica de objetos e sincronização da máquina de estados de determinados objetos apartir da ocorrência de alguns eventos.

D.3 Mapeamento em Interações Estelle

Todas as mensagens trocadas dentro do ambiente MMS foram especificadas como Interações Estelle. Em especial o canal Estelle CHN_MMS define as interações que correspondem aos seguintes serviços:

- *Initiate*;
- *Conclude*;
- *Start*;
- *Read*;
- *Write*;
- *Initiate_Down_Load_Sequence*;
- *Down_Load_Segment*
- *Terminate_Down_Load_Sequence*;
- *Delete_Domain*;
- *Create_Program_Invocation*;
- *Delete_Program_Invocation*;
- *Event_Notification*;
- *Information_Report*.

Os atributos de cada um destes serviços foram definidos como uma estrutura do tipo *Record*. O valor de cada um dos atributos (variáveis do *Record*) é definido dentro da transição e enviado juntamente com a primitiva de serviço através de uma interação Estelle.

ANEXO E

CENÁRIOS DA SIMULAÇÃO

Este anexo visa mostrar as principais sequências de disparos usadas durante a simulação do sistema de comunicação da Célula Flexível de Usinagem.

As sequências de disparo são apresentadas segundo cada uma das operações especificadas para a célula, e as transições são identificadas pelo nome, módulo de origem e o dispositivo ao qual o módulo pertence. Os diagramas mostrados não relacionam, por questão de simplificação, os estados alcançados a cada disparo

Cenário de Estabelecimento dos Ambientes MMS

Durante a fase de conexão lógica, o Supervisor tentará estabelecer os ambientes de comunicação com cada um dos dispositivos (no caso da simulação, estes dispositivos são representados pelo Centro de Usinagem Horizontal e o Robô).

A figura E.1 mostra a sequência de disparos possíveis para as transições de conexão do supervisor com o centro de usinagem (para o robô, a sequência de disparos é semelhante). O supervisor emite o pedido de *Initiate.req* (transição Conecta_Disp) e a máquina de protocolo se encarrega de emitir o sinal de ressincronização *NULL* (Transições T1 e T2). A primitiva atravessa a camada de enlace (transição L1) e chega a Função Executiva do VMD do centro de usinagem, onde pode ser aceita ou recusada (transições Aceita_Conexão e Recusa_Conexão, respectivamente). A primitiva *Initiate.rsp* retorna ao supervisor, que reconhece a conexão do centro de usinagem através da transição CN_Conectado.

As transições T5 e T28 representam o reconhecimento imediato fornecido pela máquina de protocolo ao usuário local.

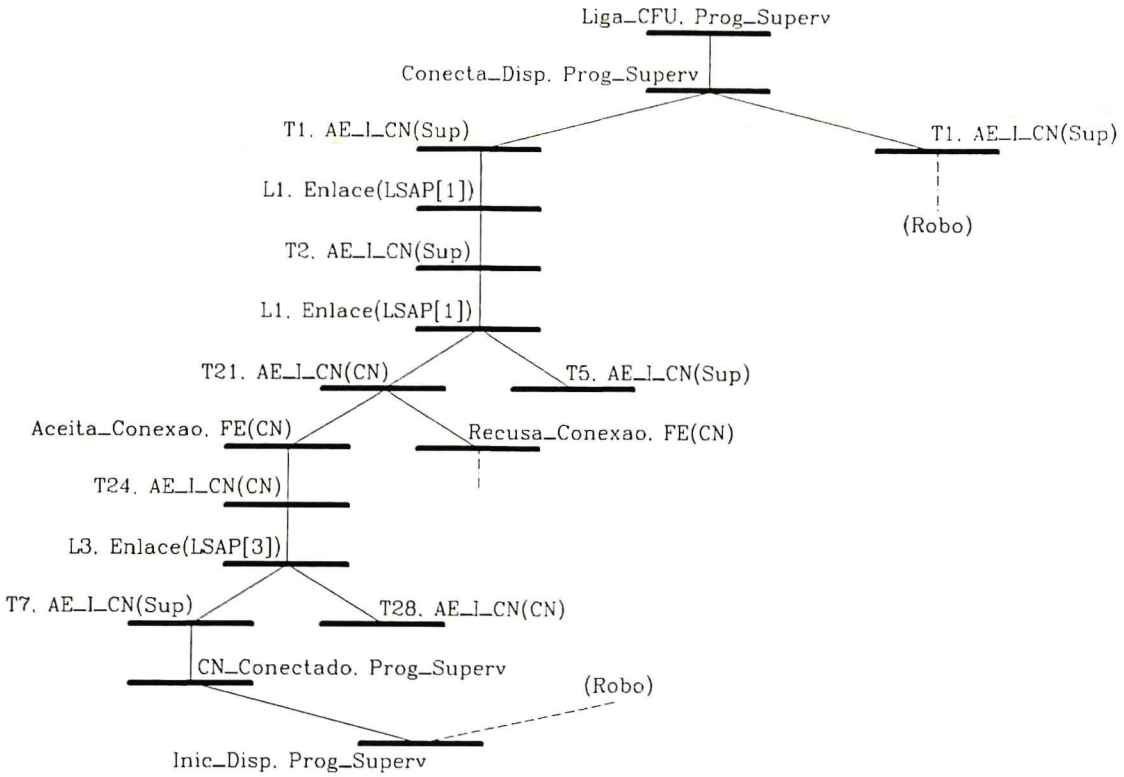


Fig. E.1 Transições de Conexão

Cenário de Inicialização dos Dispositivos

As figuras E.2 e E.3 apresentam as transições disparadas durante a inicialização do centro de usinagem. Na figura, as transições correspondem a alteração do valor da variável *N_Machine_Power*, carregamento de um domínio contendo um determinado programa-peça e a ativação de uma invocação de programa (os dois últimos objetos são instanciados em tempo de simulação).

Cenário de Carregamento de Peça a Usinar

Durante esta sequência de primitivas, o supervisor envia ao robó as primitivas para execução do programa de transporte (TRANS), que levará uma peça bruta de um armazém determinado (transições Carregar_Peça_A ou Carregar_Peça_B) até o centro de usinagem. As transições são apresentadas na figura E.4.

Cenário de Usinagem de Peça

A usinagem também foi considerada como execução de um programa ativado remotamente. As transições são apresentadas na figura E.5.

Na simulação foram considerados dois possíveis resultados da usinagem. O primeiro considera que a usinagem teve sucesso e neste caso, um evento é acionado para terminar a execução da invocação de programa (transição INV3/EVT1 com *Rendez-vous*, que manipula os objetos condição de Evento *End_Of_Program* e a invocação de programa N_PRG_x). O mesmo evento sinaliza ao supervisor, o sucesso da operação de usinagem. No segundo caso, considera-se uma falha na usinagem e uma primitiva de *Information_Report* é enviada ao supervisor (transição Informa_Def).

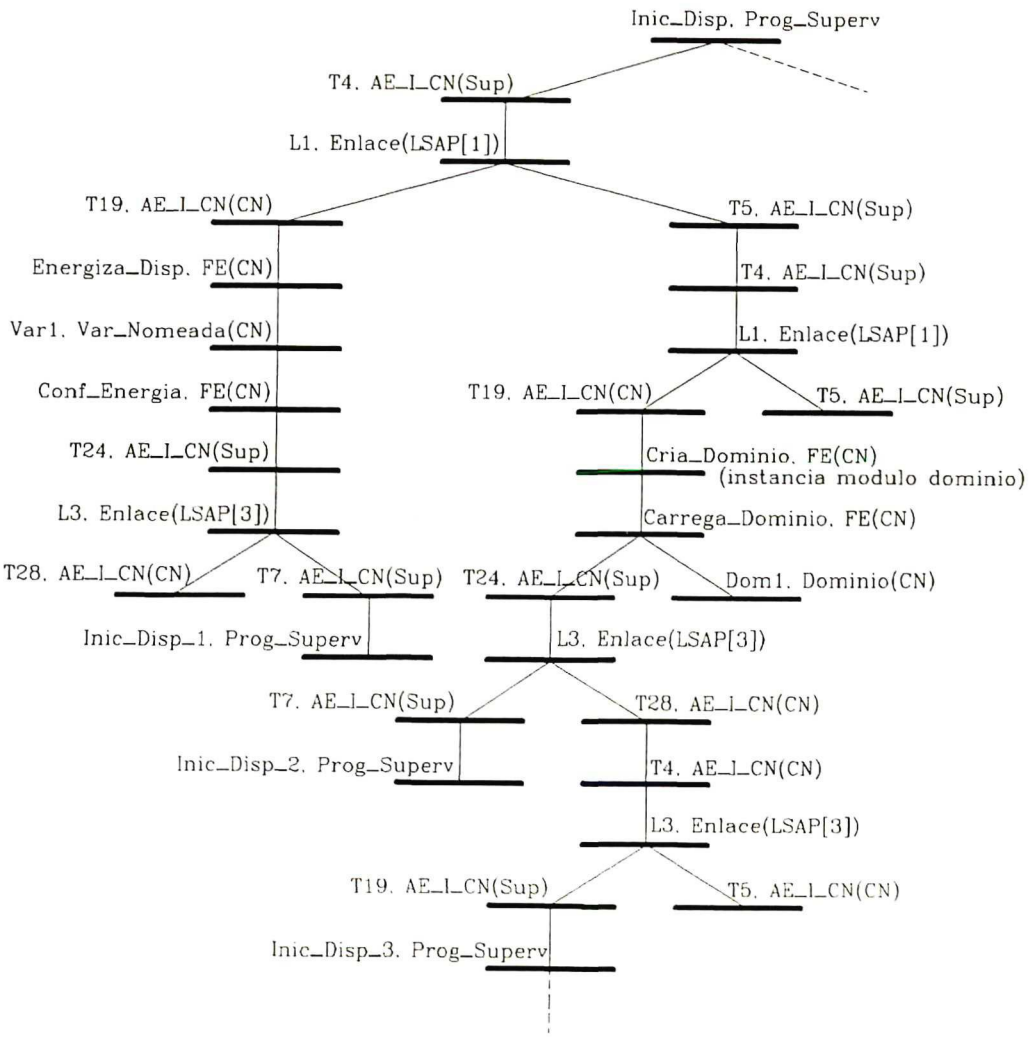


Fig. E.2 Transições de Inicialização

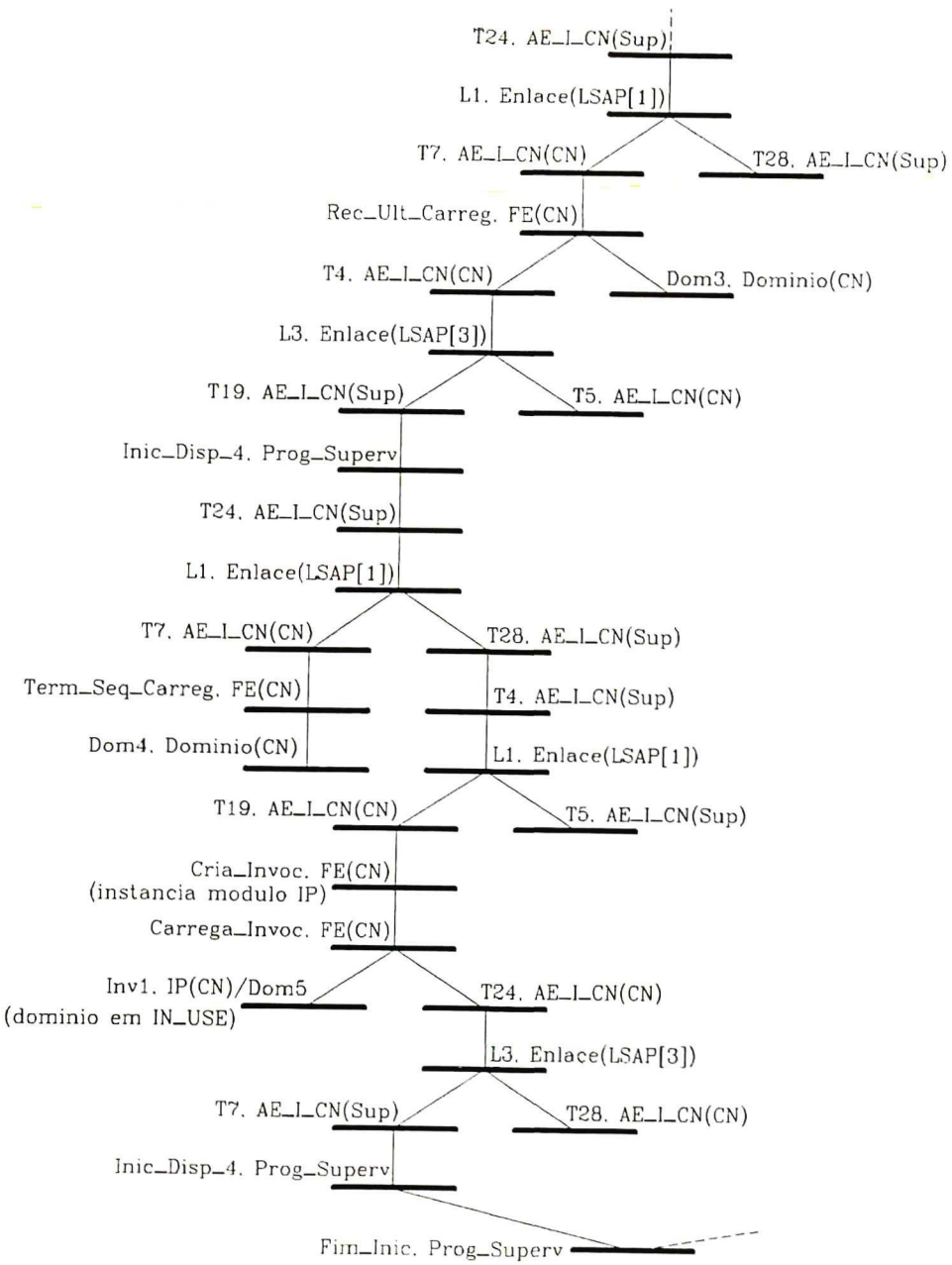


Fig. E.3 Transições de Inicialização
(continuação)

Cenário de Descarregamento de Peça Usinada

Esta sequência de transições é semelhante a apresentada para a operação de carregamento de peça bruta no centro de usinagem. A figura E.6 apresenta as transições disparadas durante a operação de retirada de peça do centro de usinagem.

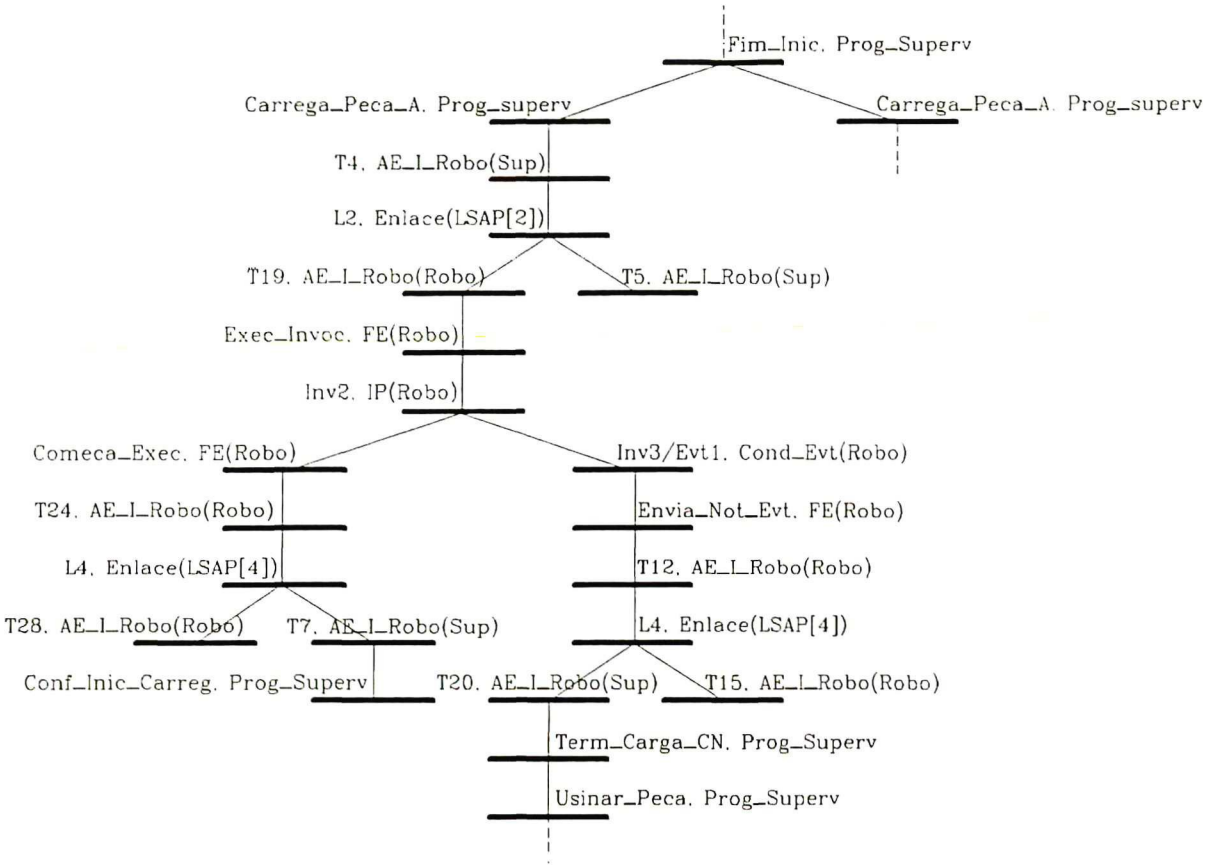


Fig. E.4 Transições de Carregamento de Peça

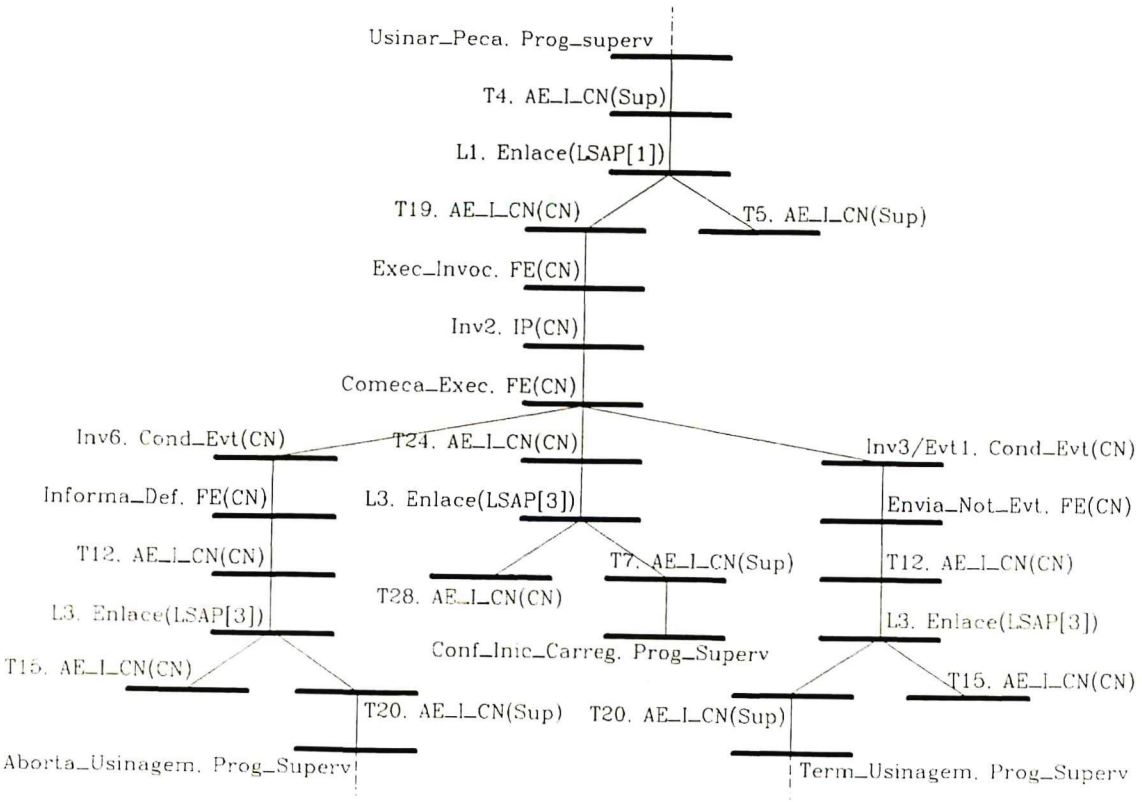


Fig. E.5 Transições de Usinagem de Peça

Cenário de Troca de Programa-Peça

Durante esta operação, o supervisor pedirá a liberação dos objetos invocação de programa e domínio (transições Del_Invoc e Del_Domínio). Após a liberação destes objetos do VMD do centro de usinagem, começa o procedimento de carga do domínio que contém o novo programa-peça e da invocação que o controla (transição Carrega_Novo_Prog). As figuras E.7 e E.8 apresentam o conjunto de transições disparadas durante a operação de troca de programa-peça.

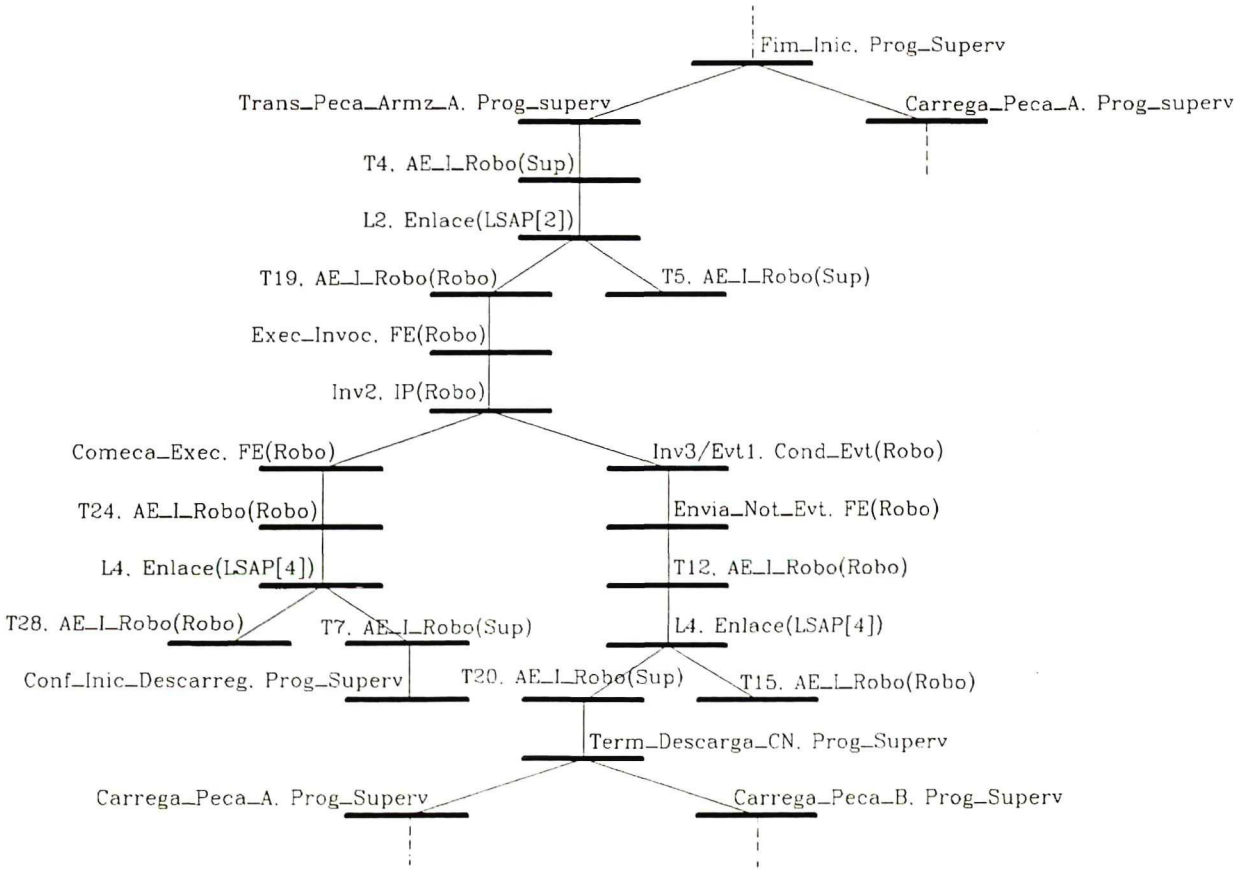


Fig. E.6 Transições de Descarregamento de Peça

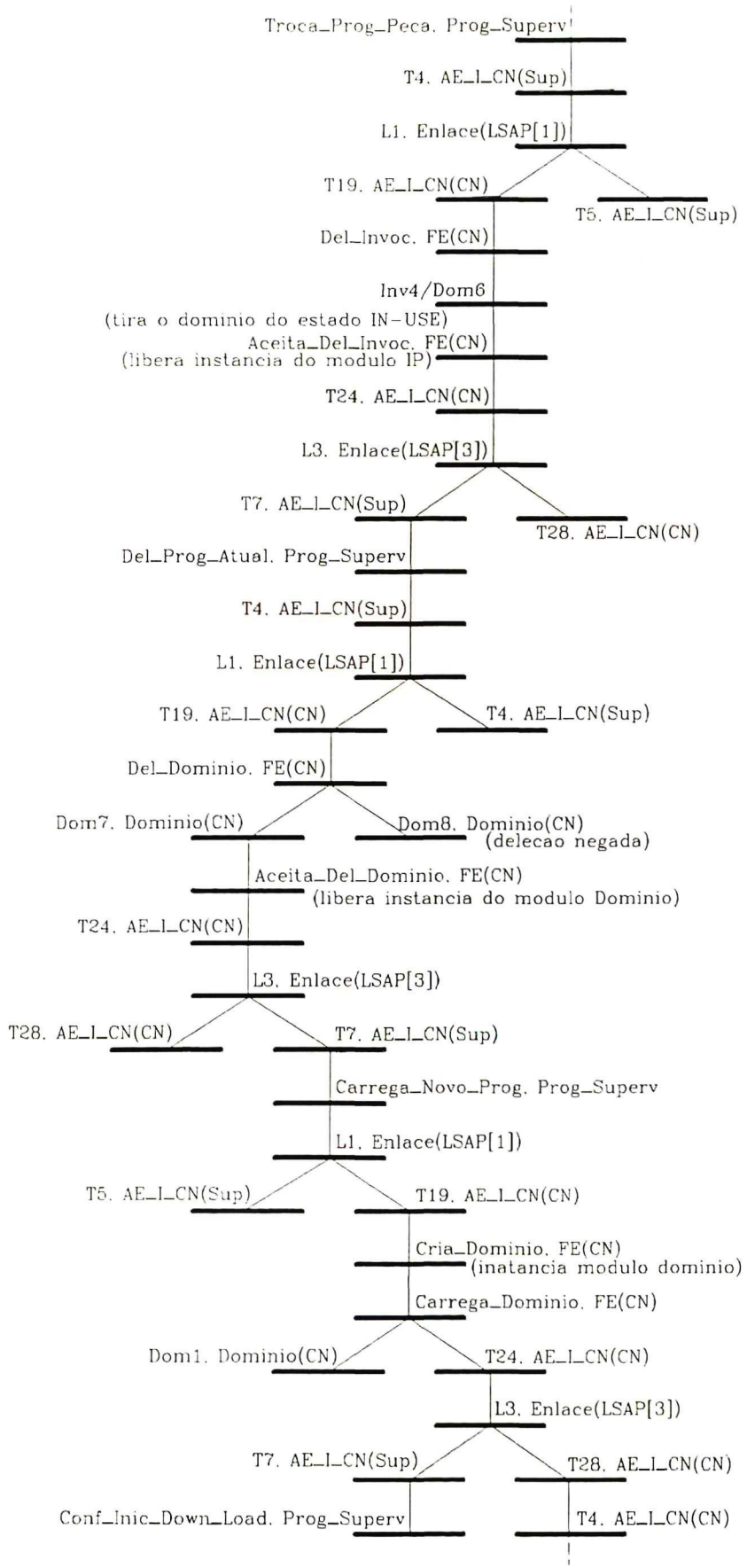


Fig. E.7 Transições de Troca de Programa-Peça

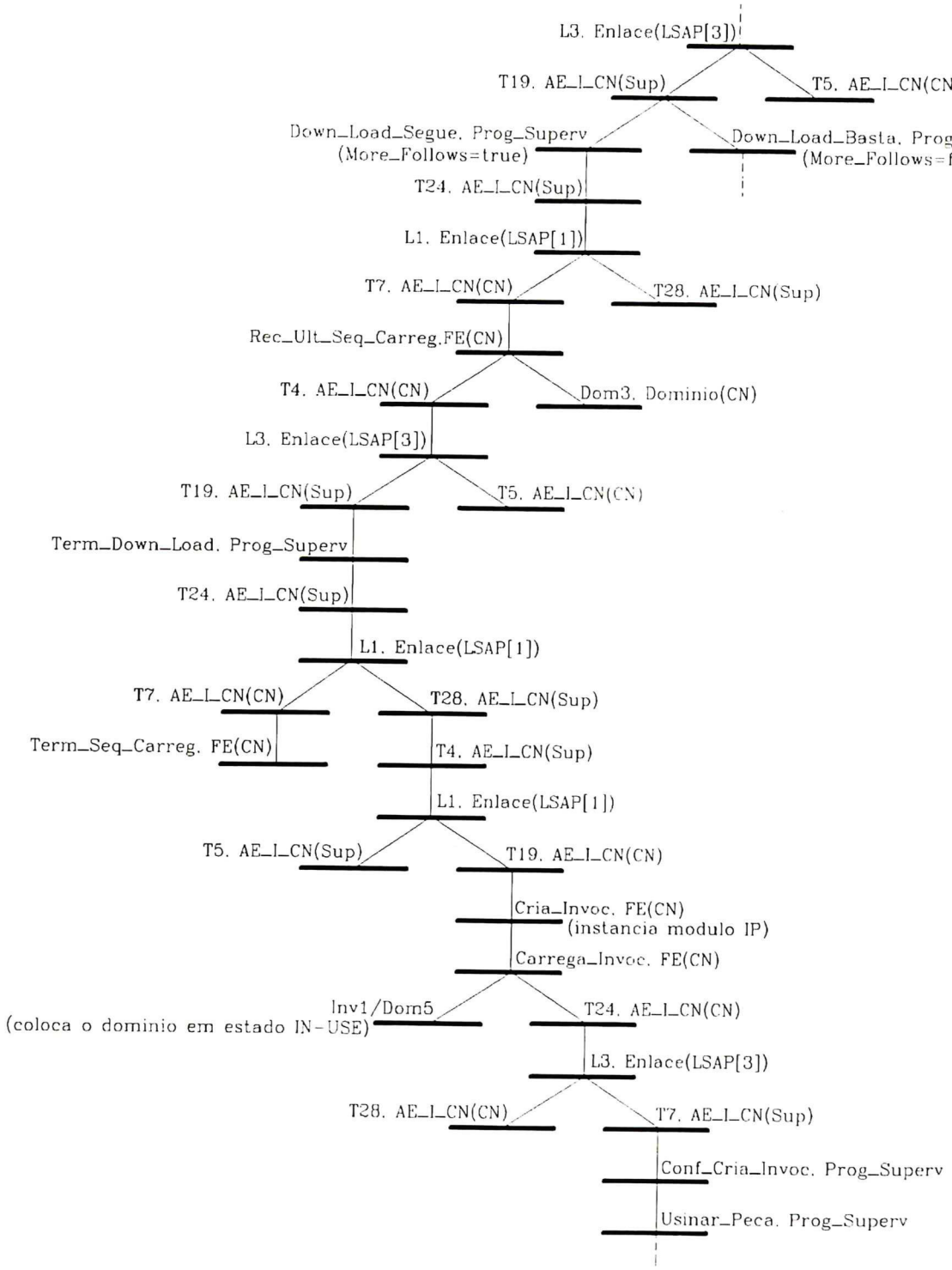


Fig. E.8 Transições de Troca de Programa-Peça
(continuação)