UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

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 DE GRAU DE MESTRE EM ENGENHARIA ELÉTRICA

ROBERTO WILLRICH

FLORIANÓPOLIS, ABRIL - 1991

UMA PROPOSTA DE UM MODELO DE IMPLEMENTAÇÃO DE SERVIÇOS DE APOIO A APLICAÇÕES INDUSTRIAIS SEGUNDO O PADRÃO MMS

ROBERTO WILLRICH

ESTA DISSERTAÇÃO FOI JULGADA PARA A OBTENÇÃO DO TÍTULO DE

MESTRE EM ENGENHARIA ELÉTRICA

ESPECIALIDADE ENGENHARIA, 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

Profa. Liane Tarouco, Dra.

Profa. Stephania Stiubiener, Dra.

AGRADECIMENTOS

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

Aos membros da Banca Examinadora, pelos comentários e sugestões feitas a esse trabalho.

A todos os professores, colegas e funcionários, do Programa de Pós-Graduação em Engenharia Elétrica e do Laboratório de Controle e Microinformática (LCMI/DEEL), que de uma maneira ou de outra contribuiram para a realização deste trabalho.

A minha família, em especial aos meus pais, Günther Willrich Júnior e Aurea M. Willrich pelo apoio e compreensão.

A UFSC e a CAPES pelo apoio financeiro que permitiu a realização deste trabalho.

SUMÁRIO

RESUMO

A presente dissertação tem por objetivo apresentar uma proposta de um modelo de implementação do padrão MMS (*Manufacturing Message Specification*) numa Arquitetura Mini-MAP, de acordo com o padrão da Interface de Programa de Aplicação MMS (MMSI). Dentro deste contexto, são analisados alguns aspectos relacionados à utilização dos padrões MMS e MMSI na arquitetura Mini-MAP.

Com intuito de apresentar o contexto geral deste trabalho, são apresentadas noções gerais sobre CIM (*Computer Integrated Manufacturing*) e os principais conceitos dos padrões de comunicação RM-OSI, MMS e MAP.

A seguir, é proposto um modelo de implementação do padrão MMS numa arquitetura Mini-MAP.

A validação do modelo e das decisões de implementação propostas foi realizada a partir da simulação de uma especificação Estelle. O teste do modelo proposto numa implementação real (apesar de simples) permitiu discutir algumas particularidades da implementação e definir as características de um Núcleo de Tempo Real como suporte para a implementação. Dos resultados obtidos, conclui-se na adequação do modelo proposto, garantindo a correção das futuras implementações do MMS numa arquitetura Mini-MAP.

ABSTRACT

This work aims to the presentation of an implementation model proposal for the Manufacturing Message Specification (MMS) standard in a Mini-MAP architecture according to the MMS Application Program Interface standard (MMSI). In this context, aspects on the utilization of the MMS and MMSI standards in a Mini-MAP architecture are discussed.

Firstly, general notions about Computer Integrated Manufacturing (CIM) are presented together with the main concepts on RM-OSI, MAP e MMS communication standards.

Afterwards, an implementation model for the MMS standard in a Mini-MAP architecture according to the MMS Application Program Interface standard (MMSI) is proposed.

The validation of both the model and the implementation decisions proposed was done by means of the simulation of an Estelle specification. The test of the proposed model in a real implementation (even though a simple one) allowed the discussion of some particularities of the implementation. Also, it permitted the definition of the features for a Real Time Kernel designed to support such an implementation. From the final results, one can conclude by the adequacy of the proposed model assuring the correctness of the future implementations of the MMS standard in a Mini-MAP architecture.

GLOSSÁRIO

Termos definidos no OSI e MAP:

AA - *Application Association* .
ACSE - *Association Control Service Element*
AE - *Application Entity*
AE-I - *Application Entity Invocation*
AP - *Application Process*
API - *Application Program Interfaces*
AP-I - *Application Process Invocation*
ASE - *Application Service Element*
ASN.1 - *Abstract Sintax Notation one*
CBB - *Conformance Building Block*
CLNP - *ConnectionLess Network Protocol*
CSMA/CD - Acesso Múltiplo com Detecção de Portadora e Detecção de Colisão
CSP - *Confirmed Service Provider*
DCB - *Data Control Block*
Dir_ava - *Directory Attribute Value Assertions*
Dir_dn - *Directory Distinguished Name*
Dir_rdn - *Directory Relative Distinguished Names*
DUA - *Directory User Agent*
EPA - *Enhanced Performance Architecture*
ES-IS - *End Systems - Intermediate Systems*
FTAM - *File Transfer, Access and Management*
HLS - *High Level Service*
HLSP - *High Level Service Provider*
IFA - *Indication Filter & Arbitrator*
ISO - *International Organization for Standardization*
LAPB - *Link Access Procedure Balance*
LLC - *Logical Link Control*
LLS - *Low Level Service*
MAC - *Medium Access Control*
MAP - *Manufacturing Automation Protocol*
MMPM - *Manufacturing Message Protocol Machine*
MMS - *Manufacturing Message Specification*
MMSI - *Manufacturing Message Specification Interface*

OSI - *Open System Interconnection*
PDU - *Protocol Date Unit*
PLP - *Packet Level Protocol*
PSAP - *Presentation Service Access Point*
PSP - *Primitive Service Provider*
RM-OSI - *Reference Model - Open System Interconnection*
RS - *Responder Service*
ROS - *Real Open System*
TOP - *Technical and Office Protocol*
VMD - *Virtual Manufacturing Device*
VT - *Virtual Terminal*
X - Serviço MMS qualquer

Outros termos:

ADES - Ambiente de Desenvolvimento e Execução de Software - LCMI
CAD - *Computer Aided Design*
CAE - *Computer Aided Engineering*
CAM - *Computer Aided Manufacturing*
CAPP - *Computer Aided Process Planning*
CIM - *Computer Integrated Manufacturing*
CNMA - *Communication Network for Manufacturing Application*
CP - Controlador Programável
CS - *Companion Standard*
ESPRIT - *European Strategic Program for Research and Development in Information Tecgnology*
ESTIM - *Estelle Simulator based on an Interpretative Machine*
FMC - *Flexible Manufacturing Cell*
FMS - *Flexible Manufactiring System*
IEC - *International Electrotechnical Commission*
IEEE - *Institute of Electrical and Electronics Engineers*
IPC - *Inter-Proccess Communication*
ISA - *Intrument Society of America*
LAAS-CNRS - *Laboratoire d'automatique et d'analyse des Systemes - Centre National de la Recherche Scientifique.* Toulouse (France);
LCMI - Laboratório de Controle e Micro-informática - UFSC
MHS - *Material Handling System*
MRP - *Manufacturing Resource Planning*

NC - *Numerical Control*
NTR - Núcleo de Tempo Real
P&D - Pesquisa & Desenvolvimento
PIPN - PROLOG *Interpreted Predicate Net*
POSIX - *Portable Operation System Interface*
SEDOS - *Software Environment for the Design of Open Systems*

CAPÍTULO 1

INTRODUÇÃO

A industria manufatureira está passando nestes últimos anos por grandes mudanças devido ao surgimento de novas tecnologias informatizadas em todos os níveis do processo de manufatura. Na maior parte das vezes, estas novas tecnologias vem sendo empregadas em setores isolados da empresa, visando unicamente implementar funções específicas deste. O compartilhamento de informações nestes diversos setores permite uma melhor coordenação do processo de manufatura, aumentando a eficiência deste e então a sua produtividade.

A filosofia de integração dos sistemas informatizados dentro de uma indústria é chamada de CIM (*Computer Integrated Manufacturing*). As redes de comunicação e base de dados são os meios que facilitam a troca e o compartilhamento da informação. Entretanto a incompatibilidade entre dispositivos de fabricantes e tecnologias diferentes é uma dificuldade para a garantia desta integração.

A necessidade de garantir a compatibilidade na troca de informações entre todos os equipamentos e sistemas utilizados na empresa é uma das causas do esforço efetuado pela comunidade internacional no sentido de definir e utilizar padrões de comunicação. Neste sentido, foram elaborados os padrões MAP (*Manufacturing Automation Protocols*) [MAP 88] e TOP (*Technical and Office Protocols*) [TOP 88]. Estes padrões seguem a arquitetura do modelo de referência OSI da ISO [ISO 84]. Entretanto, para aplicações envolvendo tempos críticos, é também proposta para ambientes de automação da manufatura, uma arquitetura reduzida a três camadas (física, enlace e aplicação) chamada Mini-MAP.

Um dos elementos de serviço da camada de aplicação da proposta MAP, chamado MMS (*Manufacturing Message Specification*) [ISO 88] é especialmente dedicado ao fornecimento de serviços para a transmissão de mensagens entre equipamentos programáveis em ambientes fabris. O padrão MMS [ISO 88] especifica a sintaxe e a semântica gerais das mensagens na manufatura, não contendo informações específicas do tipo de aplicação. Estas informações são fornecidas pelos diversos *Companions Standards* ([IEC 89], [ISA 90], [ISO 89a] e [ISO 89b]).

Para padronizar também a forma de utilização dos serviços MMS, o padrão MAP [MAP 88] propõe uma Interface para o Programa de Aplicação MMS, chamada MMSI.

A presente dissertação visa propor um modelo de implementação do padrão de comunicação MMS numa arquitetura Mini-MAP, de acordo com o padrão da MMSI.

O capítulo 2 apresenta os principais conceitos existentes no ambiente CIM.

No capítulo 3 serão apresentados alguns dos principais conceitos e definições utilizados nas normas RM-OSI [ISO 84], MMS ISO [ISO 88] e MAP [MAP 88]. São também levantados alguns aspectos que dizem respeito ao uso dos serviços MMS e da interface MMSI numa arquitetura Mini-MAP.

No capítulo 4 será descrito o modelo proposto para a implementação do padrão MMS numa arquitetura Mini-MAP, de acordo com o padrão da MMSI. Em seguida será apresentada a especificação deste modelo na linguagem Estelle e os resultados referentes a validação desta a partir da simulação da especificação Estelle.

O capítulo 5, com o intuito de ilustrar detalhes de implementação, apresenta um exemplo de implementação de alguns serviços utilizando o modelo proposto. No decorrer deste capítulo, serão definidas algumas características necessárias a um núcleo de tempo real a ser utilizado na implementação segundo o modelo proposto.

CAPÍTULO 2

CIM - MANUFATURA INTEGRADA POR COMPUTADOR

2.1 Introdução

A evolução dos sistemas informatizados tem possibilitado uma maior integração das tarefas de uma empresa, chegando-se ao final, num futuro próximo, a integração completa da empresa. Esta integração dos sistema informatizados dentro de uma empresa é denominada "CIM" (*Computer Integrated Manufacturing*). Neste capítulo serão apresentados aspectos gerais sobre o CIM, visando posicionar, dentro deste contexto, o trabalho sobre comunicação apresentado nesta dissertação.

2.2. Aspectos Gerais sobre Manufatura Integrada por Computador

Para que uma empresa manufatureira possa manter-se no mercado, ela deve ter uma estrutura que suporte as condições atuais, em particular no que diz respeito ao contínuo aumento dos custos de fabricação e a redução do ciclo de vida dos produtos. Sendo assim, a empresa deve ter uma estrutura capaz de alterar rapidamente as características do produto e o ritmo de produção, a fim de acompanhar as demandas de mercado. Para isto, torna-se fundamental a existência de um parque fabril composto de equipamentos com grande flexibilidade.

No passado, o requisito de flexibilidade reduzia, em muito, a produtividade das empresas que necessitavam desta característica para sua sobrevivência. A evolução das novas tecnologias nas áreas de computadores, microeletrônica, máquinas a comando numéricos entre outros, permitiu automatizar ações destas empresa, sem muitas perdas na flexibilidade. Esta automação da produção possibilitou o surgimento dos Sistemas de Manufatura Flexível (FMS). Neste tipo de instalação, a flexibilidade está na utilização de máquinas rapidamente reconfiguráveis, que oferecem a possibilidade de realizar peças diferentes num encadeamento rápido, aumentando a produtividade.

Na formação de um FMS, o caminho mais empregado é a utilização de células flexíveis de manufatura (FMC) individuais com possibilidade de substituição mútua, onde uma célula poderia substituir outra. Em cada célula se realiza uma série de operações sobre os elementos do produto a fabricar (peças), como por exemplo usinagem, montagem, solda e pintura. Um FMS situa-se em um nível hierárquico superior as FMC, englobando a coordenação das diversas FMC e garantindo-lhes o fornecimento de peças, insumos, ferramentas, programas e demais acessórios, de acordo com as suas necessidades [Lepikson 90].

Juntamente com o FMS, nos diversos setores da empresa surgiram vários sistemas informatizados de apoio a manufatura, como CAD/CAE (*Computer Aided Design/Computer Aided Engineering*), CAP (*Computer Aided Planning*), CAPP (*Computer Aided Process Planning*). Inicialmente, estes sistemas eram inseridos na empresa de forma independente, para satisfazer as necessidades funcionais de tarefas específicas. Esta política de automatização desvinculada resultou em áreas funcionais incapazes de se comunicarem entre si, chamadas de "ilhas de automação". Devido a este fato, o fluxo de informações entre as diversas áreas funcionais da empresa era lento, reduzindo a eficiência da produção.

A eficiência na troca de informações nos diversos setores de uma empresa pode ser aumentada através das tecnologias de redes de comunicação e base de dados distribuídos. Estas interligações permitirão a tomada de decisões mais rápidas em todos os níveis da empresa, acarretando a redução dos custos de produção a partir da utilização eficiente do tempo, material, trabalho e energia. Também possibilitarão rápidas mudanças nas características do produto em fabricação, permitindo que a empresa acompanhe as demandas de mercado. O conceito de integração dos diversos sistemas baseados em computador numa empresa é chamado de CIM (*Computer Integrated Manufacturing*).

2.3 Níveis Hierárquico do CIM

Para dar suporte ao CIM, é necessário a definição de uma hierarquia organizacional. Na literatura, várias propostas foram definidas ([ISO 87a], [McLean 86], [Williams 90]). A seguir será apresentado o modelo padrão da ISO para automação fabril ([ISO 87a], [McGuffin 88]), que divide a empresa segundo seis níveis hierárquicos:

I. Nível Empresa

As funções básicas deste nível são a de gerenciamento da corporação, assuntos financeiros, marketing, vendas, pesquisas e desenvolvimentos.

II. Nível Instalação

O nível Instalação é responsável pela implementação das funções do nível empresa. Nele são realizadas as funções seguintes: planejamento da produção, projeto e planejamento do processo, através de ferramentas de auxílios automatizadas, como CAP, CAPP e CAD/CAE.

III. Nível Seção

O nível Seção é responsável pela coordenação da produção, pelo suporte dado as tarefas do parque fabril e pela alocação de recursos a estas tarefas. Neste nível existem duas funções básicas: gerenciamento de tarefas e de recursos. A primeira, em função de uma produção específica, escalona as ordens de trabalho, manutenção de equipamento, serviços de suportes da planta. Enquanto a segundo aloca estações de trabalho, áreas de armazenamento, ferramentas e materiais para sistemas de controle a nível de célula, também em função de uma produção específica. Assim, este nível coordena a produção e atividades de suporte que serão executadas pelos controladores de célula no nível a seguir. Ferramentas assistidas por computador como CAM (*Computer Aided Manufacturing*) são utilizadas para auxiliar na execução das funções deste nível.

IV. Nível Célula

No nível Célula, o plano de produção já está separado e alocado em cada célula. Cada célula suporta um controlador, responsável pelo controle de execução do plano de produção e por vários outros serviços de suporte, tais como sistema de manuseio de material (MHS) e calibração.

V. Nível Estação

Cada controlador de célula monitora um conjunto de estações (robôs, máquinas-ferramenta, esteiras, etc.) que são responsáveis pela sequência de operação

necessários para completar as ações do plano de trabalho, alocadas para este controlador de célula particular.

Cada estação possui um controlador de estação, que é responsável pela coordenação das operações a este nível, podendo ser um NC (Controlador Numérico), CP (Controlador Programável), microcomputador, etc.

VI. Nível Equipamento

São dispositivos, como acionadores, válvulas, relés, e outros, responsáveis pela execução dos comandos que lhes foram emitidos pelo controlador de estação.

Nesta estrutura, o fluxo de informação ocorre em dois sentidos: dados de monitoração de processo, informações de status da produção e indicadores de exceção sobem para os níveis altos da hierarquia, em cada nível, o volume de informação é processado, gerando sinais de controle e ajustes que descem para os níveis baixos da hierarquia.

Os requisitos de comunicação não são os mesmos em cada nível da hierarquia organizacional do CIM. Os requisitos de tempo real nas baixos níveis da hierarquia são mais pronunciados que aqueles dos níveis mais altos. Os baixos níveis são caracterizados por mensagens frequentes e curtas, transmitidas sobre distâncias curtas, como atribuições de controle, medidas e outras. As mensagens nos níveis mais altos incluem normalmente transferência de grandes arquivos e interações humanas de entrada e saída. A figura 2.1 mostra estas características.

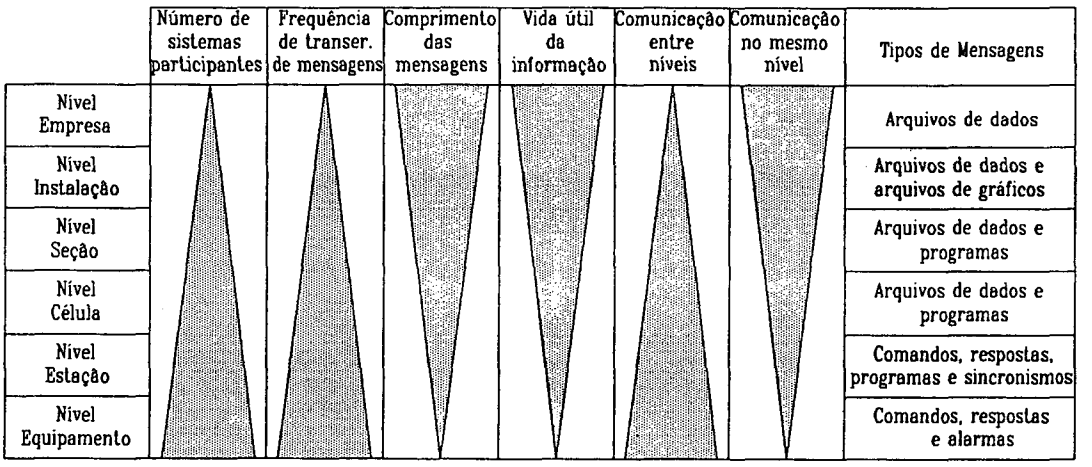


Figura 2.1 Características de comunicação no ambiente CIM [Leite 87].

2.4. A Comunicação no CIM

Um problema na integração completa da empresa é a falta de conectividade entre os vários equipamentos. A maior causa das ilhas de automação e das lacunas no compartilhamento de informações é a incompatibilidade de equipamentos fornecidos por diferentes fabricantes ([Stewart 88]). O uso de padrões internacionais de comunicação é atualmente a solução mais adequada a ser adotada pelos fabricantes quando da integração de meios computacionais e de equipamentos de tecnologias diferentes.

Devido a diversidade de requisitos de comunicação numa empresa, como mostra a figura 2.1, há a necessidade de estabelecimento de mais de um padrão de comunicação, afim de satisfazer requisitos específicos da sua área de aplicação. A figura 2.2 mostra os vários tipos de redes de comunicação dentro de um ambiente CIM.

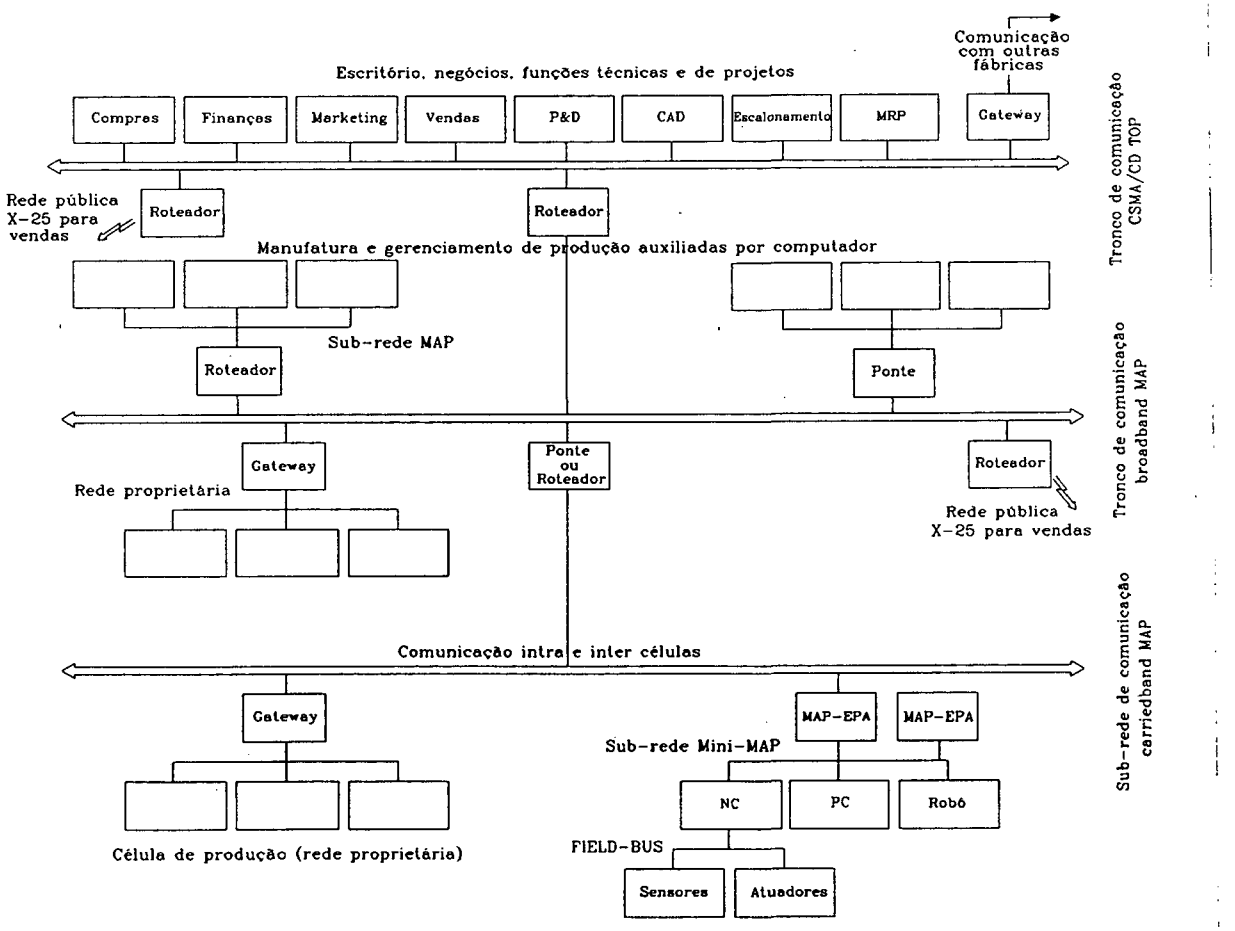


Figura 2.2 Padrões de Comunicação no CIM [Thacker 88]

Os projetos MAP (*Manufacturing Automation Protocols*) [MAP 88] e TOP (*Technical and Office Protocols*) [TOP 88] foram os primeiros passos concretos para a integração dos vários sistemas computacionais dentro de uma empresa. Também a nível de barramento de campo (Field-Bus) um esforço no sentido da padronização esta sendo feito. A seguir, estes padrões de comunicação serão apresentados.

2.4.1 Padrões MAP/TOP

Criado pela General Motors, o MAP [MAP 88] (fig. 2.3b) é uma seleção de protocolos OSI [ISO 84] (fig. 2.3a) apropriados para aplicações em ambientes fabris. O TOP [TOP 88] (fig. 2.3c) foi desenvolvido pela Boeing Computer Services, também baseado nos protocolos OSI e destina-se aos departamentos de engenharia, contabilidade, marketing e outros.

Um dos elementos de serviço da camada de aplicação do MAP é o MMS (*Manufacturing Message Specification*) [ISO 88], que destina-se a dar ao MAP a capacidade de prover serviços para a transmissão de mensagens entre equipamentos programáveis no ambiente de Manufatura e Controle de Processos (NC, CP, etc.), permitindo o acesso a dados e execução de controle em dispositivos remotos.

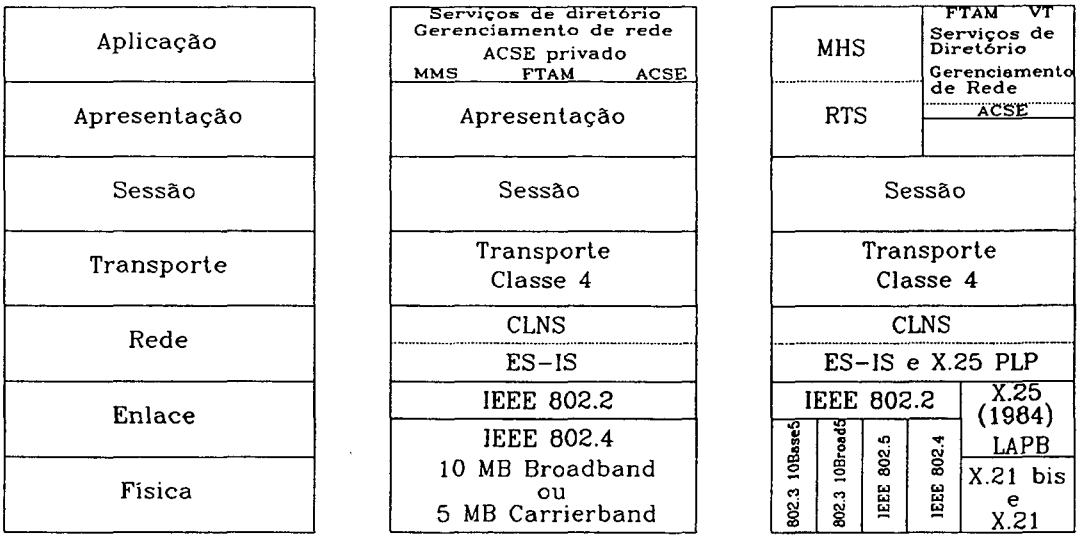


Figura 2.3 Arquiteturas RM-OSI, MAP 3.0 e TOP 3.0

Com a evolução da especificação do MAP, usuários de Controles de Processos sentiram a necessidade de tempos de resposta mais rápidos em relação a troca de mensagem no caso de aplicações de tempos críticos (como por exemplo a comunicação entre equipamentos a nível de células flexíveis de manufatura), isto apesar de sacrificar algumas das funcionalidades do MAP. Desta necessidade surgiram as arquiteturas MAP/EPA e Mini-MAP.

A arquitetura MAP/EPA (*Enhanced Performance Architecture*) provê uma "pilha dupla de protocolos" (fig. 2.4), uma pilha completa MAP e uma comprimida (EPA), sem as camadas de rede, transporte, sessão e apresentação. Com isso, a pilha EPA proporciona uma interface exposta da camada de aplicação com a camada de enlace, diminuindo o *overhead* provocado pelas camadas intermediárias. Assim, um processo de aplicação pode usar a pilha EPA para respostas mais rápidas em situações temporais críticas. É claro que o uso da pilha comprimida acarreta na perda dos serviços das camadas eliminadas.

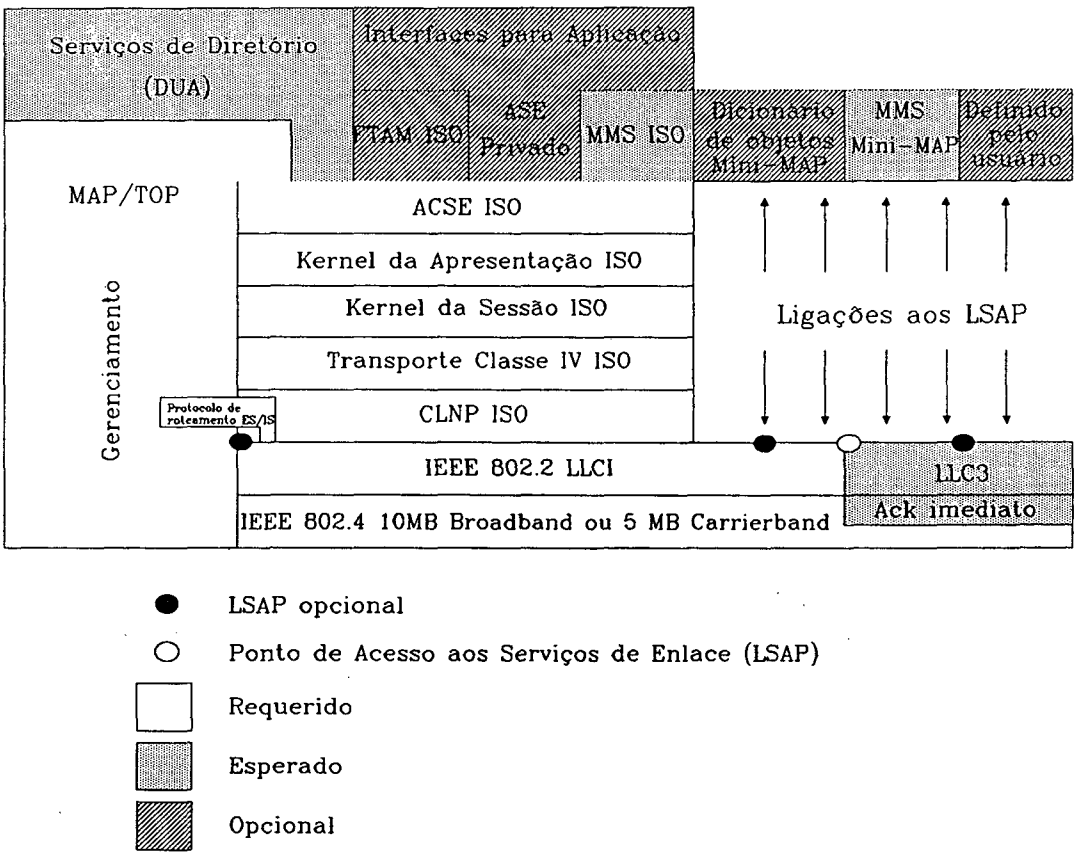


Figura 2.4 Arquitetura MAP/EPA [MAP 88]

Associado a arquitetura MAP/EPA existe uma arquitetura mais simplificada com três níveis (físico, enlace, aplicação) denominado de Mini-MAP. Esta arquitetura encontra-se associada aqueles dispositivos que conterão apenas o lado EPA da arquitetura MAP/EPA.
Os nós Mini-MAP devem ser ligados a rede MAP através de *gateways* e para isso só devem ser usados em caso de real necessidade no cumprimento de requisitos temporais estritos.

Os nós MAP/EPA apresentam a desvantagem de serem de custos mais elevados, devido a arquitetura de dupla pilha de protocolos, porém podem ser nodos que atuam também como gateways para as estruturas Mini-MAP.

2.4.2 Padrão Field Bus

Na definição ISA (*Instrument Society of America*) [ISA 87], o Barramento de Campo ou Field Bus é uma linha de comunicação serial, digital, bidirecional (de acesso compartilhado), para interligação dos dispositivos primários de automação (transmissores/sensores, atuadores/elementos de controle final e outros pequenos dispositivos inteligentes, com capacidade de processamento local), instalados na área de campo e os dispositivos de controle/automação de nível imediatamente superior (controladores), instalados na "sala de controle" [Aguiar 89].

O Field Bus adota o modelo OSI/ISO reduzido o três camadas, física, enlace e aplicação.

2.4. Conclusão

Neste capítulo foram apresentados aspectos gerais sobre o CIM. Esta discussão objetivou posicionar no contexto CIM os padrões MAP e Mini-MAP, que serão objeto do nosso estudo.

No capítulo seguinte será apresentado um estudo do padrão MMS, que é um dos elementos de serviços da camada de aplicação do MAP e Mini-MAP, serviços estes destinados à transmissão de mensagens entre equipamentos programáveis no ambiente de Manufatura e Controle de Processos. A Interface para o Programa de Aplicação MMS (MMSI) será também discutida.

CAPÍTULO 3

O PADRÃO MMSI DE INTERFACEAMENTO PARA PROGRAMAS DE APLICAÇÃO NUM AMBIENTE MMS

3.1 Introdução

A especificação de mensagens da manufatura (MMS) é um dos elementos de serviços (ASE) da Camada de Aplicação do padrão MAP e Mini-MAP. Ele foi projetado para suportar a troca de mensagem entre os dispositivos programáveis, a nível de chão de fábrica, num ambiente CIM.

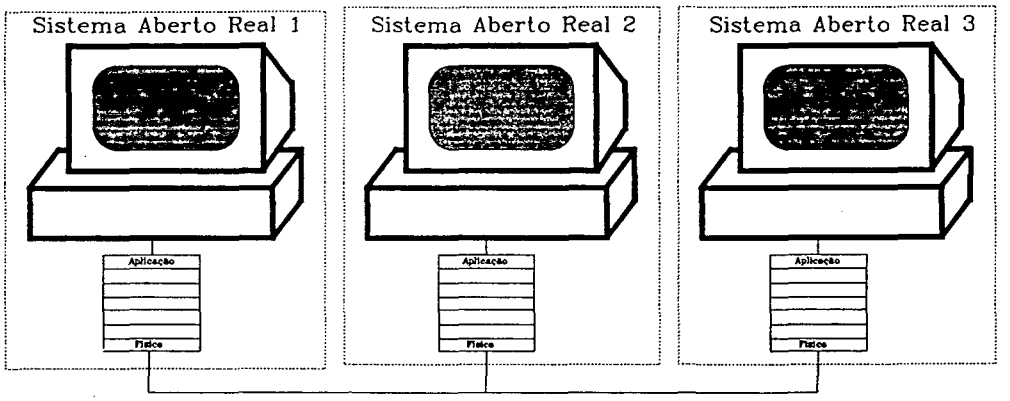
Para facilitar a utilização dos serviços MMS pelo programa usuário, foi também proposto pela norma MAP, um padrão de Interface para o Programa de Aplicação MMS (MMSI).

Neste capítulo serão descritos e comentados o padrão MMS e a Interface para o Programa de Aplicação MMS (MMSI). Após ter apresentado alguns dos conceitos adotados no RM-OSI, MMS e MMSI que são necessários a este estudo, discutir-se-á a utilização dos padrões MMS e MMSI numa arquitetura Mini-MAP.

3.2 Conceitos Básicos do Modelo de Referência OSI e da sua Camada de Aplicação

O Modelo de Referência para Interconexão de Sistemas Abertos (RM-OSI) é um padrão internacional publicado pela Organização Internacional de Padronização (ISO) que fornece um suporte arquitetural com sete camadas (conforme definido no capítulo 2) para o desenvolvimento de padrões de serviços e protocolos de comunicação. Os padrões de interconexão de Sistemas Abertos (*Open System Interconnection* - OSI) trazem uma solução para o problema de comunicação entre sistemas computadorizados de diversas tecnologias e fabricantes.

Um sistema informatizado (software e hardware) que se comunica com outro sistema, adotando o padrão OSI, é dito um Sistema Aberto Real (*Real Open System* - ROS) e pode ser representado conforme visto na figura 3.1.



3.1 Definição de Sistemas Abertos Reais

As interações entre ROS se dão através de interações entre Processos de Aplicação (**AP**) que são elementos lógicos que podem executar uma ou mais tarefas de processamento da informação.

Uma atividade (execução) específica de um AP, integrante de uma aplicação distribuida particular, é modelada por uma invocação de AP (AP-I). Desta forma, uma aplicação distribuida consiste na interação entre AP-I. Para esta interação, são utilizadas funções de comunicação, modelada por Entidades de Aplicação (AE), que são as partes de um AP envolvidas numa comunicação dentro do ambiente OSI. As AE são definidas por um conjunto de ASE, tais como MMS, ACSE e FTAM. Cada reagrupamento específico de funções de comunicação é chamado de Tipo de Entidade de Aplicação (AE-Tipo).

A execução das funções de comunicações de um AP numa comunicação específica, são realizadas por uma Invocação de AE (AE-I). Assim, a comunicação entre uma AE e seu par remoto, se dá através de suas respectivas AE-I. Para existir uma seção de comunicação entre duas AE-I, estas devem negociar inicialmente o conjunto de parâmetros que define o ambiente de comunicação, chamado contexto de aplicação. Esta operação é conhecida na norma como estabelecimento de uma Associação de Aplicação (AA).

Cada AE representa exatamente o seu AP de forma que este seja comunicável com o AP remoto. Afim de tornar-se endereçável no ambiente OSI, uma AE é ligada a um ou mais Pontos de Acesso a Serviços da camada de Apresentação (PSAP), sendo que o grupo de PSAP de uma mesma AE é associado a um mesmo endereço de apresentação. Este endereço pode ser obtido do diretório através de um dos Títulos da AE.

3.3. Descrição do MMS

3.3.1 Técnica de Modelagem Empregada

O padrão MMS utiliza uma técnica de Modelagem baseada em objetos abstratos. A ação de cada serviço e seus efeitos são entendidos a partir da troca de estados dos atributos do objeto associado ao serviço.

3.3.2 O Princípio da Comunicação entre Programas de Aplicação num Ambiente MMS

Na interação entre os AP, o padrão MMS segue o modelo de interação Cliente/Servidor. Nesta interação, os serviços MMS definem o comportamento externamente visível de um Servidor MMS. Este comportamento é modelado por um Dispositivo Virtual de Manufatura (VMD), que representa por objetos MMS as potencialidades físicas e funcionais do dispositivo, disponíveis através dos serviços MMS. O cliente MMS não tem modelo e é só limitado pela sequência válida das primitivas. A figura 3.2 é uma representação de alguns dos conceitos apresentados neste ítem.

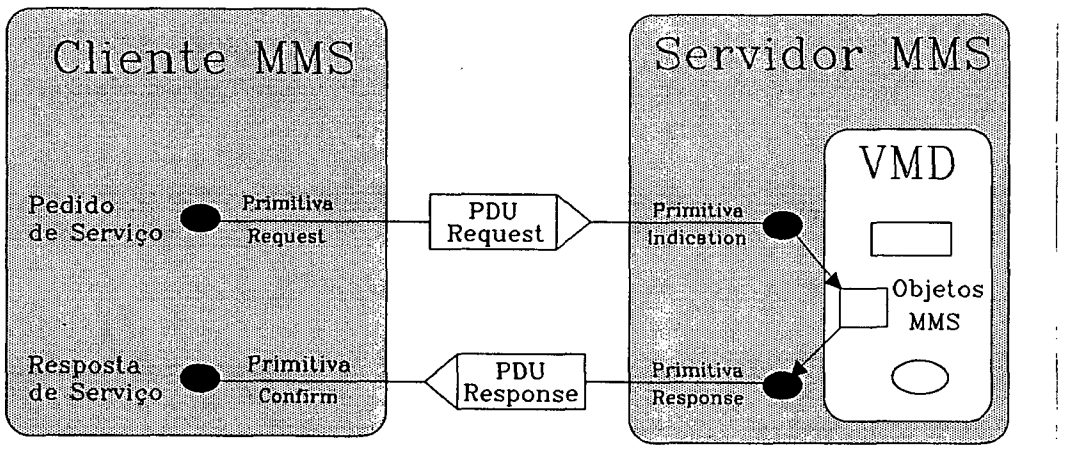


Figura 3.2 Interações entre AP no MMS

Para possibilitar a comunicação no ambiente OSI, o VMD (AP Servidor) e o AP Cliente devem ter uma ou mais AE. O padrão MMS define que cada AE-I suporta uma única AA, diferente da visão generalizada do OSI, que permite que uma AE-I tenha mais que uma AA.

3.3.3 Os Serviços MMS

Os serviços MMS especificados em [ISO 88] são agrupados em 10 classes, vistos na figura 3.3. Esta especificação provê um conjunto de serviços que são inteiramente comuns para Controles Numéricos (NC), Controladores Programáveis (CP), robôs e outros sistemas do ambiente de manufatura e que corresponde a uma representação genérica de um usuário MMS. É de responsabilidade dos *Companions Standards* particularizar o padrão MMS para cada tipo de dispositivo, por exemplo, O *Companions Standards* para o CP é definido em [IEC 89], para o NC em [ISO 89a], para o robô em [ISO 89b] e para os sistemas de controle em [ISA 90].

Unidades Funcionais	Primitivas MMS		Comentarios
Gerenciamento de Contexto	Initiate	(*)	Para instanciar, concluir, abortar, cancelar, rejeitar comunicacoes com outro usuario MMS.
	Conclude	(*)	
	Abort	req/ind	
	Cancel	(*)	
	Reject	ind	
Suporte de VMD	Status	(*)	Supre servicos de VMD procurando dar informacoes sobre objetos dos VMDs (lista de nomes, renomeacoes, etc)
	Unsolicited_status	req/ind	
	Get_name_list	(*)	
	Identify	(*)	
	Rename	(*)	
Gerenciamento de Dominio	Initiate_down_load_sequence	(*)	Os servicos sao destinados a permitir que um cliente MMS possa manipular dominios definidos pelo servidor de forma dinamica: as sequencias DOWLOAD transferem o conjunto do DOMAIN do cliente para o servidor e os UPLOAD na direcao oposta.
	Down_load_segment	(*)	
	Terminate_down_load_sequence	(*)	
	Initiate_up_load_sequence	(*)	
	Upload_segment	(*)	
	Terminate_upload_sequence	(*)	
	Request_domain_down_load	(*)	
	Request_domain_upload	(*)	
	Load_domain_content	(*)	
	Store_domain_content	(*)	
	Delete_domain	(*)	
	Get_domain_attribute	(*)	
	Domain_file	(*)	
Gerenciamento de Invocacao de Programa	Create_program_invocation	(*)	Os servicos permitem ao cliente MMS dirigir a evolucao de programas
	Delete_program_invocation	(*)	
	Start	(*)	
	Stop	(*)	
	Resume	(*)	
	Reset	(*)	
	Kill	(*)	
	Get_program_invocation_attributes	(*)	

Obs.: (*) Servicos Confirmados (req/ind/rsp/cnf).

Figura 3.3 Os Serviços MMS [Mendes 89]

Unidades Funcionais	Primitivas MMS		Comentarios
Acesso da Variavel	Read	(*)	Sao providenciadas facilidades que permitem o usuario MMS cliente acessar variaveis tipadas do VMD, em particular incluindo a definicao de cinco classes adicionais de objetos, (variavies nao nomeadas, nomeadas, de acesso distribuido, listas de variaveis nomeadas, tipos nomeados). Procede-se ao mapeamento entre os objetos virtuais (variaveis MMS) e os objetos reais (variaveis reais). "Real"=tem existencia concreta (nao ponto flutuante).
	Write	(*)	
	Information_report	(*)	
	Get_variable_access_attributes	(*)	
	Delete_named_variable	(*)	
	Define_scattered_access_attributes	(*)	
	Delete_variable_access	(*)	
	Define_name_variable_list	(*)	
	Get_named_variable_list_attributes	(*)	
	Delete_named_variable_list	(*)	
	Define_named_type	(*)	
	Get_named_type_attributes	(*)	
	Delete_named_type	(*)	
Gerenciamento de Semaforo	Take_control	(*)	Os objetos semaforos sao manipulados de tal forma a se permitir a sincronizacao, controle e coordenacao de recursos com partilhados entre diversos usuarios MMS.
	Relinquish_control	(*)	
	Define_semaphore	(*)	
	Delete_semaphore	(*)	
	Report_semaphore_status	(*)	
	Report_pool_semaphore_status	(*)	
	Report_semaphore_entry_status	(*)	
Comunicacao de Operadores	Input	(*)	Mecanismos para comunicacao com estacao do operador(display e entrada de dados). Facilidades para que o cliente defina e gerencie os eventos de um VMD e obtenha informacao da sua ocorrencia. Existem tres classes de objetos: Condicoes de Eventos, Acoes de Eventos e Registros de Eventos.
	Output	(*)	
Gerenciamento de Eventos	Define_event_condition	(*)	
	Delete_event_condition	(*)	
	Get_event_condition_attribute	(*)	
	Report_event_condition_status	(*)	
	Alter_event_condition_monitorining	(*)	
	Triger_event	(*)	
	Define_event_action	(*)	
	Delete_event_actron	(*)	
	Get_event_action_attributes	(*)	
	Report_event_action_status	(*)	
	Define_event_enrollment	(*)	
	Delete_event_enrollment	(*)	
	Get_event_enrollment	(*)	
	Report_event_enrollment	(*)	
	Alter_event_enrollment	(*)	
	Event_notification	(*)	
	Acknowledge_event_notification	(*)	
	Get_alarm_summary	(*)	
	Get_alarm_enrollment_summary	(*)	
	Attach_to_event_condition_modifier	(*)	
Gerenciamento de Jornal	Read_journal	(*)	Com o jornal permite-se a memorizacao e busca da informacao cronologica referente a eventos e outras variaveis de interesse.
	Write_journal	(*)	
	Initialize_journal	(*)	
	Report_journal_status	(*)	
Gerenciamento de Arquivo	File_open	(*)	Permite a manipulacao de arquivos (contendo programas e dados) localizados em "Filestores" de dispositivos de controle ou em servidores especializados.
	File_read	(*)	
	File_close	(*)	
	File_rename	(*)	
	File_delete	(*)	
	File_direction	(*)	

Figura 3.3 Os Serviços MMS (continuação) [Mendes 89]

A seguir serão descritos detalhadamente dois pontos importantes para entender o comportamento MMS: o estabelecimento do Ambiente MMS, requerido para o uso dos serviços MMS, e os tipos de serviços.

A) O Estabelecimento do Ambiente MMS

Antes que dois usuários MMS possam se comunicar através dos serviços MMS, é necessário definir o ambiente de comunicação entre eles, chamado de Ambiente MMS o que será obtido através do estabelecimento de uma Associação de Aplicação. O diagrama de estados da entrada/saída do ambiente MMS e os serviços utilizados (*Initiate*, *Conclude*, *Abort* e *Reject*) podem ser vistos na figura 3.4.

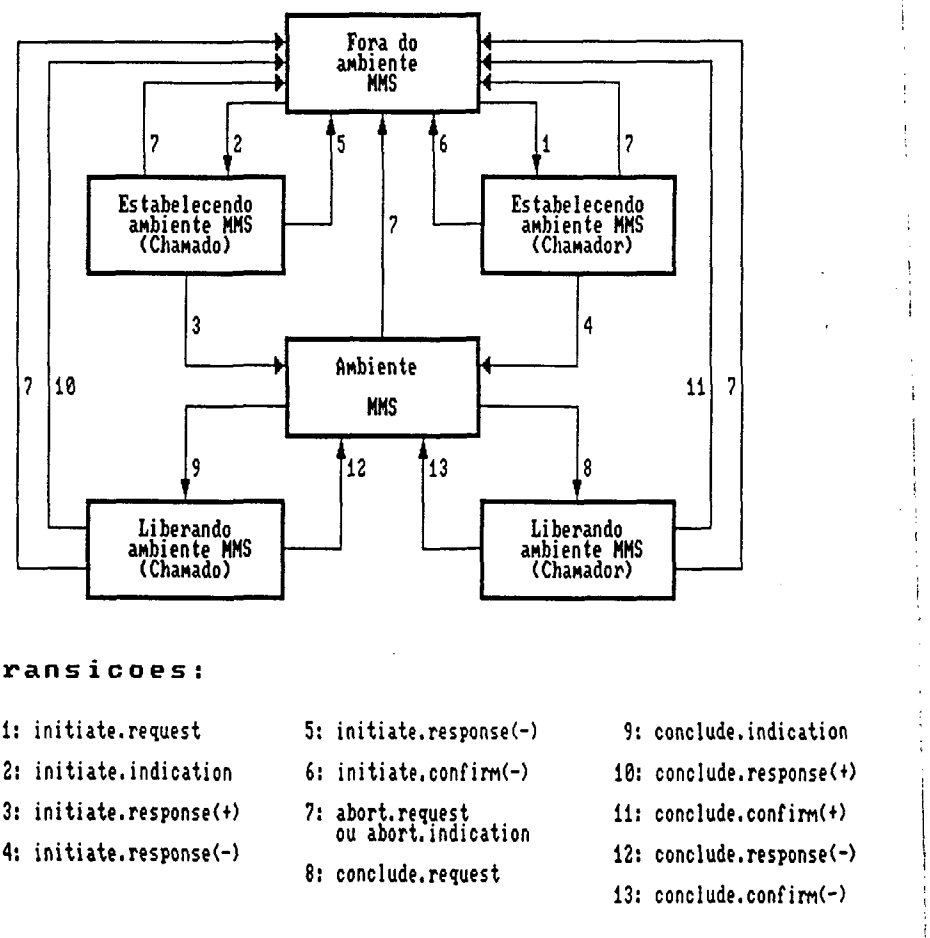


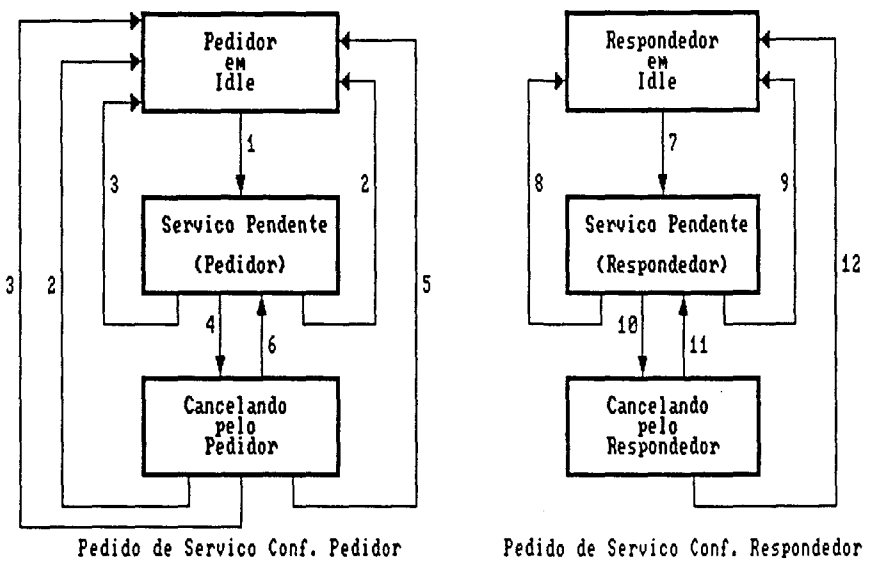
Figura 3.4 Diagrama de estados de entrada/saída do Ambiente MMS.

Durante a permanência no estado Ambiente MMS, um usuário MMS pode emitir qualquer primitiva de serviço MMS, menos *initiate.req*.

B) Os Tipos de Serviços

Os serviços MMS podem ser de dois tipos: confirmados ou não confirmados.

Os serviços não confirmados são aqueles que não implicam na espera de resposta do usuário remoto. Para estes, nenhuma informação associada ao pedido é armazenada, ou ainda, não é criada uma máquina de estados associada ao pedido (máquina de transação).



Pedido de Servico Conf. Pedidor

Pedido de Servico Conf. Respondedor

Transicoes:

1: x.request / Confirmed_RequestPDU(x)

2: Confirmed_ResponsePDU(x) / x.confirm(+)

3: Confirmed_ErrorPDU(x) / x.confirm(-)

4: cancel.request / Cancel_RequestPDU

5: Cancel_ResponsePDU e Confirmed_ErrorPDU(x) / Cancel.confirm(+) e x.confirm(-)

6: Cancel_ErrorPDU / cancel.confirm(-)

7: Confirmed_RequestPDU(x) / x.indication

8: x.response(+) / Confirmed_ResponsePDU(x)

9: x.response(-) / Confirmed_ErrorPDU(x)

10: Cancel_RequestPDU / Cancel.indication

11: Cancel.response(-) / Cancel_ErrorPDU

12: Cancel.response(+) e x.response(-) / Cancel_ResponsePDU e Confirmed_ErrorPDU(x)

Figura 3.5 Máquina de estados dos serviços confirmados

Os serviços confirmados caracterizam-se pela espera de resposta, para tanto, é criado uma máquina de estados para cada invocação de um destes serviços. A máquina de estados dos serviços confirmados é mostrada na figura 3.5. A fim de associar a máquina de estados ao serviço confirmado correspondente, o pedido de serviço confirmado deve prover um identificador de invocação (*Invoke ID*), que deve ser único numa associação.

3.4 Interface para Programa de Aplicação num Ambiente MMS (MMSI)

A norma MAP [MAP 88] especifica uma Interface para o Programa de Aplicação (API) definida para os protocolos OSI/ISO da camada de aplicação. A MMSI é uma interface padrão para o fornecimento dos serviços MMS no qual é definido uma biblioteca de funções padronizadas, conforme figura 3.6. As vantagens obtidas com a utilização da interface padrão são ([MAP 88]):

- aumento da portabilidade dos programas usuário;
- diminuição dos custos com o treinamento de programadores;
- facilidade de migração para futuras versões do MAP;
- possibilidade de desenvolvimento de software de aplicação por terceiros.

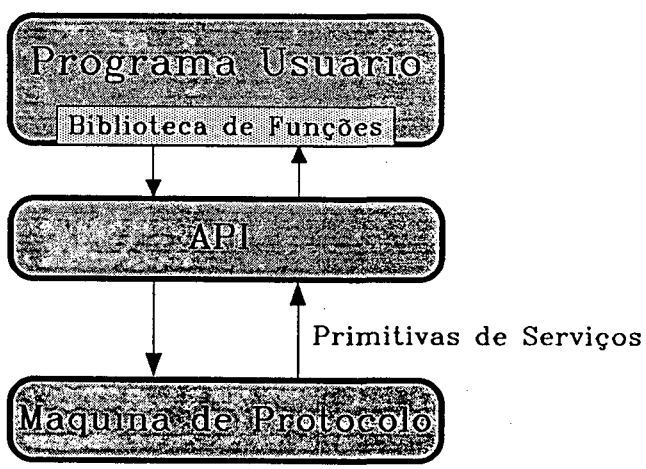


Figura 3.6 Interface para o Programa de Aplicação

3.4.1 Modelo da MMSI

Na norma MAP é especificado o modelo da Interface para o Programa de Aplicação na forma apresentada na figura 3.7. Nesta figura são indicados apenas os principais fluxos de informação (linhas cheias) e de controle (linhas tracejadas). Este modelo é composto de blocos que representam funcionalidades que atuam na execução dos pedidos de serviços por parte do usuário.

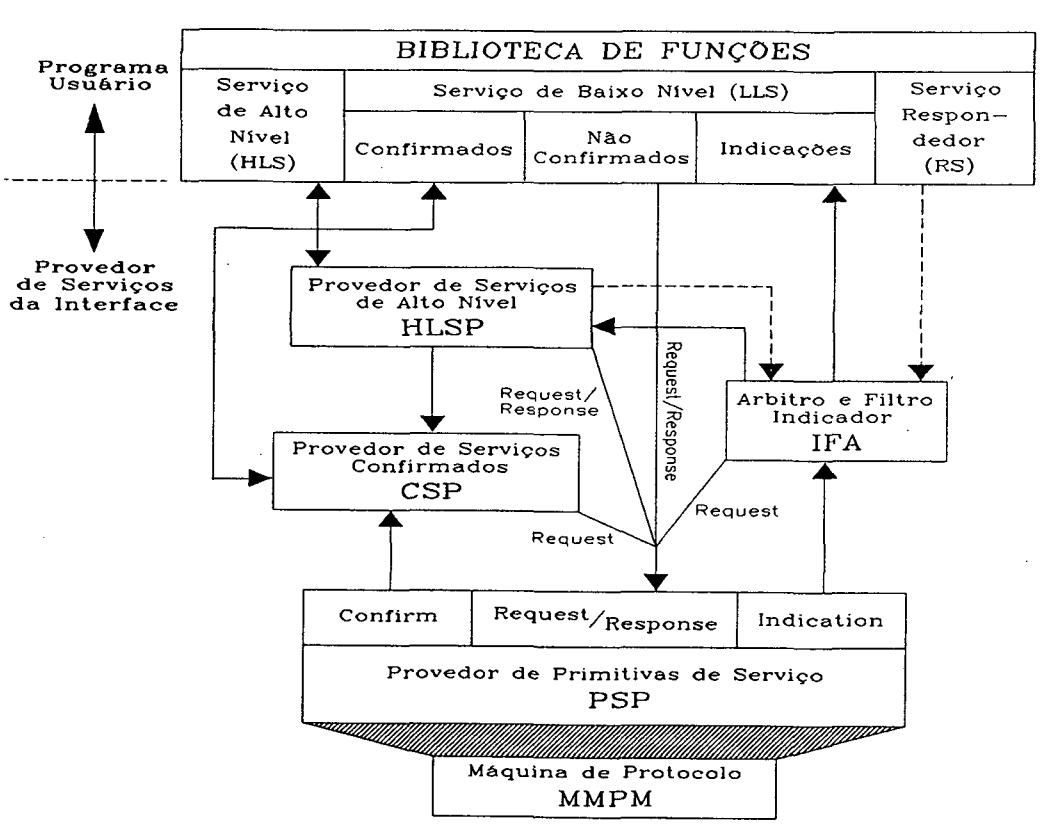


Figura 3.7 Modelo da Interface para o Programa de Aplicação [MAP 88]

Este modelo é dividido em duas partes, que são descritas a seguir: uma formada pelo conjunto de funções que são parte do Programa Usuário e uma parte relacionada com o Provedor de Serviços.

3.4.1.1 Parte do Programa Usuário

Os blocos desta parte do modelo da MMSI são funções de uma biblioteca. Através de uma chamada à esta biblioteca, o Programa Usuário acessa o Provedor de Serviços para a utilização dos serviços de comunicação MMS. Neste relacionamento, a interface MMSI sempre atua como um elemento passivo, que interage com o Programa Usuário apenas quando solicitada por este.

A especificação da interface proposta pela norma MAP, direciona sua utilização para ambientes com concorrência. Neste ambiente, a interação entre o Programa Usuário e o Provedor de Serviços é feita via serviços de mensagens e de notificação, fornecida por um mecanismo padronizado de gerência de eventos . Quando uma função da biblioteca é chamada, ocorre o envio de pedido de serviço ao provedor associado. Nesta chamada, o Programa Usuário define se o serviço será executado síncrona ou assincronamente; caso a chamada seja síncrona, o Programa usuário aguardará bloqueado o encerramento do serviço; no caso assíncrono, o Programa Usuário faz uso da função WAIT, a fim de aguardar bloqueado a resposta do pedido durante o período definido nesta chamada. Ao completar o serviço, o Provedor de Serviços armazena a resposta do serviço na área de dados de saída (previamente definida pelo Programa Usuário) e notifica este encerramento.

A comunicação entre o Programa Usuário e o Provedor de serviço dependerá da comunicação inter-processos provida pelo sistema operacional e não é visível ao Programa Usuário, pois ficará oculta na chamada de função da biblioteca da API. Enquanto que a resposta do serviço tem seu interfaceamento padronizado pela espera de um evento (função WAIT).

Há vários tipos de função na biblioteca da API, são eles:

Funções de Alto Nível (HLS) - fornece um conjunto de serviços compostos, possibilitando a redução da complexidade do Programa Usuário. Este bloco faz uso de vários serviços de baixo nível para completar cada um de seus serviços. Para a interface MMSI, as funções de alto nível são aquelas associadas ao estabelecimento de associação e alguns serviços de transferência de arquivos.

Funções Baixo Nível (LLS) - através destas funções, o Programa Usuário pode requisitar serviços que são mapeados diretamente em um único serviço MMS.

Serviços Respondedores (RS) - estas funções geram mecanismos para os Provedores de Serviço responderem automaticamente a pedidos, liberando o Programa Usuário desta operação.

Funções Auxiliares - a biblioteca de funções da MMSI apresenta algumas funções auxiliares que facilitam a passagem de parâmetros recursivos, tal como descrição de tipo de variável, descrição de acesso alternado, vindo do usuário para o Provedor de Serviços da interface e vice-versa. Estas funções não constituem uma comunicação com um dispositivo remoto.

Funções de Suporte - são também definidas funções de suporte ao Programa Usuário, que são: a função WAIT apresentada anteriormente; a função GET PRINTABLE ERROR, responsável pela translação de código de erro, retornado nas chamadas das funções da biblioteca, para *string* imprimível; e as funções DYNAMIC INITIALIZE DCB e DYNAMIC FREE DCB responsáveis pela de alocação e dealocação dos Blocos de Controle de Dados (*Data Control Block* - DCB). Um DCB corresponde a uma área de dados (*buffer*) que contém parâmetros opcionais de entrada e saída das funções da biblioteca da MMSI.

3.4.1.2 Parte do Provedor de Serviços

Segundo a norma MAP, a forma de interação entre os Provedores de Serviços é similar àquela entre o Programa Usuário e os Provedores de Serviços. Ao contrário da anterior, esta forma de interação não é normalizada, sendo objeto de decisões do projetista. O padrão da MMSI normaliza apenas o interfaceamento do Programa Usuário com o Provedor de Serviços da interface.

Os seguintes blocos compõem o Provedor de Serviços:

Provedor de Serviços de Alto Nível (HLSP)

O bloco HLSP fornece serviços de alto nível, fazendo uso dos blocos CSP (nos serviços confirmados), PSP (nos serviços não confirmados) ou IFA (para filtragens das indicações cujos pedidos devem ser levados ao HLSP). O bloco HLSP é específico da Entidade de Aplicação (AE).

Provedor de Serviços Confirmados (CSP)

Este bloco é usado em serviços confirmados, associando primitivas *confirm* às respectivas primitivas *request* (com o mesmo identificador de invocação *Invoke_id*). Também é responsável pela geração do *Invoke_id*. O bloco CSP é específico de associação e tem sua ação baseada no bloco PSP.

Árbitro e Filtro de Indicações (IFA)

O bloco funcional IFA filtra as primitivas de indicação recebidas em uma associação, envia-as ao Programa Usuário ou ao bloco HLSP, ou ainda responde a indicação de serviço automaticamente. Estas funções de filtragens são ativadas pelo bloco HLSP ou pelo Programa Usuário através dos Serviços Respondedores (RS). O Programa Usuário receberá uma primitiva de *indication* apenas quando esta não for filtrada pelo IFA, ou nem o destino for o HLSP. O bloco IFA, como o CSP, é específico de associação.

Provedor de Primitivas de Serviços (PSP)

O bloco PSP executa as funções de intermediações com a MMPM, enviando primitivas *request* ou *response* para esta máquina de protocolo e recebendo primitivas de *confirm* ou *indication* da mesma. O PSP ainda informa aos seus usuários (blocos de nível superior) quando uma indicação não solicitada recebida pode afetar suas operações (chegada de uma indicação de *abort*, por exemplo). O bloco PSP também é específico de associação.

Os requisitantes dos serviços do bloco PSP (IFA e CSP), devem fazer uma solicitação ao PSP para a obtenção de primitivas *indication* (por parte do IFA) ou *confirm* (por parte do CSP). Estas solicitações serão satisfeitas de forma que o primeiro a chegar, seja o primeiro a ser servido. Observa-se aqui o caráter passivo de um Provedor de Serviços.

Máquina de Protocolo da Mensagem da Manufatura (MMPM)

A MMPM define a máquina de estados do protocolo MMS. Suas principais funções são a de tratamento das primitivas recebidas do PSP e mapeando-as em PDU e envio a camada inferior via primitivas do ACSE (Elemento de Serviço de Controle

de Associação) ou da camada de apresentação, e vice-versa. O bloco MMPM é específico de associação.

Estes blocos modelam funcionalidades que, numa implementação são providas por uma ou mais tarefas, sendo que a definição destas tarefas é um critério de implementação.

3.4.2 Etapas de Utilização da Interface MMSI

A seguir serão definidos os principais procedimentos que devem ser efetuados pelo Programa Usuário, para que este possa fazer uso dos serviços MMS:

A) Definição de Objetos MMS na MMSI

Todos os dispositivos servidores que necessitam definir objetos MMS no Provedor de Serviços da interface, necessitam inicialmente ativar um VMD, feito através da função VMD ACTIVATION. As funções VMD DEACTIVATION e VMD RESET executam a desativação (após todos os objetos MMS terem sido desativados) ou eliminação drástica do VMD.

Após a ativação de um VMD, o Programa Usuário poderá definir objetos MMS neste. A MMSI especifica funções para a definição de objetos MMS associados ao acesso à variável e ao gerenciamento de semáforos.

B) Ativação de AE

Para que o Programa Usuário faça uso de uma AE-I, é necessário que ele declare a existência desta AE-I ao Provedor de Serviços da interface. Durante esta declaração da, algumas funções de gerenciamento local são executadas. A declaração da AE-I é executada na chamada da função AE ACTIVATION, que apresenta entre outros, o parâmetro de entrada *my_dir_name* que fornece o título registrado no diretório da AE da invocação a ser declarada e o parâmetro de saída *ae_label* que identifica a AE-I declarada, permitindo subsequentes pedidos sobre esta. Cada AP deverá declarar uma AE-I para cada associação necessária sobre esta AE, pois uma AE-I suporta apenas uma única associação.

As funções de gerenciamento local e a forma de identificação da AE-I declarada devem ser definidas quando da implementação desta interface.

Existe também a função AE DEACTIVATION que pode ser requisitado pelo Programa Usuário quando não há mais necessidade de se comunicar sobre uma determinada AE-I.

Outra função associada é a AE RESET, que permite ao usuário abortar a conexão aberta ou pendente sobre uma AE-I e emitir um STOP LISTEN (visto a seguir).

C) Estabelecimento de Associação

Para o estabelecimento de uma associação, a MMSI fornece ao Programa Usuário as funções CONNECT, para o pedido de associação, LISTEN e ANSWER para recepção e resposta no pedido de associação. Estes serviços são visto a seguir:

A função CONNECT é mapeada no serviço MMS *Initiate*, sendo os parâmetros passados na lista de argumento desta função. O serviço MMS *Initiate*, por sua vez é mapeado no serviço de controle de acesso ao meio *A_Associate*, no caso do MAP completo.

Para que o usuário remoto possa informar o desejo de aceitar uma indicação de associação sobre uma AE-I, ele deve fazer uso da função LISTEN. Antes do usuário remoto chamar esta função, todas as indicações para abertura de associação sobre esta AE-I serão automaticamente rejeitadas. O Programa Usuário deverá chamar a função LISTEN para cada AE-I que queira aceitar um pedido de associação. É possível cancelar uma função LISTEN assíncrona, através da função STOP LISTEN. Estas duas funções não geram mensagens sobre a rede.

A chamada de uma função de alto nível ANSWER, pelo Programa Usuário respondedor, causa a emissão de uma primitiva de resposta do pedido de associação. Esta chamada é mapeada na primitiva *Initiate.rsp*.

Se as funções CONNECT e LISTEN forem executadas com sucesso, será retornado um identificador de conexão (*connection_id*) aos participantes da associação, permitindo posterior identificação desta. O valor de *connection_id* será única para cada AE-I no caso do MMS (uma única associação para cada instância de AE). Este parâmetro também deverá ser definido durante uma implementação da interface.

D) Encerramento de uma Associação de Aplicação

O encerramento de uma conexão pode ser feito de modo negociado através das funções Conclude (*Conclude.req*) e Conclude Response (*Conclude.res*), ou de forma abrupta através da função Abort (*Abort.req*) ou Indication Receive com *indication_name* = MM_ABORT_IND (*abort_ind*).

E) Operações com outros serviços MMS

Para solicitar um serviço MMS, o Programa Usuário deverá chamar a função da biblioteca correspondente ao serviço MMS requerido. Os parâmetros da primitiva *request* são definidos através dos parâmetros de entrada, e o resultado, no caso dos serviços confirmados, é passado ao Programa Usuário através do DCB de entrada/saída.

Para recepção de uma indicação de serviço, o Programa Usuário deverá chamar a função INDICATION RECEIVE. Observa-se aqui novamente o caráter inerentemente passivo da MMSI, onde o usuário local é responsável por fazer o pedido para o recebimento de uma solicitação de serviço. Através do parâmetro *indication_name* o Programa Usuário reconhecerá qual é o serviço MMS pedido e no DCB de entrada/saída estarão as informações do pedido.

A resposta de um serviço MMS é enviada através da chamada da função MMSI de resposta, correspondente ao serviço solicitado, onde os parâmetros da primitiva de resposta são passados na lista de argumentos da chamada.

3.5 Aspectos da Utilização do Padrão MMS num Arquitetura Mini-MAP

Conforme introduzido no capítulo anterior, a arquitetura Mini-MAP, vista na figura 3.8, surgiu da necessidade de melhorar o desempenho da comunicação para aplicações com resposta em tempo críticos (células flexíveis, por exemplo). Nesta arquitetura, há um interfaceamento direto entre a camada de aplicação e a camada de enlace, assim os serviços MMS são diretamente mapeados nos serviços de enlace, oferecidos nos Pontos de Acesso a Serviços de Enlace (LSAP). Portanto na arquitetura Mini-MAP, uma AE está diretamente ligada a um LSAP. A seguir serão discutidos alguns aspectos da utilização dos padrões MMS e MMSI neste ambiente.

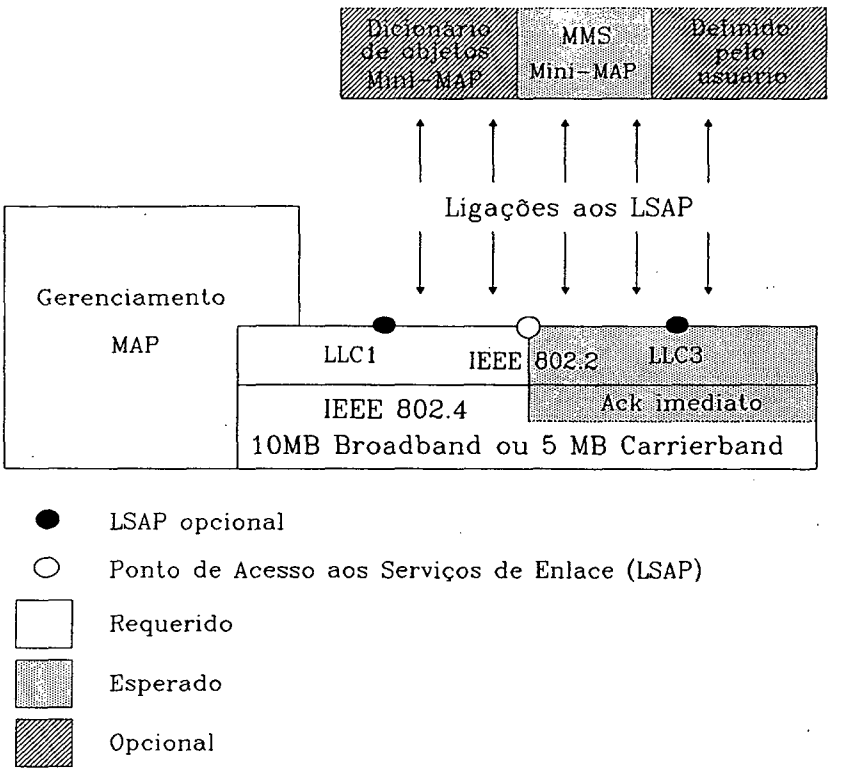


Figura 3.8 Arquitetura Mini-MAP [MAP 88]

3.5.1 O MMS Mini-MAP

Na arquitetura Mini-MAP, a MMPM (MMPM Mini-MAP) fornece os serviços MMS mapeando-os diretamente nos serviços de enlace. Os serviços fornecidos pela camada de enlace (LLC) do Mini-MAP são de único *frame*, sem conexão e com reconhecimento imediato (tipo LLC 3). Caso existam endereços *multicast*, o serviço de enlace do tipo 1 (LLC 1) são utilizados.

O uso completo da capacidade dos serviços de enlace requer a presença de um elemento de serviço, chamado ASE-*Obtain Reply*, adicionado ao ASE-MMS. O ASE-*Obtain Reply* executa pedidos do usuário Mini-MAP para um conjunto de estações que são membros da sub-rede, mas que não tem direitos a ficha no *token-passing*.

As figuras 3.9, 3.10 e 3.11 mostram o relacionamento entre o ASE-MMS Mini-MAP, ASE-*Obtain Reply* e a camada de Enlace, mostrando o conjunto de primitivas passadas através de cada interface, nos casos seguintes:

- a estações *token-passing* que efetuam comunicações apenas com outras estações *token-passing* (figura 3.9);
- estações token-passing que podem comunicar com estações que não tem direitos a ficha (figura 3.10);
- estações que não tem direitos a ficha (figura 3.11).

A figura 3.12 mostra a máquina de estados da MMPM Mini-MAP em uma estação que pode se comunicar com estações sem passagem de ficha. As transições de estados são sempre dirigidas pelos eventos das interfaces, que são a chegada de primitivas de serviço vindos do usuário MMS Mini-MAP, da camada de enlace, de um gerenciamento de estações, ou vindo do elemento de serviços *Obtain-Reply* ([MAP, 88]). As principais funções da MMPM Mini-MAP são ([MAP 88]):

- coordenar uma transação *request-response* única sobre condições normais;
- construir e desmembrar PDU's;
- informar *status* da camada de enlace para o usuário;
- abortar a transição devido a certos erros de enlace;
- estabelecer e terminar uma associação explicita sobre condições anormais;
- abortar uma associação explícita devido a certos erros de enlace; e
- processar o cancelamento de uma transação.

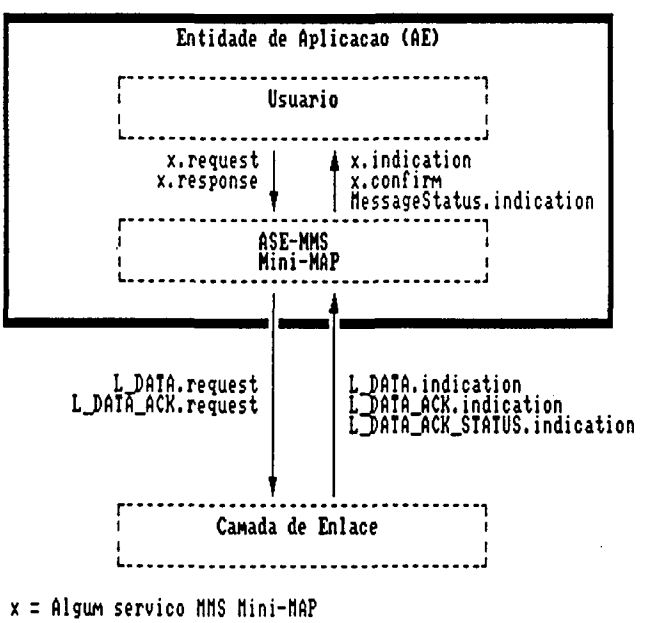


Figura 3.9 Estação *Token passing* com comunicação apenas com outras estações *Token passing* [MAP 88].

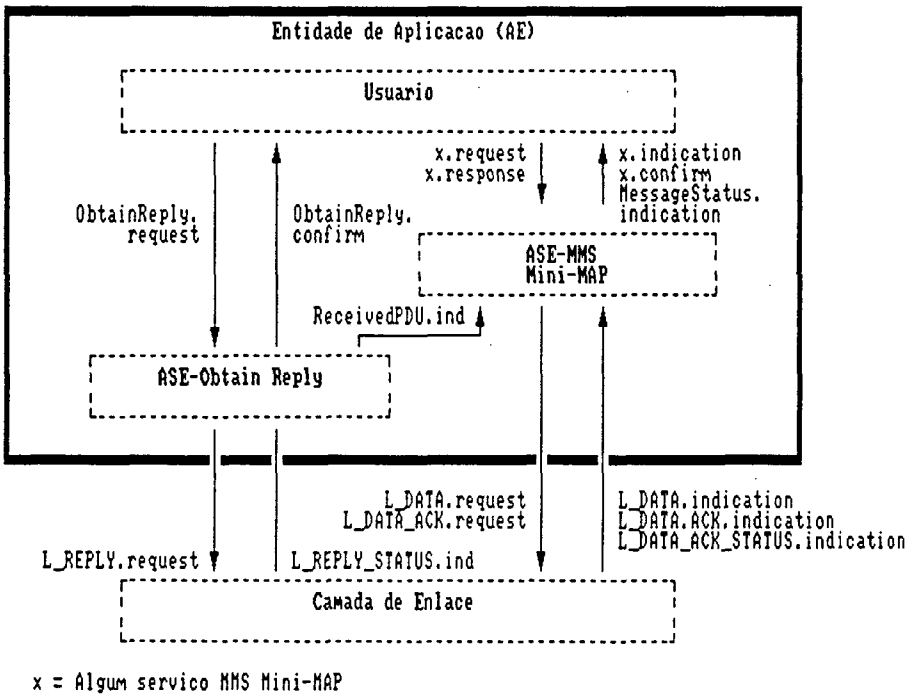


Figura 3.10 Estação *Token passing* que pode se comunicar com estações *No Token passing* [MAP 88].

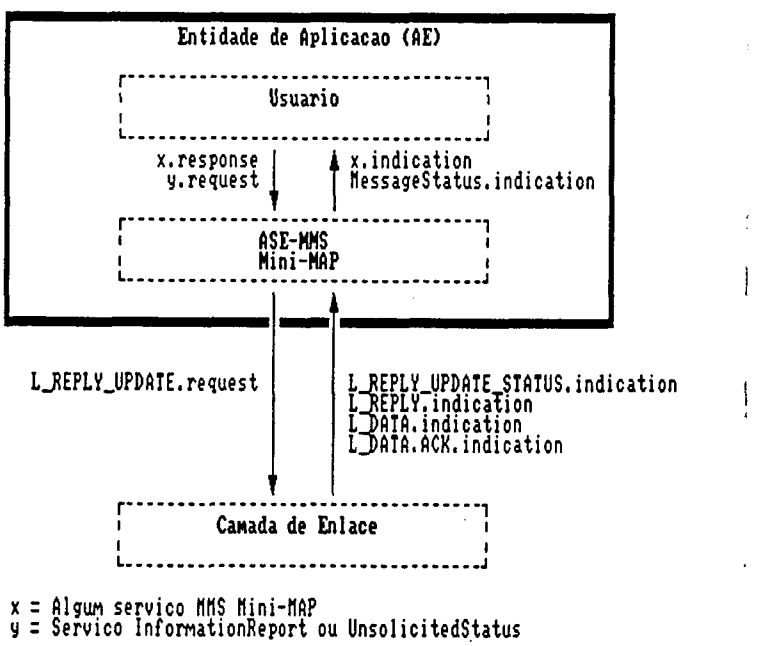
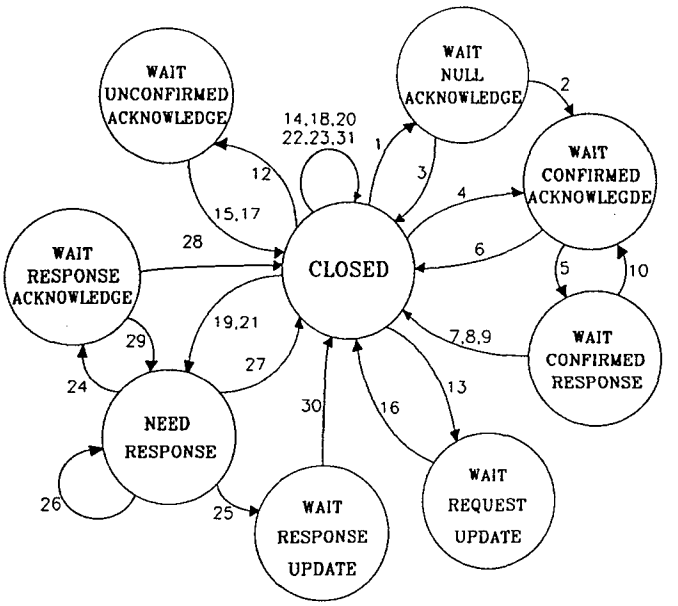


Figura 3.11 Estação *No Token passing* [MAP 88].



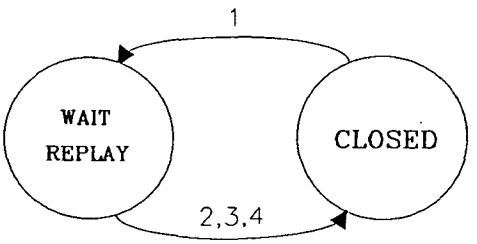
TRANSIÇÕES:

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

Figura 3.12 Máquina de Estados da MMPM Mini-MAP

Os estados representados na figura 3.12 são os estados alcançáveis por uma máquina de estados de transação. A criação de uma máquina de estados de transação é representada pela execução de uma transição de estados a partir do CLOSED para qualquer estado (incluindo o CLOSED). A máquina de estados de transação deixa de existir quando retorna ao estado CLOSED. Num determinado instante podem existir várias máquinas de transação ([MAP, 88]). Sendo assim, poderão existir várias instâncias da máquina de estados representada na figura 3.12 na MMPM.

A figura 3.13 mostra a máquina de estados do elemento de serviço *Obtain-Reply.*



TRANSIÇÕES:

1. request-a-reply
2. see-reply
3. see-response-pdu
4. cant-get-data

Figura 3.13 Máquina de Estados para o ASE-*Obtain Reply*

3.5.2 Considerações sobre a utilização do MMS na arquitetura Mini-MAP

A seguir serão levantados alguns aspectos acerca da utilização do MMS numa Arquitetura Mini-MAP:

Associação de Aplicação

Devido a inexistência das camadas de apresentação e de sessão, o protocolo ACSE não está presente no Mini-MAP, com isso, os serviços de estabelecimento de uma AA não podem ser implementados sobre esta arquitetura. Contudo a norma MAP prevê a possibilidade de continuar oferecendo uma Associação "Explícita", permitindo que a máquina de protocolo opere num ambiente similar aquele criado pelo ACSE. Desta forma, as funções de criação e manutenção da associação são desempenhadas explicitamente pela máquina de protocolo MMS Mini-MAP.

As mensagens que transitam no ambiente industrial são classificadas em oito níveis de prioridade. Este níveis são também atribuidos às associações utilizadas na transferência destas mensagens. Sendo assim, o nível de prioridade de uma associação serve como identificador entre as associações que compartilham o mesmo par de endereços LLC (LSAP) e MAC. O estabelecimento da associação explícita pode ser visto na figura 3.14.

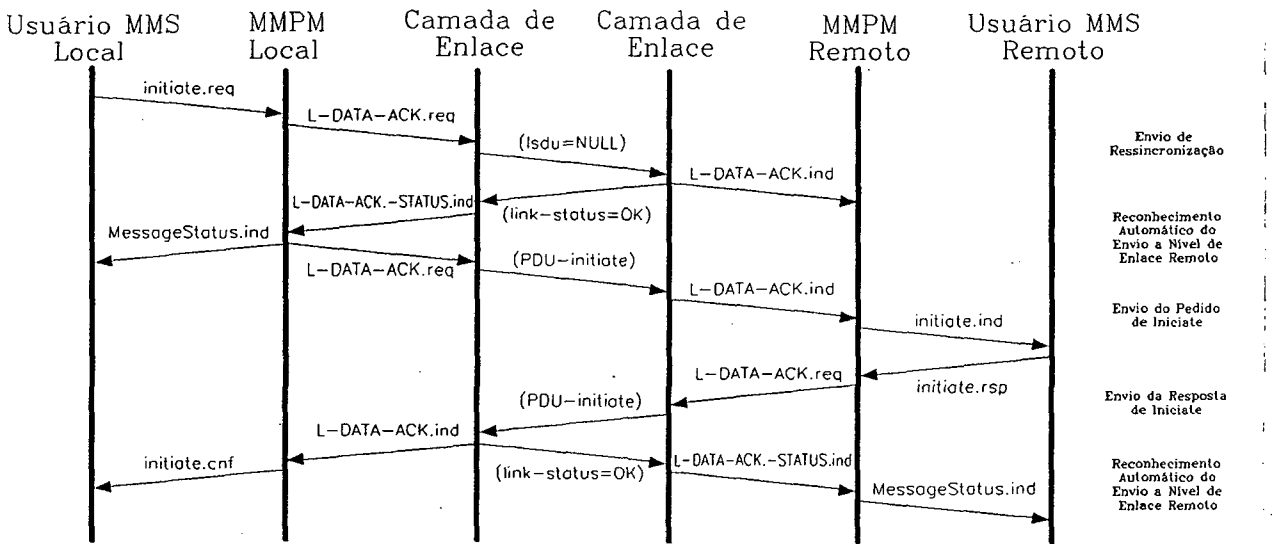


Figura 3.14 Associação de Aplicação Explícita

Mesmo utilizando a AA explícita, os parâmetros do serviços de controle de associação *A_Associate* não são negociados. Este fato causa, entre outros, a não negociação da sintaxe abstrata (negociada pelo parâmetro *context_name*). A sintaxe abstrata usada no MMS é fornecida pelo padrão genérico [ISO 88] e complementada pelos diversos *Companions Standards*. Uma possibilidade é a utilização de AE-Tipos que suportem ou a sintaxe abstrata definida no padrão MMS genérico, ou esta mesma sintaxe e complementações provenientes de um *Companion Standard*.

Na comunicação sem associação, as informações que seriam negociadas no estabelecimento da associação são substituídas por valores *default*. Em [MAP 88] é definido que o conjunto de serviços MMS disponíveis no modo sem associação são os seguintes: *Read, Write, InformationReport, Status, Cancel, Identify, UnsolicitedStatus, EventNotification, AcknowledgeEventNotification, Input e Output*.

Os LSAP, no caso de coexitirem comunicações com associação e sem associação, devem ter definidos os tipos de comunicação a que estão destinados.

É importante salientar que uma associação de aplicação explícita é necessária quando ao menos uma das seguintes funções é exigida ([MAP 88]):

- Nomes de escopo específico de AA devem ser definidos sobre a rede;
- parâmetros de comunicação, que podem ser negociados com a operação *Initiate*, devem ser alterados (de seus valores *default*) durante uma sessão de comunicação;
- é necessário pré-alocar recursos para comunicação entre dois usuários MMS.

Codificação/Decodificação ASN.1

A codificação/decodificação ASN.1 [ISO 87a] de PDUs, que no MMS ISO é provida pela camada de Apresentação, na arquitetura Mini-MAP deve ser fornecida na propria camada de aplicação.

3.5.3 Aspectos da Utilização da MMSI numa Arquitetura Mini-MAP

A seguir serão levantados alguns aspectos acerca da utilização da MMSI numa Arquitetura Mini-MAP:

Ausência do ASE ACSE

Na MMSI, tanto os modelos de comunicação orientados a conexão (no Mini-MAP, com associação explícita) quanto o sem conexão devem ser suportados. Portanto a utilização da MMSI numa arquitetura Mini-MAP permite que estas duas formas de comunicação coexistam. Neste caso é necessário definir previamente o tipo de comunicação que cada AE utilizará.

Devido também a inexistência do ACSE, os parâmetros das funções da API, que seriam mapeados nos parâmetros dos serviços de controle de associação, tem valores irrelevantes.

Ausência da Camada de Apresentação

Devido a ausência da camada de apresentação, o MMPM terá como funcionalidade adicional a codificação e decodificação ASN.1 de PDUs.

Registro de Entidade de Aplicação

Para que duas AE-I se comuniquem, é necessário o registro em diretório do endereçamento de suas AEs. Para possibilitar a comunicação com outra AE, o LSAP destino deve ser conhecido a priori, obtido através de um serviço de diretório, gerenciamento local, ou simplesmente por acordo entre os programadores das duas aplicações [MAP 88].

No registro de AE são definidos parâmetros de gerenciamento de conexão específicos de AE, que devem ser definidos quando de uma implementação. A norma MAP define os seguintes parâmetros:

- número de associações por invocações de AE: para o MMS, o número máximo é um;
- o número de invocações de AE por LSAP: cada par de endereço LLC (LSAP) e MAC podem ter no máximo oito associações.

Endereçamento

Os parâmetros de entrada *my_dir_name* da função AE ACTIVATION e *called_dir_name* da função CONNECT, fornecem o nome registrado no diretório da AE. No caso Mini-MAP, estes parâmetros indicam diretamente o endereço de enlace (LSAP) reservados a esta AE. Como o diretório Mini-MAP (ou dicionário de objetos Mini-MAP) apenas associa um nome simbólico a um endereço Mini-MAP (endereço MAC e LSAP), estes parâmetros necessitam apenas conter o valor do parâmetro *ae_qualifier* da estrutura *Dir_ava* (declaração dos valores dos atributos) contida na estrutura do *Dir_rdn* (nome distinguível relativo) da estrutura *Dir_dn* (nome distinguível do diretório).

3.6 Conclusão

Neste capítulo, os padrões MMS e MMSI foram descritos e comentados. Durante esta apresentação foram identificados pontos não definidos na especificação da MMSI e que necessitam decisões de implementação.

Foram também apresentados aspectos de utilização dos serviços MMS e da MMSI em uma arquitetura Mini-MAP, definindo algumas funcionalidades adicionais a especificação MMSI do MAP para permitir a sua operação numa arquitetura Mini-MAP. Além disso, foram vistas as restrições de uso dos serviços MMS nesta arquitetura.

Após a apresentação dos principais conceitos encontrados no MMS, MMSI e Mini-MAP, a proposta de um modelo de implementação do padrão MMS numa arquitetura Mini-MAP, de acordo com a especificação MMSI, será objeto de estudo dos próximos capítulos.

CAPÍTULO 4

MODELO DE IMPLEMENTAÇÃO DO PADRÃO MMS

4.1. Introdução

Este capítulo tem por objetivo apresentar um modelo para a implementação do padrão MMS numa arquitetura Mini-MAP, seguindo o padrão da MMSI. A definição de um modelo de implementação, tem como objetivo, fornecer uma proposta para futuras implementações do MMS numa arquitetura Mini-MAP. Uma especificação Estelle [ISO 87b] permitirá a formalização e posterior verificação deste modelo através de uma simulação.

4.2. Modelo de Implementação Proposto

O modelo de implementação proposto nesta dissertação, é o resultado da interpretações do modelo da Interface para o Programa de Aplicação MMSI definido na norma MAP.

Apesar da norma MAP pré-definir as funcionalidades do Provedor de Serviços, opções devem ser feitas quando de uma implementação. Alguns pontos que necessitam de tomada de decisão em uma implementação já foram definidos no capítulo anterior e são resumidas a seguir:

- mapeamento dos blocos funcionais do Provedor de Serviços em tarefa;
- definição das operações envolvidas na ativação/destruição das invocações de AE;
- definição de uma técnica de identificação das AE-I e associações, ou seja, a definição de uma semântica para os parâmetros *ae_label* e *connection_id*.

O modelo proposto é mostrado na figura 4.1, sendo que as tarefas envolvidas serão descritas a seguir.

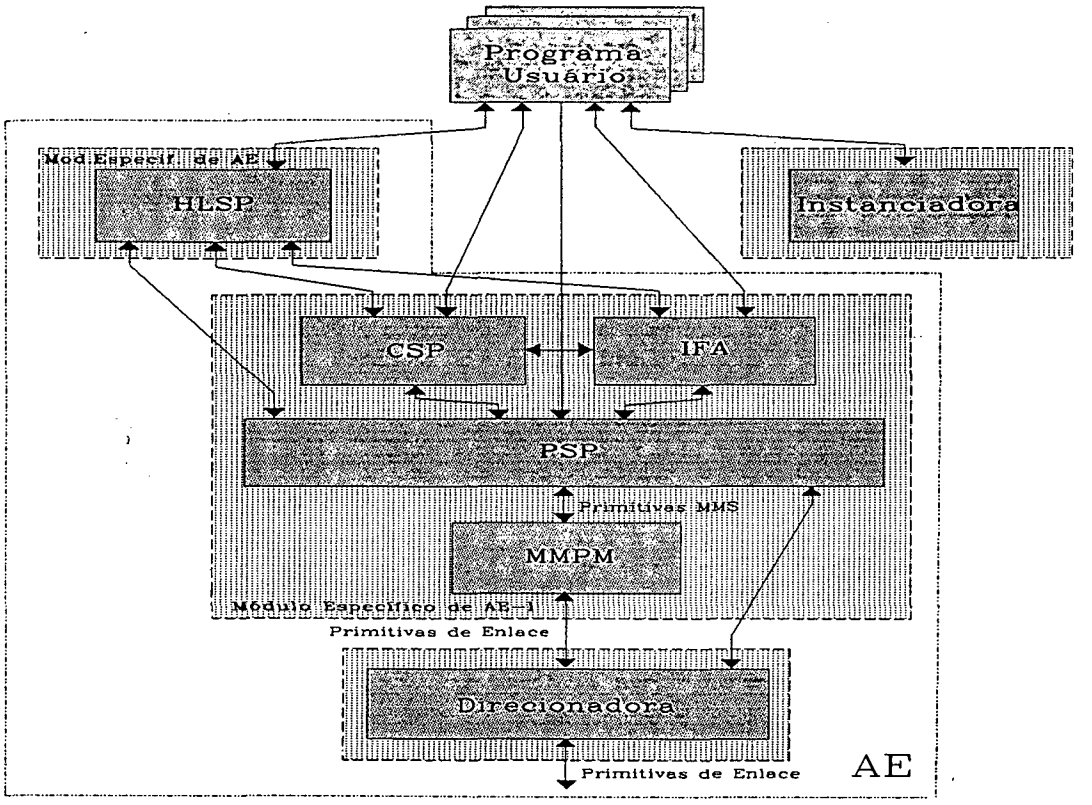


Figura 4.1 Modelo de Implementação Proposto

4.2.1 Programa Usuário MMS

O Programa Usuário pode ser composto de uma ou mais tarefas, que fazem uso dos serviços MMS através da biblioteca de funções da MMSI, conforme interfaceamento definido no capítulo 3. No modelo proposto, caso uma chamada a biblioteca da MMSI seja aceita, um pedido será enviado a instância do módulo provedor de serviço correspondente. A forma de identificação desta instância será descrita a seguir.

4.2.2 Provedor de Serviços da Interface

Para permitir que o Programa Usuário possa interagir assincronamente com o Provedor de Serviços é necessário que ambos sejam implementados como tarefas (unidades escalonáveis) distintas. Sendo assim, a parte do Provedor de Serviços da interface deve ser mapeado ao menos em uma tarefa.

A partir da constatação na norma, de que os vários blocos funcionais do modelo da interface podem agir de forma concorrente no fornecimento de vários serviços, cada um deste blocos será considerado uma tarefa.

A norma MAP define para cada bloco funcional, procedimentos associados a um determinado escopo, que são específicos de AE e de associação (visto no capítulo anterior). Por exemplo, no caso de provedores específicos de associação são definidas ações sobre uma única associação. O modelo de implementação proposto segue estes escopos, para tanto, considerando que o conceito de módulo no modelo proposto é a unidade a ser instanciada, as tarefas provedoras de serviços são agrupadas da forma seguinte:

- a tarefa provedora HLSP estará contida no **Módulo Específico de AE.** No modelo proposto existirá somente uma instância deste módulo para cada AE;
- as tarefas provedoras CSP, PSP, IFA e MMPM são encapsuladas por um **Módulo Específico de Associação**, existindo uma instância deste módulo para cada associação. Desta forma, os recursos associados a uma associação estão presentes em uma instância deste módulo. Como no modelo proposto uma AE-I suporta apenas uma associação (respeitando o especificado em [ISO 88]), este módulo é também chamado de **Módulo Específico de AE-I.**

As ações das tarefas Provedoras de Serviços são aquelas definidas no padrão MAP. Além destas ações, o modelo de implementação define funcionalidades adicionais, requeridas à arquitetura Mini-MAP, que são a de codificação/decodificação ASN.1 de PDU, no caso do PSP e a definição do nível de prioridade da mensagem utilizada no pedido de associação, no caso do HLSP. Este último utiliza-se de parâmetros de gerencia de AE.

No modelo de implementação proposto, uma AE será representada, em tempo de execução, por uma instância do módulo Específico de AE, uma ou mais instâncias do módulo Específico de AE-I, conforme o número de invocações de AE sendo tratada no instante considerado e ainda pela tarefa Direcionadora (item 4.2.4).

4.2.3 Operações de Gerência Local na Ativação/Destruição de AE: Tarefa Instanciadora

No modelo proposto, parte das operações de gerenciamento local, durante o pedido de AE ACTIVATION, consiste na instanciação dos módulos Específicos de AE e de AE-I associados ao tipo da AE a ser declarada. Para tanto, é necessário um módulo composto por uma tarefa Instanciadora, responsável pelas instanciações/destruições destes módulos. A tarefa Instanciadora é considerada um provedor de serviços da interface adicional. A seguir, as operações de ativação e destruição de AE disponíveis a partir da tarefa Instanciadora são apresentados:

Ativação de AE

Durante a execução da função AE ACTIVATION, a Tarefa Instanciadora gera uma instância do módulo Específico de AE-I e caso seja a primeira invocação desta AE, uma instância do módulo Específico de AE. No modelo proposto, o parâmetro de saída *ae_label* é o identificador da instância do módulo Específico de AE-I gerada. O valor do identificador de uma conexão (*connection_id*) será o mesmo de *ae_label*, já que é permitido apenas uma associação para cada AE-I.

A faixa de valores de *ae_label,* num total de oito para cada LSAP (associados ao níveis de prioridades da mensagem, como visto no capítulo anterior), deve ser previamente definida para cada AE. Esta informação faz parte dos parâmetros de gerenciamento de AE. O identificador da instância do módulo Específico de AE também faz parte destes parâmetros.

A definição desta semântica para os parâmetros *ae_label* e *connection_id* permite que:

- as chamadas de funções que envolvam ações específicas de AE têm como parâmetro de entrada *ae_label*. No modelo de implementação, *ae_label* identifica a instância do módulo Específico de AE-I. Este parâmetro possibilita a identificação da instância do módulo Específico de AE (onde o serviço será encaminhado inicialmente). Esta identificação é realizada verificando a qual AE pertence o valor de *ae_label*. Os parâmetros de gerenciamento de AE permitem esta verificação;
- as chamadas de funções que envolvam ações específicas de associação têm como parâmetro de entrada *connection_id*, que no modelo de implementação identifica diretamente a instância do módulo Específico de AE-I que executará o serviço.

Destruição de AE

No modelo proposto, a função AE DEACTIVATION executa a operação inversa a anterior, ou seja, destroi a instância do módulo Específico de AE-I identificado pelo parâmetro de entrada *ae_label* e caso seja a última invocação desta AE, destroi a instância do módulo Específica de AE.

A desativação de um AE-I é executada somente se não há uma associação estabelecida nesta AE-I e nenhum pedido de LISTEN pendente. A verificação da existência deste é realizada na etapa de checagem geral de erros, através dos parâmetros de gerência da AE, identificada através do parâmetro *ae_label.*

4.2.4 Direcionamento de Informações nos LSAP

Devido ao fato de poder existir várias instâncias de módulos Específicos de AE-I associados a cada LSAP, foi introduzido no modelo uma Tarefa Direcionadora, responsável pelo encaminhamento de mensagens recebidas nos LSAPs a instância apropriada do módulo Específico de AE-I da AE proprietária do LSAP. A identificação do módulo Específico de AE-I é baseada no nível de prioridade da mensagem, que identificam as Associações de Aplicação.

As funcionalidades fornecidas pela tarefa Direcionadora são específicas de AE, portanto deverá existir uma instância da Tarefa Direcionadora para cada AE.

Os procedimentos para o direcionamento de mensagens proposto no modelo de implementação são os seguintes:

- no pedido de estabelecimento de uma AA, a instância do módulo Específico de AE-I que fez o pedido será associada a um nível de prioridade de mensagem, assim, este nível de prioridade será reservada a esta associação;
- na recepção de um pedido de estabelecimento de uma AA, este será encaminhado a primeira instância do módulo Específico de AE-I sobre a qual foi chamada a função de escuta LISTEN; caso não haja um LISTEN pendente sobre uma das invocações da AE ligada ao LSAP que recebeu o pedido ou ainda, nenhuma da AE-I desta AE foi declarada (não há nenhuma instância do módulo Específico de AE-I associado a AE do LSAP), haverá a rejeição do pedido de estabelecimento por parte da Tarefa Direcionadora;

- após estabelecida a associação, as mensagens recebidas serão encaminhadas à AE-I de acordo com seus níveis de prioridade;
- No envio de uma PDU de resposta positiva do pedido de *conclude*, ou do serviço *abort*, a ligação entre o nível da mensagem e a associação será desfeita;

Quando o Programa Usuário chama a função de escuta LISTEN, o identificador da instância do módulo Específico de AE-I (*ae_label*), será encaminhado a Tarefa Direcionadora associada ao LSAP da AE, possibilitando uma futura identificação da instância que possui um LISTEN pendente. Este procedimento é uma ação adicional da tarefa PSP.

4.3 Especificação e Verificação do Modelo Proposto

Neste ítem será apresentado uma especificação Estelle do modelo proposto e a verificação deste através da simulação da especificação Estelle.

4.3.1 Especificação do Modelo Proposto

Afim de fornecer uma descrição formal do modelo de implementação proposto, neste ítem será apresentado a sua especificação Estelle. Entretanto esta especificação não é completa, sendo seu objetivo principal, a especificação das funcionalidades propostas pelo autor desta dissertação.

A figura 4.2 apresenta a especificação Estelle de um Processo de Aplicação (AP) Cliente ou Servidor, segundo o modelo proposto, e a figura 4.3 mostra a estrutura hierárquica destes módulos. No apêndice A são apresentados detalhes da especificação numa pseudo-linguagem próxima de Estelle.

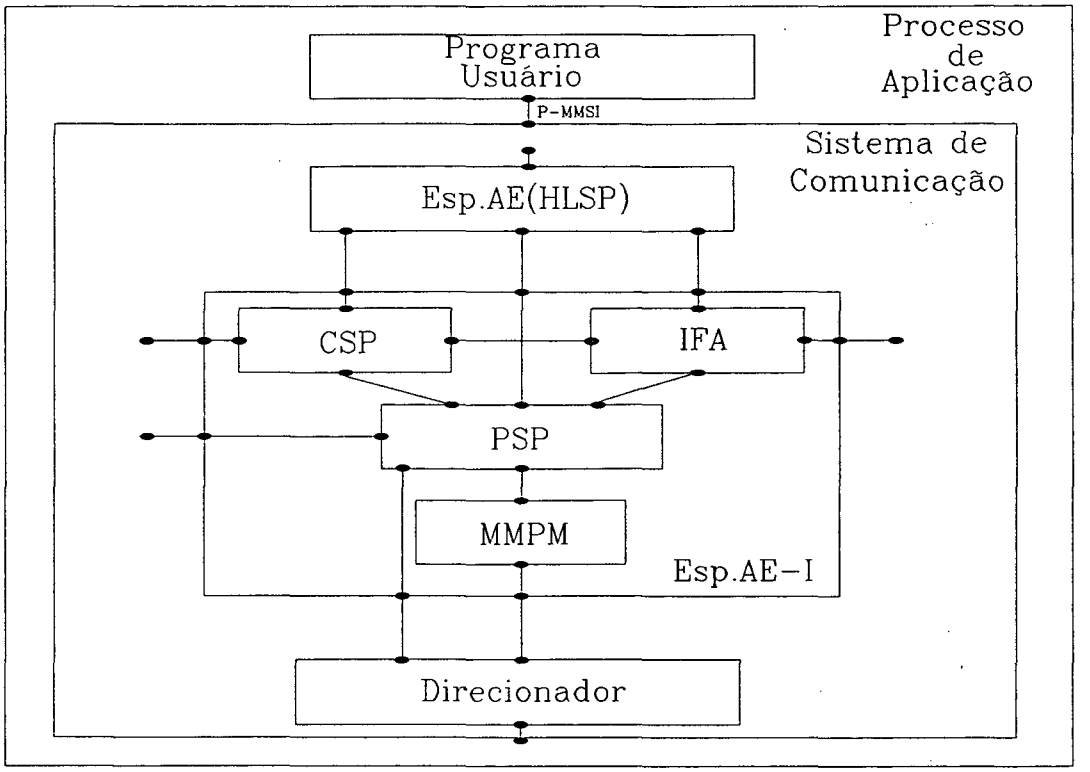


Figura 4.2 Especificação Estelle de um AP adotando o modelo proposto

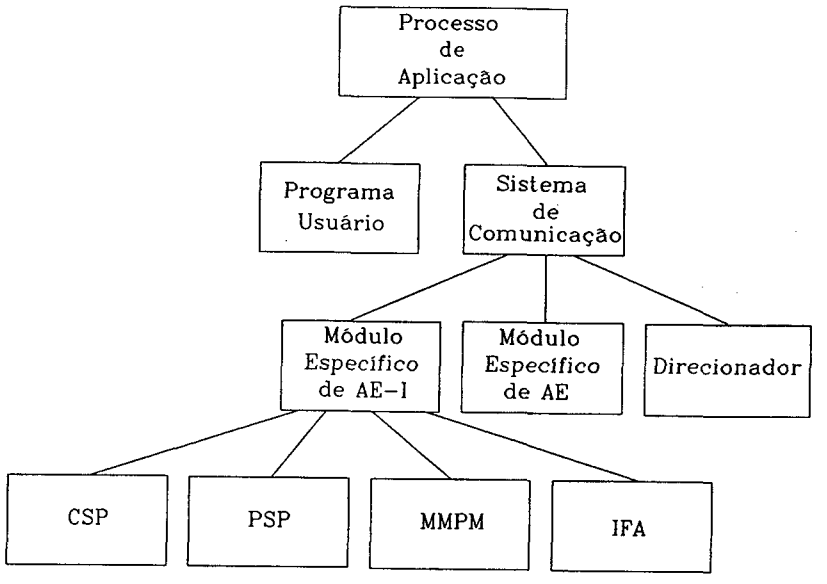


Figura 4.3 Estrutura hierárquica do modelo de implementação

Na especificação apresentada na figura 4.2 haverá várias iniciações dos módulos Estelle Específico de AE, Específico de AE-I e Direcionador, cujos procedimentos já foram definidos neste capítulo.

A seguir são apresentados alguns aspectos relevantes desta especificação:

Iniciação Estática

Na iniciação estática, são iniciados os módulos Estelle "Programa Usuário" e "Sistema de Comunicação", bem como definidas suas conexões. A iniciação do módulo "Sistema de Comunicação" tem como parâmetros, as AE registradas na estação e informações de gerência específicas de cada AE registrada, como nome da AE, seu LSAP com os respectivos níveis de prioridade das mensagens correspondentes aos diferentes pedidos de associações.

Módulo Estelle "Programa Usuário MMS"

O módulo Estelle do Programa Usuário MMS inclui a chamada de função da biblioteca da MMSI. Nesta especificação, a chamada de função da MMSI e sua resposta são associadas ao envio e recepção de mensagens contendo respectivamente os parâmetros de entrada e de saída da função. O módulo Estelle "Programa Usuário MMS" interage com o Módulo Estelle "Sistema de Comunicação" para o envio e recepção destas mensagens. Esta interação é realizada através do ponto de interação P_MMSI (ver figura 4.2), onde as declarações de interações deste ponto definem os parâmetros especificados nas funções da MMSI.

Módulo Estelle "Sistema de Comunicação"

O módulo Estelle "Sistema de Comunicação" inclui as funcionalidades da Tarefa Instanciadora definida no modelo. Também especifica o envio do pedido ao provedor correspondente, existindo pontos de interação que permitem a comunicação com todas as instâncias possíveis.

Nesta especificação, como será visto a seguir, o módulo Estelle "Direcionador" e "HLSP" necessitam conectar pontos de interação dos módulos Específico de AE-I e Direcionador. Em Estelle, as conexões entre pontos de interação de módulos de mesmo nível só podem ser definidas pelo módulo pai. Devido a esta restrição, o módulo Estelle

"Sistema de Comunicação" é responsável pela conexão entre os pontos de interação dos módulos "Específico de AE-I" e "Direcionador".

Módulo Estelle "Específico de AE"

O módulo Estelle Específico de AE contém as operações da tarefa HLSP. Nesta especificação, estas operações estão associados ao estabelecimento de uma associação somente. Estes serviços serão encaminhados ao módulo Estelle Específico de AE-I.

Uma operação importante do módulo Estelle "Específico de AE" quando do pedido de CONNECT, é a atribuição de um nível de prioridade às mensagens a serem enviadas/recebidas sobre uma associação a ser estabelecida. Nesta especificação, o módulo Estelle "Direcionador" apresenta, entre outros, oito pontos de interação, cada um associado a um nível de prioridade de mensagem. A atribuição de uma prioridade a associação se dá pela conexão de um ponto de interação do módulo Estelle MMPM ao ponto de interação do módulo Estelle "Direcionador" associado ao nível de prioridade. Esta conexão deve ser solicitada ao Módulo pai "Sistema de Comunicação".

Módulo Estelle "Específico de AE-I"

O módulo Estelle Específico de AE-I contém a iniciação dos módulos Estelle dos Provedores de Serviços.

Módulos Estelle dos Provedores de Serviços

Os módulos Estelle dos Provedores de Serviço especificam as funcionalidades dos blocos funcionais (CSP, PSP, IFA e MMPM) do modelo da interface.

Módulo Estelle "Direcionador"

O módulo Estelle Direcionador especifica as funcionalidades da tarefa Direcionadora.

Na chegada de uma mensagem remota vários casos possíveis devem ser analisados:

- a prioridade da mensagem esta ativa (ou seja, o ponto de interação está conectado a uma instância do módulo "Estelle Específico de AE-I"), então a mensagem é enviada neste ponto;
- a mensagem corresponde a um pedido de associação e existe um pedido de LISTEN pendente, então o módulo Estelle "Direcionador" solicita uma conexão entre o ponto de interação associado a prioridade da mensagem e o ponto de interação da instância do módulo Estelle "Específico de AE-I" que possui o LISTEN pendente; esta instância é identificada pelo *ae_label* passado ao módulo Direcionador no pedido de LISTEN.

Outras situações podem ser vistas no apêndice A.

4.3.2 Verificação do Modelo Proposto

A verificação do modelo de implementação e de sua conformidade com o comportamento de referência definido na norma MMSI e MMS são aspectos importantes para consolidar o modelo de implementação como base para uma metodologia segura para a implementação do MMS. Neste ítem serão apresentados os resultados da verificação do modelo proposto, realizada a partir da simulação de sua especificação Estelle, utilizando o Simulador de Especificação Estelle ESTIM [Saqui 90a] desenvolvido no LAAS-CNRS (França). O apêndice C apresenta uma descrição do simulador ESTIM. O ESTIM foi desenvolvidor para a linguagem Estelle*, dialeto do Estelle cujas características estão apresentadas também no apêndice C. Neste trabalho utilizou-se unicamente as características padrão Estelle da linguagem Estelle*.

O objetivo deste etapa do trabalho foi o de validar os fluxos de controle e informações no modelo de implementação, principalmente afim de verificar se as propostas de implementação feitas pelo autor não apresentam erros e estão em conformidade com a MMSI e com as características da arquitetura Mini-MAP. Os principais pontos a serem verificados são:

- a correspondência entre os parâmetros de entrada nas chamadas de funções da biblioteca da interface e o que foi definido na especificação da MMSI;
- a operação de ativação/destruição de AE;
- a identificação das AE-I e associações;
- o direcionamento de mensagens recebidas nos LSAPs.

Devido a restrições de memória do ambiente de simulação, a simulação Estelle da especificação do modelo de implementação foi realizada em duas etapas. Primeiramente foi especificado um ambiente mais complexo com Processos de Aplicação, a fim de analisar a fase de estabelecimento de associações e de pedido de outros serviços MMS. Numa segunda etapa, especificou-se uma aplicação mais simples, afim de analisar o encerramento de associação de aplicação e a desativação de AE-I.

Na primeira etapa da simulação definiu-se a comunicação entre três Programas Usuários, como visto na figura 4.4: um programa C1 cliente de CS2 e S3, sendo que o programa CS2 é também cliente de S3. Nesta aplicação, as AE-Is do Servidor S3 são oriundos da mesma AE, enquanto as AE-Is do Cliente C1 e Cliente/Servidor CS2 são de AE diferentes. O serviço MMS a ser fornecifo neste exemplo é o serviço *Status* (do servidor CS2 para o cliente C1 e do servidor S3 para o cliente CS2). Este exemplo foi escolhido devido à existência na mesma especificação de invocações da mesma AE e de diferentes AEs, o que abrange várias situações de interesse para a verificação do modelo.

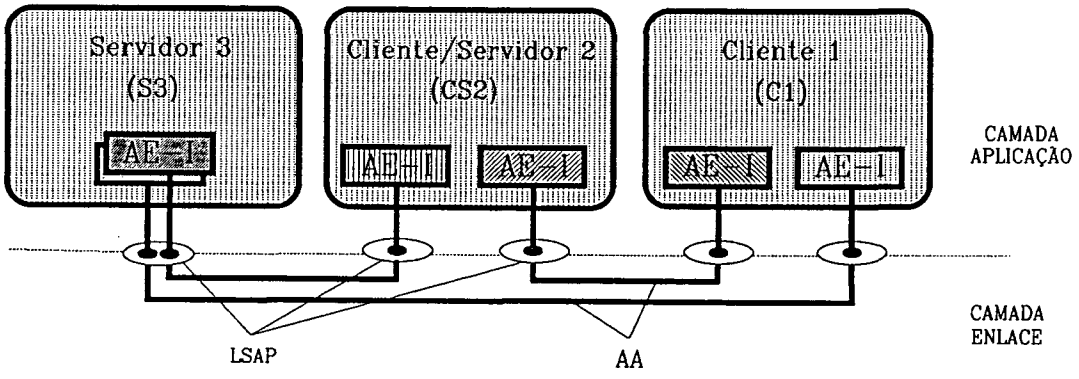


Figura 4.4 Representação da comunicação no exemplo escolhido

A figura 4.5 mostra o nível de abstração mais elevada da Especificação Estelle da aplicação que foi simulada. Nesta figura, os módulos Processo de Aplicação são refinados conforme apresentado na figura 4.3.

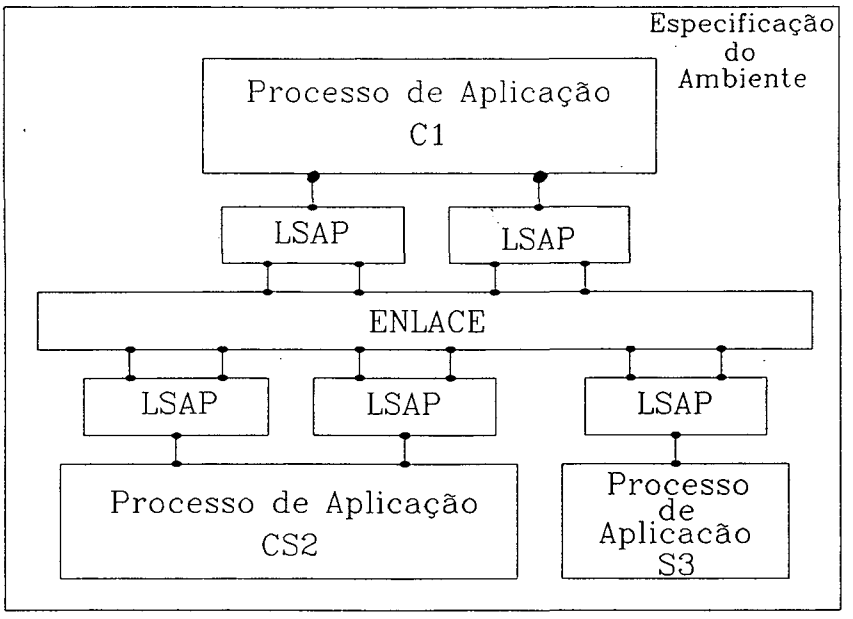


Figura 4.5 Especificação total do exemplo escolhido

O apêndice B apresenta um outro nível de abstração (inferior ao precedente) resultante de um refinamento do nível anterior dos módulos Programas Usuários (C1, CS2 e S3), LSAP e Enlace. A seguir, as funcionalidades de cada um destes módulos serão descritas.

A) Módulo Estelle "Cliente 1"

Nesta especificação, o módulo Estelle "Cliente 1" requer o estabelecimento de uma associação (CONNECT) com os módulos Estelle "Cliente/Servidor 2" e "Servidor 3" a fim de pedir o serviço STATUS a estes dispositivos. Para tanto, inicialmente este módulo deve pedir a ativação de uma AE-I de cada uma das duas AEs registradas.

B) Módulo Estelle "Cliente/Servidor 2"

O módulo "Cliente/Servidor 2" requer a ativação de uma AE-I de cada uma de suas AE. Na AE-I destinada a comunicação com o Cliente 1 ele aguarda e responde a um pedido de associação (LISTEN e ANSWER), após envia o pedido de IRECEIVE, a fim de receber um pedido de serviço, que no caso será *Status*. Após esta indicação, responde este pedido através de STRESPONSE. Na AE-I destinada a comunicação com Servidor 3, o Cliente/Servidor 2 pede o estabelecimento de uma associação com Servidor 3. Após estabelecida a AA, o módulo Cliente/Servidor 2 solicita o serviço STATUS ao Servidor 3.

C) Módulo Estelle "Servidor 3"

Nesta aplicação, o módulo "Servidor 3" deve ativar ao menos duas AE-Is de sua AE. Afim de declarar seu desejo de recepção de um pedido de associação sobre as AE-Is, o módulo Servidor 3 requer pedidos de LISTEN sobre elas. Na resposta de um LISTEN, responde ao pedido de associação através de ANSWER. Após o estabelecimento de uma AA sobre uma AE-I, este módulo envia o pedido de IRECEIVE, a fim de receber um pedido de serviço, que nesta especificação é o serviço de *Status*. Após a indicação deste serviços, o módulo Servidor 3 responde-o através do envio de STRESPONSE.

D) Módulo Estelle "LSAP"

O módulo Estelle "LSAP" define o interfaceamento entre a camada de aplicação e a de enlace, recebendo pedidos de enlace (*l_data_ack.req*) e enviando o reconhecimento imediato (*l_data_ack_status.req*), o elemento *Obtain Reply* não sendo especificado. Nesta especificação, a fim de simular a ocorrência de erros de enlace, existe a possibilidade tanto de sucesso como de fracasso na recepção de um pedido de serviço de enlace, que é notificado ao usuário remoto através do reconhecimento imediato (parâmetro *status_link*).

E) Módulo Estelle "Enlace"

O módulo Estelle "Enlace" é bastante simplificado, e representa unicamente o encaminhamento de uma mensagem ao seu destino.

Para a segunda etapa da simulação, especificou-se dois Programas Usuários (por exemplo C1 e CS2 ou CS2 e S3) estabelecendo uma associação, encerrando-a e finalmente, desativando as AE-Is utilizadas.

Através da simulação, verificou-se o funcionamento adequado da implementação baseada no modelo proposto, a conformidade desta a norma e a correção das decisões de implementações feitas pelo autor. Em particular no que diz respeito a:

A) Gerência de Ativação/Destruição de AE

A gerência de ativação/destruição de AE foi executada com sucesso utilizando apenas os parâmetros passados nas chamadas das funções AE ACTIVATION e AE DEACTIVATION, conforme proposta do autor.

B) Identificação das AE-Is e Associações

Foi possível realizar as identificações de AE-Is e associações segundo a forma definida pelo autor, utilizando apenas os parâmetros *ae_label* e *id_connection* existentes nas funções da MMSI que necessitam destas identificações;

C) Direcionamento de mensagens dos LSAPs

A simulação da especificação Estelle foi de grande ajuda para modificar a proposta do modelo inicial, conforme será mostrado a seguir.

Inicialmente na especificação do modelo de implementação, uma instância do módulo "Direcionador" só era gerada na primeira ativação da AE, visando a alocação deste recurso apenas quando era requerido pelo Programa Usuário local. Nas simulações, verificou-se a possibilidade de chegada de mensagens nos módulos LSAPs antes da ativação da AE, ocorrendo perdas de mensagens. Para evitar a ocorrência deste problema, definiu-se que a iniciação do módulo Direcionador ocorrerá na iniciação do sistema de comunicação, possibilitando agora que, caso uma mensagem chega antes de uma ativação de AE, o pedido associado seja rejeitado. Após esta correção no modelo, o direcionamento de mensagens recebidas nos LSAPs ocorreu com sucesso, utilizando apenas o nível de prioridade da mensagem, conforme especificado na arquitetura Mini-MAP.

4.4 Conclusão

Neste capítulo foi apresentado o modelo de implementação do padrão MMS numa arquitetura Mini-MAP, seguindo a especificação da MMSI, que era o objetivo principal desta dissertação.

A adoção das mesmas funcionalidades especificadas nos blocos funcionais do modelo da Interface para o Programa de Aplicação MAP simplificou a definição deste modelo de implementação. Apesar desta pré-definição, vários outros pontos devem ser definidos numa implementação. Após identificados estes pontos, foi definido um conjunto de decisões de implementação. Também foram definidas funcionalidades requeridas por uma implementação na arquitetura Mini-MAP. As principais decisões tomadas no decorrer deste trabalho dizem respeito a:

- mapeamento dos blocos funcionais do Provedor de Serviços em tarefa;
- definição das operações envolvidas na ativação/destruição das invocações de AE;
- técnica de identificação das AE-I e associações, ou seja, a definição de uma semântica para os parâmetros *ae_label* e *connection_id*;
- direcionamento de mensagens recebidas nos LSAPs.

Após feita a definição do modelo de implementação, este foi especificado em Estelle, proporcionando uma definição mais estruturada e mais detalhada das decisões de implementação.

A fim de consolidar o modelo de implementação como base para uma metodologia segura para a implementação do MMS, foi realizado a verificação deste modelo, através da simulação de sua especificação Estelle.

No próximo capítulo, o modelo de implementação proposto será finalmente testado numa implementação simplificada.

CAPÍTULO 5

ASPECTOS DE IMPLEMENTAÇÃO

5.1 Introdução

Este capítulo tem por objetivo apresentar características de implementação do padrão MMS que utiliza como base o modelo de implementação proposto.

A necessidade de concorrência, tanto na implementação do serviço de comunicação como nos processos de aplicação, leva a utilização, como suporte básico do modelo, de um núcleo que proporcione um ambiente multitarefas, com características capazes de satisfazer restrições de tempo. Neste capítulo serão apresentadas algumas características necessárias a um núcleo de tempo real (NTR), que possa servir como software de base para uma implementação segundo o modelo de implementação proposto.

Para testar o modelo proposto numa implementação real, e ainda, levantar detalhes de implementação, foi realizada uma implementação simplificada da MMSI. Esta implementação e os principais resultados também serão apresentados neste capítulo.

5.2 Características Gerais de um Suporte Básico para o Modelo Proposto

As restrições de tempo associadas aos softwares de comunicação que implementam o padrão MMS são muito importantes, em consequência, o NTR adotado deve apresentar características capazes de satisfazer estas restrições.

A especificação da interface API não cita nenhum padrão com relação ao sistema operacional, apesar do fato de que se um ambiente de sistema operacional padrão for especificado pela ISO, a especificação da interface migrará certamente para tal ambiente. Na fase inicial de especificação da API foi cogitada a adoção da proposta de padrão POSIX (*Portable Operating System Interface*) [Corwin 90], porém esta adoção foi abandonada, pois não era do escopo da comissão responsável pela especificação da interface a adoção de um sistema operacional. A adoção de tal padrão, permitiria que um

código fonte dos softwares de comunicação implementados possa ser transportado para outros ambientes que adotem o mesmo padrão.

A seguir, serão apresentadas algumas das funcionalidades requeridas por um NTR a fim de possa servir como software de base numa implementação utilizando como suporte o modelo de implementação proposto.

- Instanciação

No modelo de implementação, existe várias instanciações dinâmicas dos módulos Específico de AE, Específico de AE-I e Direcionador, sendo assim, o NTR deve suportar esta característica, que é a instanciação dinâmica de módulos de mesmo tipo.

- Comunicação Inter-processos (IPC)

Este é um dos elementos mais importantes do NTR, pois tem grande influência no desempenho do sistema, devido principalmente ao grande volume de informações trocadas entre tarefas do software de comunicação. Portanto é necessário que o IPC gere o menor *overhead* possível na comunicação.

Basicamente existem duas classes de mecanismos de comunicação:
- Chamada de Procedimento;
- Troca de Mensagem.

Algumas observações retiradas de [Nacamura 88] a respeito destas duas classes são citadas a seguir:
- As duas classes são duais;
- Os programas de uma classe são logicamente idênticos ao da outra ao de outra classe;
- O desempenho de um programa considerando-se os tamanhos das filas, tempo de espera, velocidade de serviços, etc, são idênticos ao ser programa dual desde que se considere a mesma estratégia de escalonamento.

Apesar destas considerações, devido a característica de instanciação dinâmica de módulos de mesmo tipo, o IPC deve prover mecanismos para identificação dos processos em tempo de execução, portanto deve ser adotado a comunicação por troca de mensagem, já que esta classe fornece esta facilidade.

O uso de primitivas de troca de mensagens que provêem, além da comunicação, várias formas de sincronismos entre os processos comunicantes, evita que os processos tenham que fazê-lo explicitamente.

- Gerência de Eventos

Outro ponto a considerar é o fornecimento de um mecanismo para gerência de eventos, que é a forma definida na interface padrão para promover a notificação de finalização de um pedido de serviços, como apresentado no capítulo 3. Este mecanismo não necessariamente deve ser fornecido pelo NTR. Pode haver outras formas de implementação, uma destas será vista na descrição da implementação realizada. A utilização ou não de um mecanismo de eventos dependerá unicamente do desempenho desejado.

- Escalonamento

A política de preempção e a capacidade de mudança dinâmica de prioridade possibilitam um escalonamento razoavelmente adaptada para os requisitos de tempo. Outras funcionalidades requerida para o núcleo são os mecanismos de suspensão e ativação de tarefas controladas por tempo.

- Gerência de Memória

O NTR deve prover mecanismos de alocação e dealocação dinâmica de áreas de memória, necessário na alocação e dealocação das estruturas DCBs, apresentadas no capítulo 3.

5.3 Aspectos de Implementação

Para testar o modelo proposto como esquema para uma implementação, foi realizada uma implementação simplificada da MMSI. Esta implementação permitiu também discutir detalhes de uma implementação real, que não foram abrangidos no capítulo anterior.

Neste ítem também serão levantados alguns problemas encontrados nas norma MMS e MMSI e apresentadas soluções adotadas nesta implementação.

A implementação da MMSI foi realizada em uma rede de microcomputadores compatíveis IBM-PC, sendo que suas principais simplificações estão descritas a seguir:

- Simulação da camada enlace a partir de uma rede comercial disponível (do tipo Ethernet), onde as mensagens serão enviadas à estação remota através do mecanismo de comunicação remota disponível no núcleo adotado;
- Ausência do Codificação/Decodificação ASN.1 de PDUs. Nesta implementação, a PDU enviada na comunicação remota é uma variável do tipo "estrutura", contendo os mesmos elementos da especificação ASN.1 da PDU MMS (MMSpdu), fornecida na parte 2 de [ISO 88] e mais alguns outros elementos de controle, a fim de permitir a identificação de parâmetros;
- Implementação apenas das seguintes funções da MMSI: AE ACTIVATION, CONNECT, LISTEN e ANSWER, para o estabelecimento de uma associação; STATUS, INDICATION RECEIVE e STATUS RESPONSE para a Leitura de Status de um dispositivo remoto, além das funções de suporte WAIT, DYNAMIC INITIALIZE DCB e DYNAMIC FREE DCB.
- O sistema de comunicação será unicamente orientado a associação, podendo mais tarde suportar a comunicação sem associação;
- Outra simplificação adotada consiste em utilizar unicamente estações que participam da passagem de ficha, com isso o ASE-*Obtain Reply* não será necessário.

Apesar destas simplificações, a idéia que orientou esta etapa do trabalho foi o de manter a mesma estrutura de software que aquela presente em uma implementação real.

A seguir, serão apresentadas as principais características da implementação realizada.

5.3.1 Núcleo de Tempo Real Adotado

O NTR adotado foi o existente no LCMI [Nacamura 88] e componente do projeto Ambiente de Desenvolvimento e Execução de Software para Aplicações Distribuidas (ADES), que suporta o princípio da decomposição modular, onde são definidas entidades elementares de configuração, chamados de módulos, apresentando interfaces bem definidas (portos de entrada e de saída). Cada módulo pode encapsular várias tarefas.

Em tempo de execução, a noção de módulo é substituída pela noção de tarefa (unidade elementar de concorrência). O contexto de execução destas tarefas é definido através da instanciação de um módulo. Este processo de instanciação de módulo consiste na alocação de uma área de dados estáticos e dinâmicos comum a instância do módulo, e a atribuição de uma área de *stack* para cada uma das tarefas do módulo. Como parâmetro de entrada na instância de um módulo, é fornecido um índice correspondente.

A cooperação entre tarefas é realizada unicamente por troca de mensagens. Tarefas comunicam-se por transmissões síncronas e assíncronas através de seus portos. As mensagens são enviadas por portos de entrada e recebida por portos de saída. Os portos de saída são conectados aos portos de entrada em tempo de configuração ou execução. Nesta conexão, os portos são endereçados através do índice do porto (um endereço local ao módulo) e do índice da instância do módulo (endereço global), esta forma de endereçamento do porto permite a identificação direta de tarefas de mesmo tipo, que é um dos requisitos para o NTR servir de suporte ao modelo de implementação.

A seguir serão apresentados os serviços básicos fornecidos pelo NTR proposto em [Nacamura 88]:

- Gerenciamento de tarefas: criar/destruir/suspender tarefas. O escalonamento segue a política de pré-esvaziamento (*pre-empty*) por prioridade;
- Suporte para o gerenciamento de configuração estática e dinâmica, envolvendo operações sobre módulos, instâncias e estabelecimento de canais de comunicação;
- Comunicação e sincronismo entre tarefas. O serviço de comunicação entre tarefas é realizado de maneira transparente sobre a rede através de tarefas servidoras de rede.
- tratamento de interrupção;
- gerenciamento de memória.

5.3.2 Características de Implementação

A figura 5.1 mostra a estrutura do software de comunicação implementado que mapeia diretamente o modelo de implementação proposto.

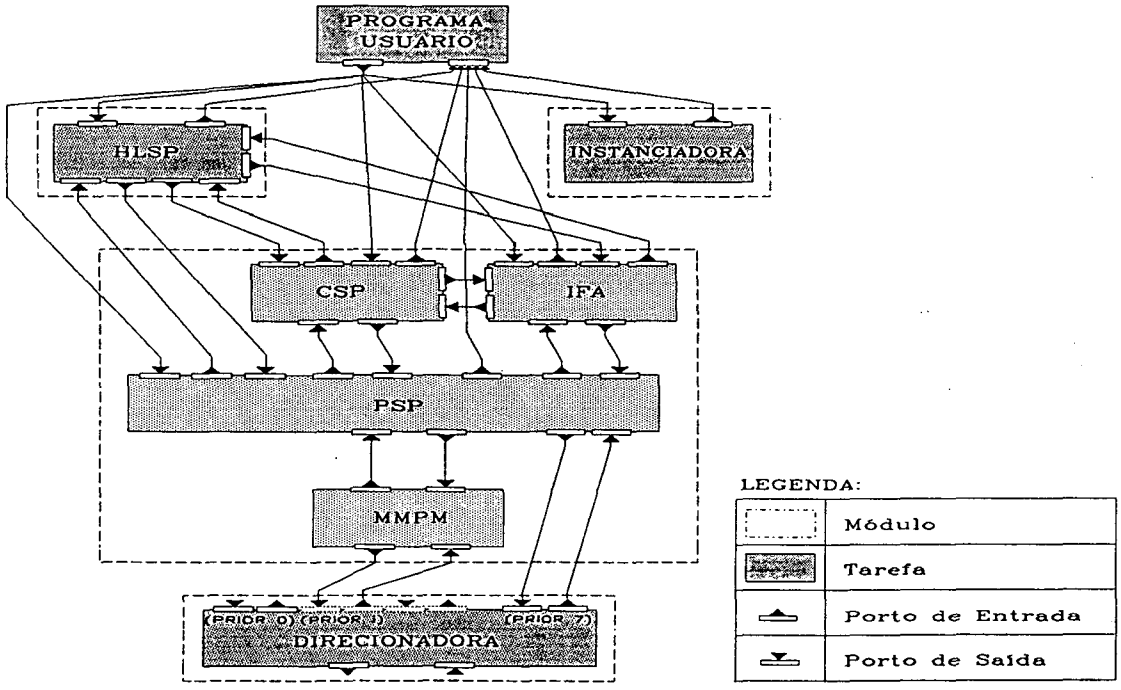


Figura 5.1 Software de Comunicação Implementado

A seguir, serão descritos algumas características desta implementação:

A) Procedimento de Iniciação

A primeira operação executada por uma estação é o procedimento de iniciação. Este procedimento apresenta duas funções básicas, o registro de informações gerais e a iniciação do ambiente multi-tarefas.

As informações registradas são as associadas ao Programa Usuário, informações de gerência de AE e de diretório.

As informações do Programa Usuário estão associados parâmetros de controle de cada tarefa usuária, tais como:

- índice da instância do módulo que contém a tarefa usuária;
- registro do nomes de eventos locais, onde serão armazenados os parâmetros *local_event_name*;
- registro dos eventos atendidos;

Os principais parâmetros de gerência de AE são:
- número de invocações de AE por LSAP (ver ítem 3.5.3);
- Tipo da AE, para permitir a verificação de que tipos de parâmetros definidos pelo *Companion Standard* serão utilizados por esta AE (ver ítem 3.5.2);
- níveis de prioridades das associações a serem requeridas.
- índice das instâncias iniciais dos módulos Específicos de AE e de AE-I;

Todas estas informações estarão contidas em variáveis globais.

O procedimento de iniciação também é responsável pelas instanciações dos módulos do Programa Usuário, Instanciador e dos módulos Direcionadores.

B) Interação Programa Usuário - Provedores da MMSI

No modelo proposto, a biblioteca de funções constitui-se de um conjunto de funções que poderão ser chamadas por qualquer tarefa do Programa Usuário. Cada função terá um procedimento de verificação dos parâmetros, outros processamentos necessários e o encaminhamento do pedido ao provedor destino apropriado. Para este encaminhamento, todas as funções da biblioteca chamam uma função de envio de mensagem. Como pode-se notar, na interação entre o Programa Usuário e os provedores de serviços, o sistema de comunicação inter-processos fornecido pelo núcleo será encapsulado pelas funções da biblioteca MMSI.

A resposta da chamada de função, como especificado na norma MAP, é realizada após a chamada da função WAIT, implícita ou explicitamente.

A seguir, o envio do pedido e a sua recepção serão descritos:

B.1 Envio do Pedido

Para pedir serviços da MMSI, uma tarefa usuária chama as funções da biblioteca padronizada. As principais operações realizadas quando da chamada destas funções são:
- verificação dos parâmetros de entrada, conforme checagem de erro especificada na norma MAP. Caso seja detectado algum erro, será retornado o erro ocorrido. Os valores destes erros, nesta implementação, são aqueles definidos no guia de implementação CNMA [CNMA 89];

- caso os valores dos DCBs de entrada e entrada/saída são definidos como valores *default*, é realizado o pedido de DYNAMIC INITIALIZE DCB; esta função aloca memória e atribui os valores *default* do tipo do DCB, nesta implementação, os valores *default* são os especificados no guia de implementação CNMA;
- iniciação dos apontadores definidos no DCB de entrada/saída;
- outras operações específicas do tipo de função da MMSI;
- obtenção de um nome de evento global ao sistema (*global_event_name*), constituído do *local_event_name* e um apontador para as informações da tarefa usuária (iniciadas durante o procedimento de iniciação);
- construção da mensagem a ser enviada ao provedor, contendo principalmente o nome da função, apontadores para os DCBs e para a área de dados dos parâmetros de saída, além dos parâmetros de entrada expostos;
- chamada da função de envio de mensagem;
- caso o pedido seja síncrono, é chamado a função WAIT.

A função de envio de mensagem é responsável pela conexão de um porto de saída da tarefa usuária que chamou a função, ao porto apropriado de entrada da tarefa provedora de serviço que executará inicialmente o pedido. Após isto, ela executa o envio da mensagem sobre o porto de saída. Esta função também é responsável pela criação de dois portos da tarefa usuária, um de entrada e outro de saída, dedicados a comunicação com a MMSI, isto ocorre no primeiro pedido de envio por parte de uma tarefa usuária.

Para a redução de cópias de mensagens e o consequente aumento de eficiência, ao invés da mensagem propriamente dita, são trocados apenas os apontadores para *buffers* compartilhados. Este procedimento é tomado sempre que apropriado.

B.2 Função WAIT

Nesta implementação, a chamada da função WAIT terá como sinalização da ocorrência de um evento a chegada de uma mensagem em um porto de entrada durante o intervalo de tempo definido na chamada da função WAIT. Para prover seus serviços, esta função atua sobre parâmetros da tarefa usuária, descritos no procedimento de iniciação.

C) Provedores de Serviços da Interface

Os módulos "Específicos AE" e "Específico de AE-I" existentes no modelo proposto são definidos como módulos suportados pelo núcleo, proporcionando a identificação direta das tarefas pertencentes aos módulos do modelo, através do índice da instância do módulo.

D) Tarefa Instanciadora

Nesta implementação, a tarefa Instanciadora atua como provedor de serviços que envolvam instanciações/destruições de instâncias dos módulos "Específico de AE" e "Específico de AE-I", bem como as ligações de portos de módulo. O parâmetro *ae_label* tem como valor o índice da instância do módulo "Específico de AE-I".

O núcleo adotado apresenta características que simplificam a implementação dos conceitos definidos no modelo de implementação, principalmente na definição do parâmetro *ae_label*, pois a entidade elementar de configuração (módulo) pode englobar um conjunto de tarefas. Em núcleos que não suportam esta característica, é necessário a utilização de outras formas de identificação de um grupo de tarefas (uso de tabelas, por exemplo);

E) Direcionamento de Mensagens

Nesta implementação, a tarefa Direcionadora apresenta oito pares de portos de entrada e saída, associados aos níveis de prioridade das mensagens. Suas operações são aquelas definidas no módulo Estelle "Direcionador", apresentado no capítulo anterior.

A tarefa Direcionadora também é responsável pela simulação da camada de enlace, tendo por objetivo prover, à máquina de protocolo, os serviços de enlace, que serão mapeados em trocas de mensagem remota provido pelo núcleo.

5.4 Problemas Encontrados e Soluções Adotadas

Durante a fase de codificação, foram detectados alguns problemas relacionados aos parâmetros especificados pelos *Companions Standards* durante o estabelecimento do ambiente MMS, além da ausência de alguns parâmetros MMS nas chamadas de função da MMSI. Estes problemas e soluções adotadas serão apresentados a seguir.

5.4.1 Problemas na Extensão do Serviço Initiate

Os parâmetro opcionais *MMS Init Request Detail* e *MMS Init Response Detail* do serviço *Initiate*, contém parâmetros relacionados com a comunicação segundo o contexto de apresentação derivado da sintaxe abstrata definida na norma MMS (ISO 9506). Através destes são negociados o número da versão (*Version Number*), CBB (*Parameter CBB*) e serviços suportados (*Service Supported)* que são especificados pelo documento ISO 9506 ([ISO 88]). Nenhum dos parâmetros do *MMS Init Detail* refere-se a sintaxe abstrata definida nos *Companion Standards*.

Os parâmetros opcionais *list of cs request detail* e *list of cs response detail*, também do serviço *Initiate*, contém parâmetros adicionais a serem negociados no estabelecimento de uma associação e que serão fornecidos pelos *Companions Standards*. O documento ISO 9506 recomenda que estes parâmetros contenham campos similares aqueles definidos em *MMS Init Request Detail* e *MMS Init Response Detail* e outros campos adicionais (definidos pelo CS). Se uma aplicação suporta um serviço definido na sintaxe abstrata do padrão ISO 9506 e na sintaxe abstrata definida no *Companion Standard*, então um bit relacionado será setado no parâmetro *services suported calling* no *MMS Init Request Detail* e no parâmetro apropriado no *list of cs req detail*.

Nos *Companions Standards* analisados ([EIA 89], [ISO 89a], [ISO 89b] e [ISA 90]), os parâmetros *list of cs req/rsp detail* são mudados para *Init Request/Response Detail*, o que pode ser origem de confusão.

O *Companion Standard* NC [ISO 89a] associa ao *Init Request/Response Detail* os seguintes parâmetros:

- *Proposed/Negotiated NCCS Version Number*;
- *Proposed/Negotiated NCCS Application Class*;
- *Proposed/Negotiated Parameter CBB*;
- *Services Supported Calling/Called*.

Os dois últimos parâmetros são idênticos aos definidos em ISO 9506. Entretanto faltou incluir nestes parâmetros os novos serviços e CBBs introduzidos neste *Companion Standard*. Este fato cria um problema na negociação destes.

O *Companion Standard* associado ao CP [IEC 89] apresenta o mesmo problema que o *Companion Standard* do NC.

O *Companion Standard* associado a Controle de Processos [ISA 90] extende os parâmetros *Init Request/Response Detail* com os seguintes parâmetros:

- *Proposed/Negociated Version Number*;
- *Proposed/Negociated Parameter CBB*;
- *Services Supported Calling/Called*;
- *Additional Services Supported Calling/Called*;
- *Additional Cbb Supported Calling/Called.*

Os dois últimos parâmetros inclue os serviços e parâmetros CBB introduzidos por este *Companion Standard*. Esta expansão está de acordo com a ISO 9506.

O *Companion Standard* associado ao Robô [ISO 89b] faz a expansão dos parâmetros *Init Request/Response Detail* da mesma forma que a realizada pelo *Companion Standard* de Controle de Processos, portanto também está de acordo com a ISO 9506.

Este *Companion Standard* não extensão de parâmetros CBBs, portanto, como o caso anterior, esta extensão está de acordo com a ISO 9506.

A solução adotada nesta implementação, no caso dos *Companions Standards* do CP e NC que apresentam erro, foi a inclusão, dos novos serviços e parâmetros CBBs em parâmetros adicionais, de forma idêntica aos *Companions Standards* de Controle de Processos e Robô.

5.4.2 Ausência de Parâmetros nas Funções da MMSI

Outra deficiência encontrada, desta vez na especificação da MMSI, é a ausência do parâmetro *list_of_rsp_detail* nos *inout_dcb* das funções *mm_connect* e *mm_listen* e no *in_dcb* da função *answer*. Nesta implementação, este parâmetro foi incluído nestes DCBs.

5.5 Conclusão

Neste capítulo foram apresentados alguns aspectos da implementação do padrão MMS baseado no modelo de implementação proposto.

Inicialmente foram apresentadas as principais características necessárias a um núcleo de tempo real, para servir de suporte a uma implementação segundo o modelo proposto.

Afim de testar o modelo proposto como metodologia para uma implementação real, foi realizada uma implementação simplificada, utilizando como suporte o núcleo de tempo real proposto em [Nacamura 88] que tem as características levantadas anteriormente.

Apesar do número limitado de funções, foram testados os três tipos de funções existentes na interface (alto nível, confirmada, não confirmada). Após a realização de testes de comunicação na implementação realizada, verificou-se que o modelo proposto mantém as propriedades que caracterizam o comportamento especificado na norma MAP.

Esta implementação permitiu também a elucidação de alguns detalhes da uma implementação real, que faltavam para, junto com o modelo de implementação, formar uma proposta para uma implementação do padrão MMS.

CAPÍTULO 6

CONCLUSÃO

Neste trabalho, foi proposto um modelo de implementação do padrão MMS numa arquitetura Mini-MAP, segundo o padrão da MMSI. Foram ainda apresentadas decisões relativas a implementações da MMSI nesta arquitetura.

A validação do modelo e das decisões de implementação propostas foi realizada a partir da simulação de uma especificação Estelle. O teste do modelo proposto numa implementação real (apesar de simples) permitiu discutir algumas particularidades da implementação e definir as características de um Núcleo de Tempo Real como suporte para a implementação. Dos resultados obtidos, conclui-se na adequação do modelo proposto, garantindo a correção das futuras implementações do MMS numa arquitetura Mini-MAP.

No que diz respeito as contribuições deste trabalho, destaca-se, além do modelo proposto que foi validado e testado, os seguintes pontos:

1. No que concerne a implementação da MMSI numa arquitetura Mini-MAP:

   * foi decidido que cada bloco funcional do Provedor de Serviços seja implementado como uma tarefa (ítem 4.2.2);
   * foram definidas as operações que permitem a ativação/destruição das invocações de AE (ítem 4.2.3);
   * foi definido uma semântica para os parâmetros *ae_label* e *connection_id* utilizados na identificação das AE-I e associações (ítem 4.2.3);
   * foram definidos os valores máximos dos parâmetros de gerenciamento de AE (ítem 3.5.2);
   * foram propostas funcionalidades complementares aos provedores de serviços para codificação/decodificação ASN.1 e para obter o nível de prioridade.

* foi definido um direcionamento das informações que chegam nos LSAP (ítem 4.2.4).

2. Foi detectado o problema da não negociação do parâmetro *context_name* (associado no MAP ao serviço ACES *A_Associate*) quando se utiliza os serviços MMS numa arquitetura Mini-MAP. A utilização de AE-Tipos que suportam a sintaxe abstrata do MMS ou de um *Companion Standard* é a solução proposta. (ítem 3.5.2)

3. Foram constatadas incoerências para dois parâmetros (*list of cs req detail* e *list of cs req detail*) do serviço MMS *Initiate* entre o MMS e os *Companions Standards* do PC e do Robô. (ítem 5.4.1).

Como perspectivas de continuação deste trabalho, apontamos para uma validação completa do modelo de implementação e para a sua posterior implementação numa aplicação real.

7. BIBLIOGRAFIA

[Aguiar 89] Aguiar, M.W.C. "Análise Comparativa de Desempenho da Camada de Enlace de Dados do PROFIBUS e do FIP, Propostas Candidatas ao Padrão FIELD-BUS". Dissertação submetida à Universidade Federal de Santa Catarina para a obtenção do grau de mestre em engenharia elétrica. Florianópolis, 1989.

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

[CNMA 89] Communication Network for Manufacturing Applications - CNMA Implementation Guide 4.0. ESPRIT Project 2617. September, 1989.

[Corwin 90] Corwin, W.M et all."Overview od the IEEE P1003.4 Realtime Extension to POSIX". Real-Time Systems Newsletter, Vol 6, N.1, pág 9-18. 1990.

[Courtiat 88] Courtiat, J-P. "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.

[Diaz 89] Diaz, M. & Vissers, CH. "SEDOS: Designing Open Distributed Software", IEEE Software, Vol. 6, nº 6, November 1989.

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

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

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

[ISO 87a] International Organization For Standardization. "A Reference Model for Discrete Parts Manufacturing", technical report ISO TC184/SC5/WG1-N58.

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

[ISO 87c] 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 90a] ISO 9506/3. "Robots for Manufacturing Environment - Robot Specific Message System (Robot Companion Standard for MMS)". 1989.

[ISO 90b] ISO 9506/4. "Numerical Control of Machines - Numerical Control Message Specification (Companion Standard to ISO?IEC 95061+2)". 1989.

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

[Lepikson 90] Lepikson,H.A. "Padronização e Interação das Unidades de Fabricação, Inspeção e Manipulação de um Célula Flexível de Manufatura". Dissertação submetida à Universidade Federal de Santa Catarina para a obtenção do grau de mestre em engenharia elétrica. Florianópolis, 1990.

[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 N^{O} 3. May 1988.

[McLean 86] McLean, C.R. & Brown, P.F. "The Automated Manufacturing Research Facility at the National Boureau of Standards". IFIP W.G. 5.7 Working Conference on New Technologies for Production Management Systems. 1986.

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

[ML 85] "The ML Handbook", INRIA ("FORMEL" project), November 1985.

[Nacamura 88] Nacamura Júnior, L. "Projeto e Implementação de um Núcleo de Sistema Operacional Distribuido com Mecanismos para Tempo Real". Dissertação submetida à Universidade Federal de Santa Catarina para obtenção de grau de mestre em engenharia elétrica. 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.

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

[Thacker 88] Thacker, B. "The Computer Integrated Organization: Some Business and Technical Issues for the OSI MAP/TOP Solution". Universal Computer Applications. 1988.

[Williams 90] Williams, T.J. "A Reference Model for Computer Integrated Manufacturing from the Viewpoint of Industrial Automantion". 11th IFAC World Congress. Vol 10, p. 1-11. August, 1990.

APÊNDICE - A

ESPECIFICAÇÃO DO MODELO DE IMPLEMENTAÇAO EM PSEUDO-CÓDIGO

Este apêndice contém um pseudo-código da especificação Estelle do Processo de Aplicação (AP) segundo o modelo de implementação proposto. A apresentação de uma pseudo-linguagem, em vez da especificação Estelle gerada, tem por objetivo fornecer um passo intermediário da especificação Estelle, facilitando o entendimento do comportamento proposto, em vez de se ater a detalhes que neste trabalho são desnecessários.

```
SPECIFICATION Processo de Aplicação

    MODULE Programa_Usuário ACTIVITY;
    END;

    MODULE Sistema_de_Comunicação ACTIVITY (info_AEs);
    END;


    BODY Corpo_Prog_Usuário FOR Programa_Usuário;

        (*---- Ações especificadas pelo usuário da MMSI ----*)
    END;

    BODY Corpo_Sist_Comunicação FOR Sistema_de_Comunicação;

        MODULE Direcionador ACTIVITY:
        END;

        MODULE Específico_de_AE ACTIVITY (ae_id);
        END;

        MODULE Específico_de_AE-I ACTIVITY (slsap);
        END;

        BODY Corpo_Direc FOR Direcionador;

            INITIALIZE:
                TO livre
                BEGIN
                    Setar todas as prioridades como inativas;
                    Setar todas as AE-Is como sem escuta;
                END;

            TRANS
            FROM livre TO SAME
            WHEN listen_req FROM algum Psp
                BEGIN
                    Setar AE-I identificada por ae_label como em escuta;
                END;

            TRANS
            FROM livre TO SAME
            WHEN l_data_ack_req FROM algum MMPM
                BEGIN
                    Setar prioridade da mensagem como ativa;
                    OUTPUT l_data_ack_req TO Lsap.
                END;

            TRANS
            FROM livre TO SAME
            WHEN l_data_ack_ind FROM Lsap
            PROVIDED(prioridade da mensagem estiver ativa)
                BEGIN
                    OUTPUT l_data_ack_ind TO AE-I;
                END;
```

```
    TRANS
    FROM livre TO ligando
    WHEN l_data_ack_ind FROM Lsap
    PROVIDED((prioridade da mensagem não estiver ativa) AND (serviço for initiate) AND
            (existe AE-I em escuta))
        BEGIN
            Obter primeiro ae_label em escuta;
            Setar prioridade da mensagem como ativa;
            Setar parâmetros de l_data_ack_ind;
            OUTPUT ped_ligação TO Sistema_de_Comunicação.
        END;

    TRANS
    FROM ligando TO livre
    WHEN resp_assoc FROM Sistema_de_Comunicação
        BEGIN
            OUTPUT l_data_ack_ind registrada TO Mmpm;
        END;

    TRANS
    FROM livre TO SAME
    WHEN l_data_ack_ind FROM Lsap
    PROVIDED (prioridade da mensagem não estiver ativa) AND (serviço não for initiate com AE-
        I em escuta))
        BEGIN
            Se lsdu não for nula então;
                Obter PDU de erro;
                OUTPUT l_data_ack_req TO Lsap;
        END;

    TRANS
    FROM livre TO SAME
    WHEN l_data_ack_status_ind FROM Lsap
        BEGIN
            SE a prioridade estiver ativa;
                OUTPUT l_data_ack_status_ind TO Mmpm;
        END;

END;

BODY Corpo_Esp_AE FOR Específico_de_AE;

    INITIALIZE
        BEGIN
            Setar identificador deste módulo, que foi recebido na iniciação.
        END;

    TRANS
    WHEN mm_connect_req FROM Sistema_de_Comunicação
        BEGIN
            OUTPUT mm_connect_req TO Csp do módulo Específico_ de_AE-I identificado
                por ae_label;
        END;
```

TRANS
WHEN mm_connect_rsp FROM algum Csp
BEGIN
OUTPUT mm_connect_rsp TO Programa_Usuário;
END;

TRANS
WHEN mm_listen_req FROM Sistema_de_Comunicação
BEGIN
OUTPUT listen_req TO Psp identificado por *ae_label*;
Registrar *local_event_name*;
END;

TRANS
WHEN listen_rsp FROM algum Psp
PROVIDED(SE *serviço* igual a initiate)
BEGIN
OUTPUT mm_listen_rsp com *connection_id* igual ao identificador da instância do módulo Específico_de_AE-I do Psp que enviou a resposta e *local_event_name* registrado TO Psp identificado por *ae_label*
END;

TRANS
WHEN mm_answer_req FROM Sistema_de_Comunicação
BEGIN
OUTPUT primitiva_rsp com *serviço* igual a initiate e com *result* igual ao dado pela entrada TO Psp identificado por *ae_label*;
Registrar *local_event_name*.
END;

TRANS
WHEN primitiva_ind FROM algum Psp
PROVIDED(*serviço* igual aMessageStatus)
BEGIN
OUTPUT mm_answer_rsp com *result* igual ao *link_ status* e dado pela entrada e *local_event_name* igual ao registrado AO Programa_Usuário;
END;

END;

BODY Corpo_Esp_AE-I FOR Específico_de_AE-I;

MODULE Csp ACTIVITY (slsap);
END;

MODULE Psp ACTIVITY;
END;

MODULE Ifa ACTIVITY;
END;

MODULE Mmpm ACTIVITY;
END;

```
BODY Corpo_Csp FOR Csp;
    INITIALIZE
        BEGIN
            Setar lsap ativo;
        END;

    TRANS
    WHEN mm_connect_req FROM Hlsp
        BEGIN
            Setar nome de evento de conexão com local_ event_name;
            Montar primitiva_req;
            Setar identificador da AEi com connection_id;
            OUTPUT primitiva_req TO Psp;
        END;

    TRANS
    WHEN primitiva_cnf FROM Psp
    PROVIDED (serviço igual a initiate)
        BEGIN
            OUTPUT mm_connect_rsp com local_event_name e connection_id registrados
                TO Psp;
        END;

    TRANS
    WHEN mm_xxxx_req FROM Programa_Usuário
        BEGIN
            Obter invokeID e seta-lo como em uso;
            Setar nome de evento de invokeID a local_event_ name;
            Montar primitiva_req;
            OUTPUT primitiva_req TO Psp;
        END;

    TRANS
    WHEN primitiva_cnf FROM Psp
    PROVIDED(serviço for diferente de initiate)
        BEGIN
            Setar invokeID como livre;
            MONTAR mm_xxxx_rsp, com parâmetros e local_ event_name registrado;
            OUTPUT mm_xxxx_rsp TO Programa_Usuário;
        END;
END;

BODY Corpo_Ifa FOR Ifa;

    INITIALIZE
        TO Aguardando;
        BEGIN
        END;

    TRANS
    FROM Aguardando TO Espera_ind
    WHEN mm_ireceive_req FROM Sistema_de_Comunicação
        BEGIN
            REGISTRAR local_event_name;
        END;
```

```
    TRANS
    FROM Espera_ind TO Aguardando
    WHEN primitiva_ind FROM Psp
        BEGIN
            Obter mm_ireceive_rsp com local_event_name registrado;
            OUTPUT mm_ireceive_rsp TO Sistema_de_Comunicação;
        END;
END;

BODY Corpo_Psp FOR Psp

INITIALIZE
    TO Não_conectada
    BEGIN
    END;

    TRANS
    FROM Não_conectada TO Em_conexão
    WHEN primitiva_req FROM Csp
    PROVIDED(serviço for initiate)
        BEGIN
            MONTAR PDU;
            OUTPUT PDU TO Mmpm;
        END;

    TRANS
    FROM Em_conexão TO Conectada
    WHEN PDU FROM Mmpm
    PROVIDED(tipo_PDU for Response_PDU) AND (serviço for initiate)
        BEGIN
            Obter primitiva_cnf a partir de PDU;
            OUTPUT primitiva_cnf TO Csp;
        END;

    TRANS
    FROM Em_conexão TO Não_conectada
    WHEN PDU FROM Mmpm
    PROVIDED((tipo_PDU for Error_PDU ) OR (tipo_PDU for Reject_PDU))
        BEGIN
            Obter primitiva_cnf a partir de PDU;
            OUTPUT primitiva_cnf TO Csp;
        END;

    TRANS
    FROM Não_conectada TO Em_conexão
    WHEN listen_req FROM Hlsp
        BEGIN
            Setar connection_id como ae_label;
            OUTPUT listen_req TO Direcionador;
        END;
```

```
TRANS
FROM Em_conexão TO Resp_con
WHEN PDU DE Mmpm
PROVIDED((tipo_PDU for Request_PDU) AND (serviço for initiate))
    BEGIN
        Obter primitiva_ind a partir de PDU;
        OUTPUT primitiva_ind TO Csp;
    END;

TRANS
FROM Resp_con TO Ag_Ack_co
WHEN primitiva_rsp FROM Hlsp
PROVIDED((serviço for initiate) AND (resultado for sucesso))
    BEGIN
        MONTAR PDU do tipo Response_PDU;
        OUTPUT PDU TO Mmpm;
    END;

TRANS
FROM Conectada TO SAME
WHEN primitiva_rsp FROM Csp
PROVIDED(serviço não for initiate)
    BEGIN
        MONTAR PDU do tipo Response_PDU;
        OUTPUT PDU TO Mmpm;
    END;

TRANS
FROM Conectada TO SAME
WHEN PDU FROM Mmpm
PROVIDED((tipo_PDU não for Request_PDU) AND (serviço não for initiate))
    BEGIN1
        Obter primitiva_cnf a partir de PDU;
        OUTPUT primitiva_cnf TO Csp;
    END;

TRANS
FROM Conectada TO SAME
WHEN PDU FROM Mmpm
PROVIDED((tipo_PDU for Request_PDU) AND (serviço não for MessageStatus))
    BEGIN
        Obter primitiva_ind a partir de PDU;
        OUTPUT primitiva_ind TO Ifa;
    END;

TRANS
FROM Conectada TO Ag_Ack_rsp
WHEN mm_xxresponse_req FROM Sistema_de_Comunicação
    BEGIN
        Montar PDU;
        Setar invokeID como serviço pendente;
        OUTPUT PDU TO Mmpm;
    END;
```

```
    TRANS
    FROM Ag_Ack_rsp TO Conectada
    WHEN PDU FROM Mmpm
    PROVIDED (serviço for MessageStatus)
        BEGIN
            Obter mm_xxresponse_rsp indicando link_status como resultado;
            OUTPUT mm_xxresponse_rsp TO Sistema_de_Comunicação;
        END;

    TRANS
    FROM Ag_Ack_co TO Conectada
    WHEN PDU FROM Mmpm
    PROVIDED ((serviço for MessageStatus) AND (link_ status for OK))
        BEGIN
            Obter primitiva_ind;
            OUTPUT primitiva_ind TO Hlsp;
        END;

    TRANS
    FROM Ag_Ack_co TO Não_conectada
    WHEN PDU FROM Mmpm
    PROVIDED ((serviço for MessageStatus) AND (link_ status não for OK))
        BEGIN
            Obter primitiva_ind;
            OUTPUT primitiva_ind TO Hlsp;
        END;
END;

BODY Corpo_Mmpm FOR Mmpm;

    INITIALIZE
        TO Closed
        BEGIN
            Setar correntemente_associado a falso;
            Setar todas as máquinas de transação a UNEXISTENT
        END;

    TRANS
    FROM Closed TO Wait_null_ack
    WHEN PDU FROM Psp;
    PROVIDED (serviço for initiate)
        BEGIN
            Setar lsdu de initiate com PDU;
            Obter l_data_ack_req com lsdu nula;
            OUTPUT l_data_ack_req TO Direcionador;
        END;

    TRANS
    FROM Wait_null_ack TO Wait_conf_ack
    WHEN l_data_ack_status_ind
    PROVIDED (link_status indicar OK)
        BEGIN
            Obter l_data_ack_req com lsdu de initiate;
            OUTPUT pm_lsap.l_data_ack_req TO Direcionador;
        END;
```

TRANS
FROM Wait_null_ack TO Closed
WHEN l_data_ack_status_ind FROM Direcionador
PROVIDED (*link_status* não indicar OK)
    BEGIN
        Obter PDU indicando erro no initiate;
        OUTPUT PDU TO Psp;
    END;

TRANS
FROM Closed TO SAME
WHEN PDU FROM Psp
PROVIDED ((*tipo_PDU* for Response_PDU) AND (correntemente_associada for verdadeiro))
    BEGIN
        Setar máquina de transação identificada por *invokeID* a WAIT_CONFIRMED_ACK;
        Setar serviço pendente com *invokeID*;
        Obter l_data_ack_req;
        OUTPUT l_data_ack_req TO Direcionador;
    END;

TRANS
FROM Wait_conf_ack TO Wait_conf_rsp
WHEN l_data_ack_status_ind FROM Direcionador
PROVIDED(*link_status* indicar OK)
    BEGIN
    END;

TRANS
FROM Closed TO SAME
WHEN l_data_ack_status_ind FROM Direcionador
PROVIDED((máquina de transação do serviço pendente estiver no estado WAIT_CONFIRMED_ACK) and (*link_ status* indicar OK))
    BEGIN
        Setar máquina de transação do serviço pendente a WAIT_CONFIRMED_RESP.
    END;

TRANS
FROM Wait_conf_ack TO Closed
WHEN l_data_ack_status_ind FROM Direcionador
PROVIDED(*link_status* não indicar OK)
    BEGIN
        Obter PDU indicando erro no serviço initiate;
        OUTPUT PDU TO Psp;
    END;

TRANS
FROM Closed TO SAME
WHEN l_data_ack_status_ind FROM Direcionador
PROVIDED((máquina de transação do serviço pendente estiver no estado WAIT_CONFIRMED_ACK) AND (*link_status* não indicar OK))
BEGIN
Obter PDU indicando erro no serviço pendente;
Setar máquina de transação do serviço pendente a UNEXISTENT;
OUTPUT PDU TO Psp
END;

TRANS
FROM Wait_conf_rsp TO Closed
WHEN l_data_ack_ind FROM Direcionador
PROVIDED ((*serviço* for initiate) and (*tipo_PDU* for Response_PDU))
BEGIN
Obter PDU;
Setar correntemente associado a verdadeiro;
OUTPUT PDU TO Psp;
END;

TRANS
FROM Closed TO SAME
WHEN l_data_ack_ind FROM Direcionador
PROVIDED((máquina de transação identificada por *invokeID* estiver no estado WAIT_CONFIRMED_RESP) AND (*tipo_PDU* for Response_PDU))
BEGIN
Obter PDU;
OUTPUT PDU TO Psp;
END;

TRANS
FROM Wait_conf_rsp TO Closed
WHEN l_data_ack_ind FROM Direcionador
PROVIDED ((*tipo_PDU* for Error_PDU) or (*tipo_PDU* for Reject_PDU))
BEGIN
Obter PDU indicando o erro;
OUTPUT PDU TO Psp;;
END;

TRANS
FROM Closed TO SAME
WHEN l_data_ack_ind FROM Direcionador
PROVIDED((máquina de transação identificada por *invokeID* estiver no estado WAIT_CONFIRMED_RESP) AND ((*tipo_PDU* for Error_PDU) OR (*tipo_PDU* for)))
BEGIN
Obter *PDU* indicando o erro;
Setar máquina de transação identificada por *invokeID* a UNEXISTENT;
OUTPUT PDU TO Psp;
END;

```
TRANS
FROM Closed TO NR
WHEN l_data_ack_ind FROM Direcionador
PROVIDED((serviço for initiate) AND (tipo_PDU for Request_PDU) AND
    (correntemente associado indicar falso))
    BEGIN
        Obter PDU;
        OUTPUT PDU TO Psp;
    END;

TRANS
FROM Closed TO SAME
WHEN l_data_ack_ind FROM Direcionador
PROVIDED((serviço não for initiate) AND (tipo_PDU for Request_PDU) AND
    (correntemente associado for verdadeiro))
    BEGIN
        Setar máquina de transação identificada por invokeID estiver no estado
            NEED_RESP;
        Obter PDU;
        OUTPUT PDU TO Psp;
    END;

TRANS
FROM Wait_resp_ack TO Closed
WHEN l_data_ack_status_ind FROM Direcionador
    BEGIN
        Obter PDU de MessageStatus indicando link_status;
        Setar correntemente_associado a verdadeiro;
        OUTPUT PDU TO Psp;
    END;

TRANS
FROM Closed TO SAME
WHEN l_data_ack_status_ind FROM Direcionador
PROVIDED((máquina de transação identificada pelo serviço pendente estiver no estado
    WAIT_CONFIRMED_RESP) AND (correntemente_associado for verdadeiro))
    BEGIN
        Obter PDU de MessageStatus indicando link_status;
        Setar máquina de transação identificada por invokeID a UNEXISTENT;
        OUTPUT PDU TO Psp;
    END;

TRANS
FROM Need_resp TO Wait_resp_ack
WHEN PDU FROM Psp
PROVIDED ((serviço for initiate ) AND ((tipo_PDU for Response_PDU) OR (tipo_PDU
    for Error_PDU)))
    BEGIN
        Obter l_data_ack_req;
        OUTPUT l_data_ack_req TO Direcionador;
    END;
```

```
        TRANS
        FROM Closed TO SAME
        WHEN pm_psp.env_pdu
        PROVIDED ((máquina de transação identificada pelo serviço pendente estiver no estado
            NEED_RESP ) AND ((tipo_PDU_for Response_PDU ) OR (tipo_PDU for
            Error_PDU )) AND (correntemente_associado for verdadeiro))
            BEGIN
                Setar máquina de transação identificada por invokeID a
                    WAIT_RESPONSE_ACK;
                Setar serviço pendente com invokeID;
                Obter l_data_ack_req;
                OUTPUT l_data_ack_req TO Direcionador;
            END;

    END;

    INITIALIZE
        BEGIN
            INIT Csp WITH Corpo_csp (slsap);
            INIT Psp WITH Corpo_psp;
            INIT Ifa WITH Corpo_ifa;
            INIT Mmpm WITH Corpo_mmpm;
        END;
END;

INITIALIZE
    BEGIN
        SETAR informações de diretório a partir de parâmetros de iniciação;
        INIT Direcionador para cada AE registrada WITH Corpo_Direc;
    END;

TRANS
WHEN mm_aeact_req FROM Programa_Usuário
BEGIN
    Obter informações de diretório a partir de my_ae_name;
    Se número de ativações for zero então
        INIT Específico_AE WITH Corpo_Esp_AE;
    INIT Específico_de_AE-I WITH Corpo_Esp_AE-I passando seu lsap;
    Incrementar número de ativações;
    OUTPUT mm_aeact_rsp com ae_label igual ao identificador do módulo Específico_de_AE-I
        TO Direcionador;
END;

TRANS
WHEN mm_conne_req FROM Programa_Usuário
    BEGIN
        Obter informações de diretório a partir de ae_label;
        Obter prioridade da mensagem livre desta AE;
        Setar prioridade da mensagem como em uso;
        Ligar Mmpm do módulo Específico_de_AE-I identificado por ae_label ao Direcionador da
            AE através do ponto de interação associado ao nível de prioridade;
        OUTPUT mm_conne_req TO Hlsp associado a AE;
    END;
```

```
TRANS
WHEN mm_conne_rsp FROM algum Hlsp
    BEGIN
        OUTPUT mm_conne_rsp TO Programa_Usuário;
    END;

TRANS
WHEN mm_xxxx_req FROM Programa_Usuário
PROVIDED(xxxx for um serviço confirmado)
    BEGIN
        OUTPUT mm_xxxx_req TO Csp do módulo Específico_de_AE-I identificado por
            connection_id;
    END;

TRANS
WHEN mm_xxxx_rsp FROM algum Csp
    BEGIN
        OUTPUT mm_xxxx_rsp TO Programa_Usuário;
    END;

TRANS
WHEN mm_listen_req FROM Programa_Usuário
    BEGIN
        OUTPUT mm_listen_req TO Hlsp da AE obtido através de connection_id;
    END;

TRANS
WHEN mm_listen_rsp FROM algum Hlsp
    BEGIN
        OUTPUT mm_listen_rsp TO Programa_Usuário;
    END;

TRANS
WHEN mm_answer_req FROM Programa_Usuário
    BEGIN
        OUTPUT mm_answer_req TO Hlsp da AE obtido através de connection_id;
    END;

TRANS
WHEN mm_answer_rsp FROM algum Hlsp
    BEGIN
        OUTPUT mm_answer_rsp TO Programa_Usuário;
    END;

TRANS
WHEN mm_ireceive_req FROM Programa_Usuário
    BEGIN
        OUTPUT mm_ireceive_req TO Ifa do módulo Específico_ de_AE-I identificado por
            connection_id;
    END;

TRANS
WHEN mm_ireceive_rsp FROM algum Ifa
    BEGIN
        OUTPUT mm_ireceive_rsp TO Programa_Usuário;
    END;
```

```
        TRANS
        WHEN mm_xxresponse_req FROM Programa_Usuário
            BEGIN
                OUTPUT mm_xxresponse_req TO Psp do módulo Específico_ de_AE-I identificado por
                    connection_id;
            END;

        TRANS
        WHEN mm_xxresponse_rsp FROM algum Psp
            BEGIN
                OUTPUT mm_xxtresponse_rsp TO Programa_Usuário;
            END;

        TRANS
        WHEN ped_ligação FROM algum Direcionador
            BEGIN
                Ligar Mmpm de módulo Específico_de_AE-I identificado por inst ao Direcionador da AE
                    através do ponto de interação associado ao nível de prioridade;
                Setar prioridade como em uso;
                OUTPUT rsp_ligação TO Direcionador;
            END;
    END;

    INITIALIZE:
        INIT Programa_Usuário WITH Corpo_Prog_usuário;
        INIT Sistema_de_Comunicação (info_AEs) WITH Corpo_Sist_ comunicação;
END.
```

APENDICE - B

ESPECIFICAÇÃO DO AMBIENTE DE COMUNICAÇÃO EM PSEUDO-CÓDIGO

Este apêndice contém um pseudo-código da especificação do ambiente de comunicação utilizado no exemplo de simulação. Esta pseudo-linguagem apresenta as mesmas característica da apresentada no apêndice A.

APENDICE - B

ESPECIFICAÇÃO DO AMBIENTE DE COMUNICAÇÃO EM PSEUDO-CÓDIGO

Este apêndice contém um pseudo-código da especificação do ambiente de comunicação utilizado no exemplo de simulação. Esta pseudo-linguagem apresenta as mesmas característica da apresentada no apêndice A.

SPECIFICATION Ambiente_para_Simulação

MODULE Programa_Cliente_1 ACTIVITY;
END;

MODULE Programa_Cliente_Servidor_2 ACTIVITY;
END;

MODULE Programa_Servidor_3 ACTIVITY;
END;

MODULE Sistema_de_Comunicação ACTIVITY (Info_AEs);
END;

MODULE Lsap ACTIVITY;
END;

MODULE Enlace ACTIVITY;
END;

BODY Corpo_Cliente1 FOR Programa_Cliente_1;

INITIALIZE
TO Inicial
BEGIN
END;

TRANS
FROM Inicial TO Operando
BEGIN
OUTPUT mm_aeact_req com *my_ae_name* igual ao de AE_C11 e *local_event_name* igual a 1 TO Sistema_de_ Comunicação;
OUTPUT mm_aeact_req com *my_ae_name* igual ao de AE_C12 e *local_event_name* igual a 2 TO Sistema_de_ Comunicação;
END;

TRANS
FROM Operando TO SAME
WHEN mm_aeact_rsp FROM Sistema_de_Comunicação
PROVIDED(*return_code* indicar sucesso)
BEGIN
Se *local_event_name* igual 1 então
OUTPUT mm_conne_req com ae_label retornado de mm_aeact_rsp e remote_ae igual ao AE_CS21 TO Sistema_de_Comunicação;
Se *local_event_name* igual 2
OUTPUT mm_conne_req com ae_label retornado de mm_aeact_rsp e remote_ae igual ao AE_CS31 TO Sistema_de_Comunicação;
END;

TRANS
FROM Operando TO SAME
WHEN mm_conne_rsp FROM Sistema_de_Comunicação
PROVIDED (*return_code* indicar sucesso)
BEGIN
OUTPUT mm_status_req com *connection_id* igual ao retornado TO Sistema_de_Comunicação;
END:

TRANS
FROM Operando TO SAME
WHEN mm_status_rsp FROM Sistema_de_Comunicação
BEGIN
END;
END;

BODY Corpo_ClienteServidor2 FOR Programa_Cliente_Servidor_2;

INITIALIZE:
TO Inicial
BEGIN
END;

TRANS
FROM Inicial TO Operando
BEGIN
OUTPUT mm_aeact_req com *my_ae_name* igual ao de AE_CS211 e *local_event_name* igual a 1 TO Sistema_ de_Comunicação;
OUTPUT mm_aeact_req com *my_ae_name* igual ao de AE_CS22 e *local_event_name* igual a 2 AO Sistema_ de_Comunicação;
END;

TRANS
FROM Operando TO SAME
WHEN mm_aeact_rsp FROM Sistema_de_Comunicação
PROVIDED ((*return_code* indicar sucesso) AND (*local_event_ name* (de WAIT) igual a 1))
BEGIN
OUTPUT mm_listen_req com ae_label retornado de mm_aeact_rsp TO Sistema_de_Comunicação;

TRANS
FROM Operando TO SAME
WHEN mm_listen_rsp FROM Sistema_de_Comunicação
PROVIDED(SE *return_code* indicar sucesso)
BEGIN
OUTPUT mm_answer_req com *connection_id* retornado de mm_listen_rsp TO Sistema_de_Comunicação;
OUTPUT mm_ireceive_req com *connection_id* retornado de mm_listen_rsp TO Sistema_de_Comunicação;
END;

TRANS
FROM Operando TO SAME
WHEN mm_answer_rsp FROM Sistema_de_Comunicação
BEGIN
END;

TRANS
FROM Operando TO SAME
WHEN mm_ireceive_rsp FROM Sistema_de_Comunicação
PROVIDED(*serviço* for status)
BEGIN
OUTPUT mm_stresponse_req TO Sistema_de_Comunicação;
OUTPUT mm_ireceive_req TO Sistema_de_Comunicação;
END;

TRANS
FROM Operando TO SAME
WHEN mm_stresponse_rsp FROM Sistema_de_Comunicação
BEGIN
END;

TRANS
FROM Operando TO SAME
WHEN mm_aeact_rsp FROM Sistema_de_Comunicação
PROVIDED(*local_event_name* igual a 2)
BEGIN
OUTPUT *mm_conne_req* com *ae_label* retornado de mm_ aeact_rsp e *remote_ae* igual ao AE_S31 TO Sistema_ de_Comunicação;
END;

TRANS
FROM Operando TO SAME
WHEN mm_conne_rsp FROM Sistema_de_Comunicação
PROVIDED(*return_code* indicar sucesso)
BEGIN
OUTPUT mm_status_req com *connection_id* igual ao retornado TO Sistema_de_Comunicação.
END;

TRANS
FROM Operando TO SAME
WHEN mm_status_rsp FROM Sistema_de_Comunicação
BEGIN
END;

END;

BODY Corpo_Servidor3 FOR Programa_Servidor_3;

INITIALIZE:
TO Inicial
BEGIN
Número de conexões igual a zero;
END;

TRANS
FROM Inicial TO Operando
BEGIN
OUTPUT mm_aeact_req com *my_ae_name* igual ao de AE_S31 TO Sistema_de_Comunicação;
OUTPUT mm_aeact_req com *my_ae_name* igual ao de AE_S31 TO Sistema_de_Comunicação;
END;

TRANS
FROM Operando TO SAME
WHEN mm_aeact_rsp FROM Sistema_de_Comunicação
PROVIDED(*return_code* indicar sucesso)
BEGIN
OUTPUT mm_listen_req com *ae_label* retornado de mm_ aeact_rsp TO Sistema_de_Comunicação;
END;

TRANS
FROM Operando TO SAME
WHEN mm_listen_rsp FROM Sistema_de_Comunicação
PROVIDED(*return_code* indicar sucesso)
BEGIN
OUTPUT mm_answer_req com *connection_id* retornado de mm_listen_rsp AO Sistema_de_Comunicação;
Incrementar número de conexões;
Associar *connection_id* ao número de conexões;
OUTPUT mm_ireceive_req com *connection_id* retornado de mm_listen_rsp e *local_event_name* igual ao número de conexões AO Sistema_de_Comunicação;
END;

TRANS
FROM Operando TO SAME
WHEN mm_answer_rsp FROM Sistema_de_Comunicação
BEGIN
END;

TRANS
FROM Operando TO SAME
WHEN mm_ireceive_rsp FROM Sistema_de_Comunicação
PROVIDED(*serviço* for status)
BEGIN
OUTPUT mm_stresponse_req com *connection_id* associado ao número da conexão recebido em *local_event_name* TO Sistema_de_Comunicação;
OUTPUT mm_ireceive_req TO Sistema_de_Comunicação;
END;

TRANS
FROM Operando TO SAME
WHEN mm_stresponse_rsp FROM Sistema_de_Comunicação
BEGIN
END;

END;

```
BODY Corpo_Sist_Comunicação FOR Sistema_de_Comunicação

    (* Especificado no apêndice A *)

END;

BODY Corpo_Lsap FOR Lsap

   INITIALIZE
         TO Closed
         BEGIN
         END;

    TRANS
    WHEN l_data_ack_req FROM Enlace.
        BEGIN
            OUTPUT data_ack_status_req com link_status = OK TO Enlace;
            OUTPUT l_data_ack_ind TO Direcionador;
        END;

    TRANS
    WHEN l_data_ack_req FROM Enlace
        BEGIN
            OUTPUT data_ack_status_req com link_status = UN TO Enlace;
            Montar lsdu de MessageStatus_ind com link_status = UN;
            OUTPUT l_data_ack_ind TO Direcionador;
        END;

    TRANS
    WHEN l_data_ack_req FROM Direcionador
        BEGIN
            OUTPUT l_data_ack_req TO Enlace;
        END;

    TRANS
    WHEN l_data_ack_status_req FROM Enlace
        BEGIN
            OUTPUT l_data_ack_status_ind TO Direcionador;
        END;

END;

BODY Corpo_Enlace FOR Enlace;

    INITIALIZE
        BEGIN
        END;

    TRANS
    WHEN l_data_ack_req FROM Lsap
        BEGIN
            OUTPUT l_data_ack_req TO Lsap definido em dlsap;
        END;
```

```
        TRANS
        WHEN l_data_ack_status_req FROM Lsap
            BEGIN
                OUTPUT l_data_ack_status_req TO Lsap definido em dlsap.
            END;
    END;

    INITIALIZE:
            INIT Programa_Cliente 1;
            INIT Sistema_de_Comunicação [1] com parâmetros de gerenciamento de todas as AEs;
            INIT Lsap [1];
            INIT Lsap [2];
            INIT Programa_Cliente_Servidor2;
            INIT Sistema_de_Comunicação [2] com parâmetros de gerenciamento de todas as AEs;
            INIT Lsap [3];
            INIT Lsap [4];
            INIT Programa_Servidor3;
            INIT Sistema_de_Comunicação [3] com parâmetros de gerenciamento de todas as AEs;
            INIT Lsap [5];
            INIT Enlace.

END.
```

APÊNDICE - C

ESTIM

O desenvolvimento do ESTIM [Saqui-Sannes 90] (*Estelle Simulator based on an Interpretative Machine*) é parte do projeto ESPRIT-SEDOS [Diaz 89], como parte do projeto de pesquisa do LAAS-CNRS que investiga a técnica de descrição formal Estelle.

O ESTIM foi escrito na linguagem funcional ML [ML 85], e sua semântica é consistente com Estelle* [Courtiat 89], as diferenças entre Estelle e Estelle* podem ser vistas na figura C.1. Pode-se dizer que ESTIM torna uma especificação Estelle* executável. Ele permite a observação do comportamento dos objetos da especificação, permitindo a validação, pela simulação e pela verificação de uma especificação Estelle*. Uma visão do ambiente ESTIM é visto na figura C.2.

	Estelle ISO	Estelle*
Atributo 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	não permitida disabilitada
Clausula rendezvous	— —	ip?interação ip!interação

C.1 Estelle* vs Estelle [Saqui 90b]

GENESTIM é um tradutor Estelle, posteriormente executa uma analise sintática do arquivo fonte da especificação e uma checagem de tipos estáticos. Como resultado final é produzido uma árvore abstrata ML da especificação, que será entrada para o ambiente ESTIM.

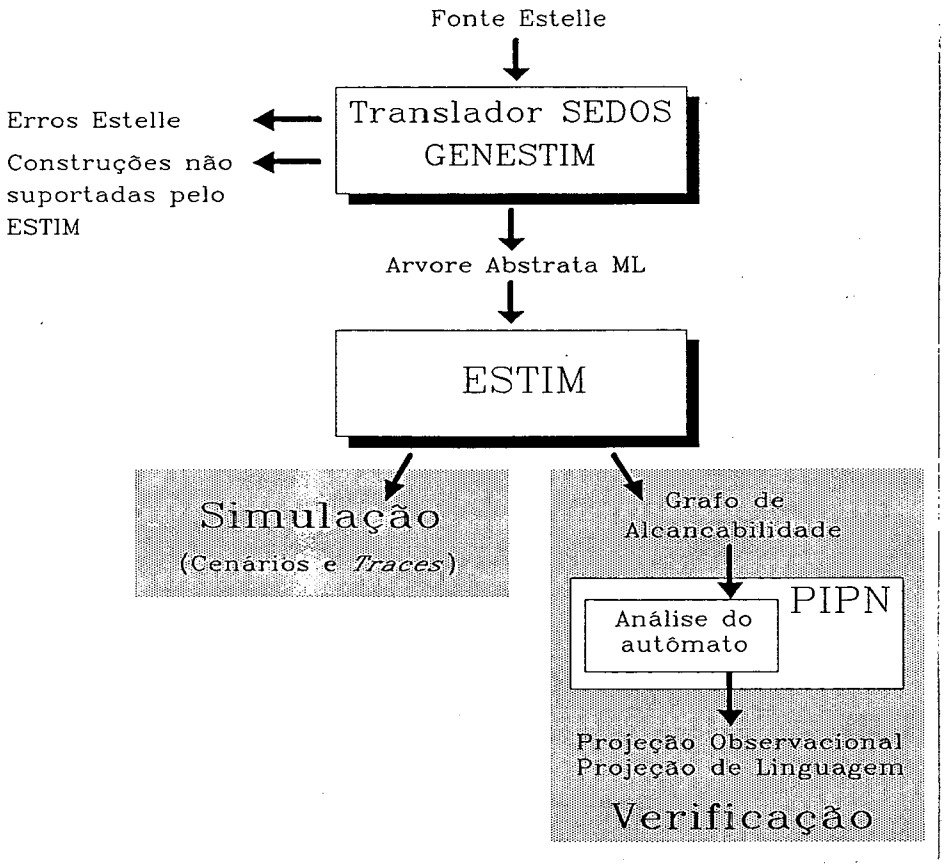


Figura C.2 Ambiente ESTIM [Saqui 90b]

A simulação é uma estratégia de validação que explora alguns caminhos particulares do grafo de alcançabilidade. Para a simulação, o ESTIM opera de maneira interativa com o usuário, fornecendo em cada estado, um conjunto de transições disparáveis. Um conjunto de comandos podem ser inseridos na especificação a fim de dirigir a simulação (por exemplo, *Breakpoints*).

A verificação é outra estratégia de validação, onde é requerido o grafo de alcançabilidade completo. Neste caso, o ESTIM oferece interfaces para outras ferramentas de validação que são o PIPN [Azema 85] e ALDEBARAN [Fernandez 88].