UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA DA COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

ASPECTOS DE IMPLEMENTAÇÃO DA INTERFACE DOS PROGRAMAS DE APLICAÇÃO PARA O PROTOCOLO MMS E SEUS PADRÕES ASSOCIADOS: GERENCIAMENTO DE CONEXÃO E EXEMPLO DE APLICAÇÃO.

por: ENG. JAYME NICOLATO CORRÊA.
orientador: PROF. DR. MANUEL DE JESUS MENDES.

/90

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

Julho de 1990

PARA MÔNICA.

Este trabalho contou com o apoio financeiro da
Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CAPES

AGRADECIMENTOS

Aos meus pais: Jayme e Maria José e à tia Dodora por todo apoio, amizade e amor que nunca me faltaram.

Ao Prof. Dr. Manuel de Jesus Mendes. Não só pela orientação neste trabalho como também pela amizade e confiança em mim depositados.

Ao prof Edmundo Madeira, pela amizade e ajuda sem a qual este trabalho não teria sido concluído.

Ao grupo do Sistema Didático, especialmente à amiga Verônica, pela amizade e pelas incontáveis discussões que fizeram o trabalho evoluir.

Aos amigos Alberto, Rômulo, Luciano, Maurílio, Leonardo, Suhel, João, Waldomiro e Jean pelo companherismo e bons momentos.

À todos os amigos do LCA (e antigo LAB 25).

À Universidade Estadual de Campinas por ter me dado a possibilidade de fazer este trabalho.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES - pelo apoio financeiro.

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

RESUMO

Este trabalho apresenta os aspectos relacionados à implementação dos serviços de Gerenciamento de Conexão da API ("Application Program Interface") do protocolo MMS ("Manufacturing Message Specification"). É parte integrante de um projeto para implementação do Sistema Didático da Aplicação, SISDI-MAP. São apresentados os conceitos relacionados diretamente à API ou ao MMS e seus Padrões Associados. Finalmente, descreve-se um exemplo de utilização deste ambiente, através de um Programa de Aplicação.

ABSTRACT

This work presents the implementation of the connection management procedure performed by the MMS Application Interface. It is part of a project called SISDI-MAP (Application Didactic System). The concepts related to the API or the MMS with its Companions Standards are presented. Finally, it is shown an example of use of the environment by means of an Application Program.

ÍNDICE

1.0 - O PROJETO MAP

Sendo a maior indústria automatizada no mundo, a GM ("General Motors") tinha em 1985 por volta de 40.000 equipamentos programáveis instalados em suas fábricas. Somente 15 percento destes equipamentos estavam aptos para se comunicarem uns com os outros. O resultado desta situação foi a criação de um problema chamado de "ilhas de automação" - computadores e equipamentos programáveis que não estão aptos para se comunicarem além de sua própria célula.

Este isolamento entre os equipamentos contraria os objetivos de uma implementação CIM ("Computer Integrated Manufacturing") onde todos, ou quase todos, os segmentos da manufatura estão interligados. Na realidade, a comunicação é a ferramenta CIM responsável pela melhoria dos custos, da produtividade, da flexibilidade e da qualidade da manufatura. A título de exemplo, a comunicação entre o sistema de CAE ("Computer Aided Engineering") e os controladores das máquinas de ferramentas, permite instantâneas atualizações dos projetos e realimentações do controle de qualidade.

Para a solução deste problema, um grupo de trabalho foi criado pela GM. Através da adoção de padrões internacionais já existentes, a GM procurou criar um protocolo para automação da manufatura que permitisse a comunicação de qualquer tipo de equipamentos programáveis, sem um alto custo de desenvolvimento de hardware e software para cada aplicação.

Historicamente, a ISO ("International Standard Organization") - com o suporte nos EUA através da ANSI ("American National Standards Institute"), da CBEMA ("Computer and Business Equipment Association") e da NBS ("National Bureau of Standards") - começou a trabalhar em meados dos anos 70 em uma série de padrões de comunicação conhecidos como OSI ("Open Systems Interconnection"). Ao mesmo tempo o IEC ("International Eletrotechnical Commission") - com o suporte nos EUA através da ISA ("Instrument Society of America") - começou a trabalhar no padrão PROWAY ("Process Data Highways"). Em 1980, o IEEE ("Institute of Electrical and Electronic Engineers") formou a comissão que iniciou os trabalhos na especificação de padrões para Redes Locais de Computadores (LAN - "Local Area Network"). Estes padrões tornaram-se a base para o protocolo MAP ("Manufacturing Automation Protocol") [09], alcançando o objetivo de um padrão aberto para interconexão, uma vez que nenhuma parte do MAP é de propriedade da GM.

O objetivo do MAP é o de facilitar a interconexão das "ilhas de automação" e computadores, para formar sistemas integrados da manufatura que não levem em consideração a incompatibilidade que existe hoje entre as máquinas, estações de trabalho, controladores e computadores.

Em resposta a uma pressão de mercado, realizada isoladamente por algumas industrias ou através do Grupo de Usuários MAP, muitos fabricantes de equipamentos programáveis estão implementando produtos em conformidade com a especificação MAP [22]. Esta especificação define um conjunto padrão de protocolos que ou foram escolhidos da família de protocolos ISO, OSI compatíveis, ou foram desenvolvidos especificamente para preencher as necessidades da Automação Industrial.

A arquitetura padrão MAP foi escolhida com base no modelo RM-OSI/ISO e contempla as sete camadas. Já nas arquiteturas PROWAY, MINI/EPA-MAP e FIELD-BUS são eliminadas algumas camadas, de forma a serem satisfeitos os aspectos de tempo real.

Dentro do protocolo MAP, o protocolo de aplicação MMS ("Manufacturing Message Specification") [04] [05] - destinado para a troca de mensagens entre equipamentos programáveis da manufatura - é o componente chave. O MAP sem o MMS é um mero sistema de comunicação. Mas os usuários e fabricantes que utilizarem a especificação adequadamente terão uma poderosa ferramenta de CIM à sua disposição. Através do MMS, podem-se modelar os equipamentos como se fossem caixas-pretas para controle ou outras funções específicas do chão de fábrica. Estas caixas pretas irão se diferenciar somente em aspectos específicos de cada equipamento. Como ferramenta para tal, definem-se na especificação, os VMD's ("Virtual Manufacturing Devices") que tratarão de uma forma abstrata dos elementos estruturais e dos objetos que compõem os dispositivos reais.

A especificação do MMS toma a forma de um esqueleto padrão que não faz referência a equipamentos específicos tais como Comandos Numéricos Computadorizados (CNC's), Controladores Lógicos Programáveis (CLP's), etc. Ele é um padrão genérico para qualquer equipamento do chão de fábrica. A relação entre o MMS genérico e os requisitos de um equipamento específico deverão ser fornecidos por associações de industrias através de Padrões Associados ("Companion Standards") [18], [19], [20].

A idéia por trás do padrão OSI é de tornar possível a comunicação de computadores de fabricantes diferentes. Contudo a natureza da interface entre camadas, o gerenciamento das camadas e o gerenciamento global dos recursos são uma questão de implementação local. Este tipo de abordagem leva a uma ênfase em protocolos,

onde a definição de serviços também é fornecida, mas em forma de primitivas abstratas.

Para a definição das camadas intermediárias, a definição abstrata dos serviços não causa nenhum problema, a menos que o usuário, inadivertidamente, decida adquirir cada camada de um fabricante diferente. Para sistemas produzidos por um único fabricante, o usuário não deve se preocupar com as interfaces entre as seis camadas inferiores. A única coisa que interessa é saber como o Programa de Aplicação irá conversar com a Camada de Aplicação. Contudo, o padrão OSI não especifica esta última interface em muitos detalhes, e as especificações dadas não são nem mesmo implementáveis.

Como consequência desta especificação incompleta, cada fabricante terá de especificar sua própria interface entre o Programa de Aplicação e o Provedor de Serviços de Protocolo da Camada de Aplicação. O resultado disto será que os programas escritos utilizando-se a interface A não poderão ser executados em ambientes que suportem a interface B.

O projeto MAP considerou esta situação inaceitável e especificou uma interface concreta para os Programas de Aplicação. O usuário que queira utilizar os serviços do sistema de comunicação, deverá utilizar esta interface específica, chamada API ("Aplication Program Interface") [14], [15], [16], [17], através de chamadas de funções de biblioteca com os parâmetros apropriados. A API irá gerar, em seguida, a primitiva adequada de requisição do serviço de rede correspondente.

1.1. - MOTIVAÇÃO DO TRABALHO

A motivação principal, por trás do desenvolvimento desta dissertação de mestrado, advinha da necessidade de se realizar um estudo detalhado dos documentos MAP relacionados aos Programas de Aplicação (AP) e à Interface de Aplicação (API) para o protocolo MMS. Este estudo detalhado deveria servir como base para a implementação destas entidades no Sistema Didático da Aplicação, SISDI-MAP. No início, o alvo principal era a unidade funcional de gerenciamento de conexão.

Durante o decorrer do trabalho, sentiu-se a necessidade do desenvolvimento de novos estudos para a compreensão dos conceitos relacionados aos objetos abstratos definidos pelo MMS, bem como a utilização de seus serviços pertencentes às demais unidades funcionais, de tal forma que se pudesse implementar um Programa de Aplicação coerente com as necessidades reais de uma aplicação da manufatura. Este trabalho adicional englobou também os Padrões associados e foi um fator de acréscimo da motivação do trabalho.

1.2 - DESENVOLVIMENTO DO TRABALHO

Ao longo do desenvolvimento desta dissertação pretende-se, numa primeira etapa, introduzir as definições e conceitos a ela associados para, numa segunda etapa, mostrar como foi realizado o mapeamento destes conceitos abstratos em uma implementação real.

No capítulo dois, chamado Sistema de Aplicação, introduzem-se os conceitos relacionados ao MMS, aos Padrões Associados e à API. Durante a apresentação do MMS, dá-se uma ênfase maior ao VMD e aos domínios e objetos a ele associados, bem como aos serviços disponíveis para a manipulação destas entidades. Complementando a especificação do MMS, mostram-se a composição básica de um padrão associado e as informações específicas por eles fornecidas. Finalmente, introduzem-se, de uma maneira mais completa, os conceitos funcionalidades e modelos funcionais de uma Interface de Aplicação geral e, depois, de uma Interface de Aplicação específica para o MMS, tendo como escopo principal os serviços de gerenciamento de conexão.

No capítulo três, chamado Aspectos de Implementação, apresentam-se as experiências adquiridas durante a implementação da unidade funcional de gerenciamento de conexão para a Interface de Aplicação do MMS. Mostram-se também os aspectos de implementação da API no SISDI-MAP com a criação de um processo API que gerencia toda a troca de mensagens, alocação de memória, entre outras funcionalidades. Um ponto relevante nesta implementação foi o investimento realizado tanto na parte de estrutura de dados, quanto na estruturação dos algoritmos implementados, para que o sistema não abandonasse a arquitetura conceitual definida pelo Modelo de Referência OSI [01], figura 1.0.

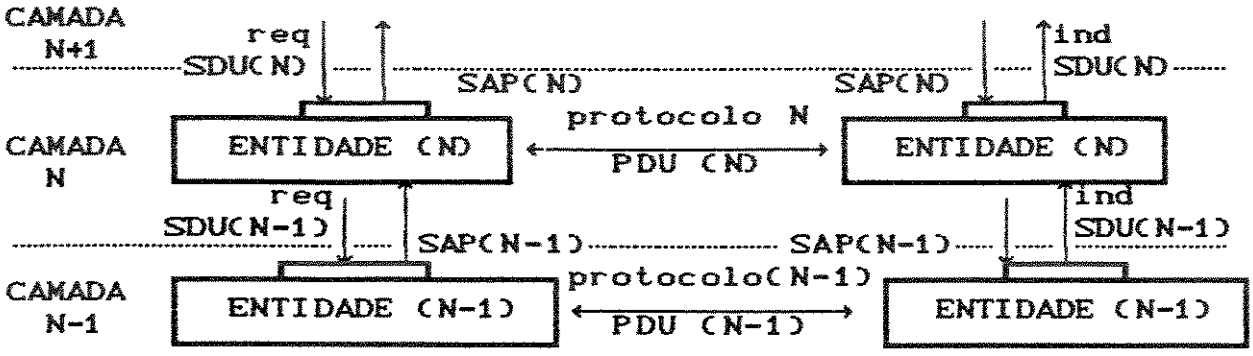


FIG. 1.0 - ARQUITETURA RM-OSI

No capítulo quatro, chamado Problema da Aplicação, apresentam-se os pré-requisitos necessários para a implementação de um programa de aplicação. Mostra-se também, um exemplo de implementação de um programa de aplicação no SISDI-MAP, baseado no Padrão Associado para os CLP's.

No capítulo 5, apresentam-se as conclusões do trabalho realizado, ressaltando que uma implementação correta da API e do Programa de Aplicação requer um investimento substancial de tempo, para o entendimento dos padrões especificados e para uma avaliação das implicações relacionadas com a escolha de diferentes modelos de implementação.

2.0 - INTRODUÇÃO

A interoperação em sistemas abertos é modelada através de interações dos Programas de Aplicação (AP's) que são representações abstratas dos aspectos das tarefas de processamento distribuído. Para a confecção de um AP com o objetivo de troca de mensagens entre os equipamentos programávis de uma rede MAP, o programador deverá ter conhecimento da API [14], [15], [16], [17] do MMS [04], [05] e de seus Padrões Associados [18], [19], [20]. Neste capítulo, introduzem-se os conceitos relacionados com estas entidades.

Durante este capítulo, dá-se uma ênfase especial ao Gerenciamento de Conexão e aos conceitos de Domínios e Invocação de Programas. Sendo que, estes dois últimos conceitos estão diretamente associados ao conceito do Dispositivo Virtual da Manufatura (VMD).

2.1 - O PROTOCOLO MMS

O serviço de Mensagens da Manufatura (MMS) é um protocolo da Camada de Aplicação proposto pela ISO e adotado pelo MAP [09]. Este protocolo possibilita a troca de mensagens entre Controladores Numéricos Computadorizados (CNC'S), Controladores de Robôs (CNR'S), Controladores Programáveis (CLP'S), Veículos Guiados Automaticamente (AGV'S), etc e vários tipos de computadores. Neste escopo, aplica-se o modelo cliente-servidor, onde o cliente é, em geral, um equipamento mais universal, enquanto que o servidor é projetado usualmente para uma aplicação mais específica. Contudo, um certo equipamento poderá ter que exercer, em instantes diferentes, alternadamente os papéis de cliente e de servidor.

A especificação do MMS define a notação das mensagens e contem informações do que deve ocorrer quando uma mensagem é recebida. Nesta notação, define-se o objetivo, tamanho e conteúdo de cada elemento de uma mensagem. Este serviço de mensagens permite aplicações como a troca de arquivos entre Computadores de Célula, ou a alteração, parcial ou por completo, realizada pelo Computador de Célula, de um programa aplicativo de controle de robô, localizado no seu CNR respectivo, ou a troca de informações entre os equipamentos programáveis de uma Célula, etc... Os serviços fornecidos permitem a implementação de um grande número de funções de chão de fábrica.

2.1.2 - SERVIÇOS E PRIMITIVAS

Como os outros protocolos da Camada de Aplicação, cada Entidade MMS comunica-se com os seus usuários hierarquicamente superiores

através da troca de primitivas de serviços e com qualquer outra Entidade MMS, em um outro nó da rede MAP, através de PDU'S. Existem quatro tipos de PDU'S no MMS ("request", "response", "error" e "reject") e quatro tipos de primitivas ("request", "indication", "response" e "confirmation"), figura 2.0.

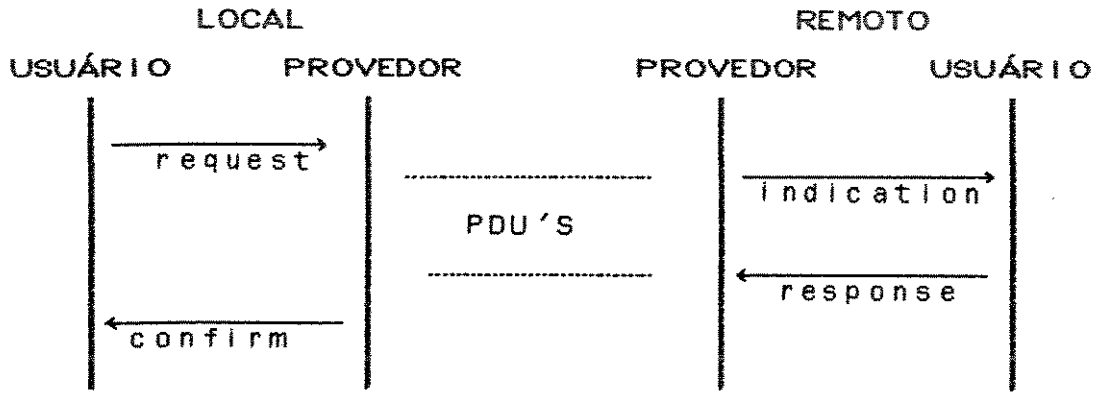


FIG. 2.0 - TROCA DE PRIMITIVAS ENTRE USUÁRIOS MMS

Se o usuário requisita um serviço ao MMS, ele irá gerar uma PDU para ser transmitida a uma Entidade MMS par. Por exemplo, as primitivas de "response" ou "confirmation" são positivas,se as requisições para um serviço MMS são realizadas com sucesso e, negativas, em caso contrário. Isto significa que, se uma entidade MMS recebe uma primitiva de "response" positiva do usuário ela irá transmitir uma PDU de "response". Mas se ela recebe uma primitiva de "response" negativa ela irá transmitir uma PDU de "error". PDU'S de "reject" são transmitidas pela entidade MMS quando da ocorrência de erros de protocolo ou no recebimento de PDU'S não identificadas.

2.1.2 - MODELAMENTO DOS OBJETOS

O comportamento do MMS visível ao usuário é modelado por uma entidade chamada VMD, que tratará, de forma abstrata, dos elementos estruturais e dos objetos envolvidos nos dispositivos reais. Cada objeto é referenciado pelo seu nome e é caracterizado por um número de atributos, que descrevem as características do dispositivo modelado. Alguns serviços MMS podem modificar estes objetos e, portanto, seu mapeamento no equipamento real, pode ser modelado por uma instanciação de um ou mais atributos do objeto. Na figura 2.1, mostra-se um exemplo de aplicação do modelamento por objeto. Este exemplo faz parte das especificações do Padrão Associado para os CNR's [20]. O objeto Manipulador do Rôbo (Robot Arm), descrito, contem todos os elementos do braço de robô que são normalmente controlados pela Invocação de Programa Manipulador do Robô.

Na implementação de um servidor MMS deverá providenciar-se o mapeamento do modelo VMD com aspectos funcionais e recursos do dispositivo real da manufatura.

O VMD é composto dos seguintes blocos funcionais:

- **Uma Função Executiva**: este bloco funcional administra o acesso aos recursos do VMD, tornando-os visíveis para os clientes-usuários. A Função Executiva exprime os recursos na forma de memórias, processadores, portas de I\O, etc, que serão usadas para descrever os elementos constituintes do Domínio:

```
Object: Robot Arm
Attribute: Local (LOCAL, NOT LOCAL)
Attribute: DevicePowerOn (TRUE, FALSE)
Attribute: DeviceCalibrated (CALIBRATED, NOTCALIBRATED,
                             CALIBRATING)
Attribute: NumberOfJoints - Integer
Attribute: ControlType (VELOCITY, POSITION, FORCE...)
Attribute: BASE WORLD - Pose
Attribute: Servomechanism
  Attribute: MICS-BASE - Pose
  Attribute: OrderedListOfJointDescription
    Attribute: JointType (REVOLUTE, PRISMATIC)
    Attribute: Calibrated ( CALIBRATED, NOTCALIBRATED,
                            CALIBRATING)
    Attribute: JointBrakes (TRUE, FALSE)
    Attribute: UpperBound - Floating point
    Attribute: LowerBound - Floating point
    Attribute: Joint Servo
      Attribute: TerminationConditionMet (TRUE, FALSE)
      Attribute: ActualControlValue - Floating point
      Attribute: DesiredControlValue - Floating point
      Attribute: ActualJointValue - Floating point
      Attribute: MovingEnabled (TRUE, FALSE)
Attribute: End Effector
  Attribute: ID Numbers
  Attribute: Toll Descriptor
  Attribute: TOOL MICS - Pose
  Attribute: User Detail
Attribute: PathPlanner
  Attribute: USER BASE - Pose
  Attribute: DesiredTOOL-USER Pose
  Attribute: SpeedFactor - Floating point
  Attribute: SpeedMaxBound - Floating point
  Attribute: ProgrammedSpeed - Floating point
```

FIG. 2.1 - O MODELAMENTO DE UM OBJETO

- **Um ou mais Domínios**: Um Domínio consiste dos aspectos do VMD, que estão associados com um elemento específico de uma estratégia de controle ou monitoração;

- **Zero ou mais Estacão do Operador**: consiste no conjunto de capacidades de entrada e saída do VMD, que estão associadas à troca de mensagens com o operador;

- **Um Sistema de Arquivos Virtuais Opcional**: consiste de um conjunto de arquivos rotulados, que quando presentes, atuam como um substituto para o armazenamento de dados e programas. O Sistema de Arquivos Virtuais MMS é definido como um subconjunto de Arquivos Virtuais do FTAM [04].

2.1.3 - OS SERVIÇOS DE GERENCIAMENTO DE CONTEXTO

O Protocolo MMS define dois tipos de serviços: os confirmados e os não confirmados. Os serviços confirmados são aqueles em que, o usuário solicitante espera por uma resposta do usuário par, para que a execução deste tipo de serviço seja considerada completa.

Para a execução de todos os serviços MMS, tanto confirmados como não confirmados, faz-se necessário o estabelecimento de um contexto entre os usuários MMS comunicantes. Este contexto é estabelecido através de serviços específicos do Gerenciamento de Contexto.

O MMS define os seguintes serviços de Gerenciamento de Contexto:

- **Initiate**: Permite iniciar a comunicação com outros usuários MMS e estabelecer os requisitos, capacidades e as classes de conformidade que a comunicação irá suportar;

- **Conclude**: Permite encerrar a comunicação com outros usuários MMS de uma maneira normal;

- **Abort**: Permite encerrar a comunicação com outros usuários MMS de uma maneira abrupta;

- **Cancel**: Permite o cancelamento de um serviço requerido e ainda não confirmado;

- **Reject**: Permite ao Provedor MMS notificar, ao usuário local e ao remoto, a ocorrência de erros de protocolo;

Estes serviços de contexto estão baseados na máquina de estado do protocolo mostrada na figura 2.2 (vista tanto pelo usuário local quanto pelo usuário remoto).

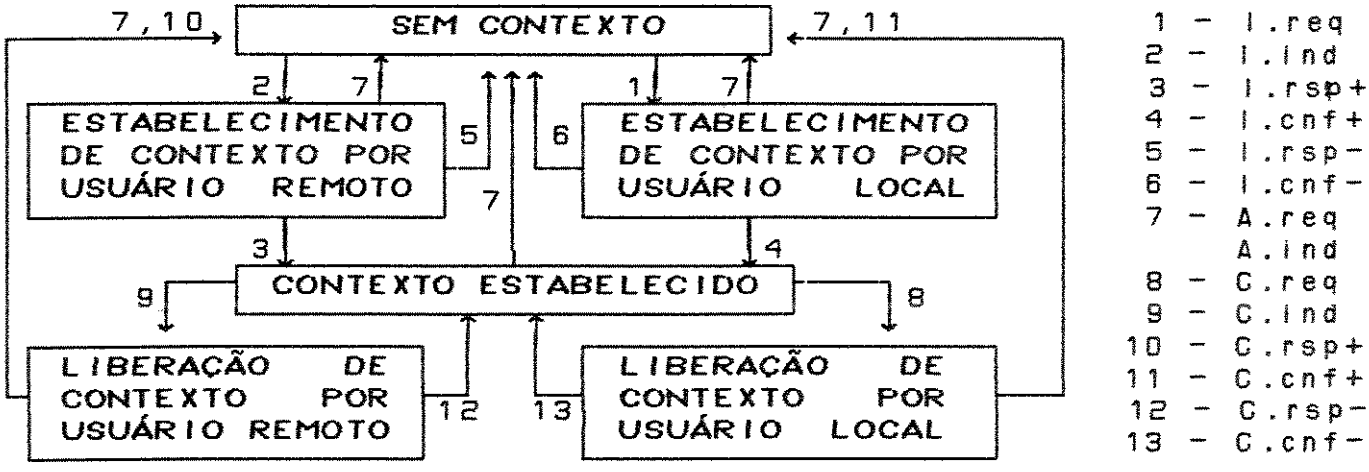


FIG. 2.2 - MÁQUINA DE ESTADO PARA O GERENCIAMENTO DE CONEXÃO

2.1.4 - SERVIÇOS GERAIS

Os serviços que podem ser solicitados dentro de um contexto MMS estabelecido são:

- **Serviço de suporte de VMD**: permite ao usuário MMS obter o estado de um VMD (por solicitação ou por mensagem espontânea), obter o nome dos vários objetos definidos no VMD, obter informações de identificação do usuário MMS respondedor, requerer do respondedor a mudança do nome do

identificador de um objeto;

- **Serviço de Gerenciamento de Domínio:** Permite ao usuário MMS operar dinâmicamente sobre domínios, solicitando seus atributos ou a criação/deleção de novos domínios de forma local ou remota. São disponíveis funções para o carregamento ("download") e o descarregamneto ("upload") de um programa em um domínio de um equipamento programável;

- **Serviço de Gerenciamento de Invocação de Programas:** Permite ao usuário MMS criar, cancelar e obter atributos de uma Invocação de Programa. Permite-se também operar sobre ela através de serviços como os de iniciação, parada, continuação e reiniciação;

  Uma Invocação de Programa é um objeto abstrato que consiste de um conjunto de elementos procedurais e de dados dentro de um VMD, podendo ser predefinida, ou dinamicamente definida através dos Serviços de Invocação de Programas.

- **Serviço de Acesso à Variáveis:** Permite ao usuário cliente acessar variáveis definidas no VMD servidor. Os serviços incluem definir e cancelar nomes de variáveis, tipos e lista de variáveis; obter informações, ler e escrever sobre atributos destes elementos de dados [04], [37];

- **Serviço de Gerenciamento de Semáforos:** Permite a sincronização, controle e coordenação dos recursos compartilhados entre usuários MMS. Para tal, definem-se serviços para a criação, cancelamento e requisição de informações associadas ao tipo do semáforo e aos seus respectivos estados;

- **Serviço de Comunicação com a Estação do Operador**: Permite ao usuário MMS obter dados ou entrar com dados de/para uma estação de operação do VMD;

- **Serviço de Gerenciamento de Eventos**: Permite ao usuário Cliente MMS definir e gerenciar eventos num VMD, bem como, obter notificações de eventos ocorridos, condicionados às situações de tempo real. Os serviços incluem criar, alterar, deletar e obter informações sobre Condições de Eventos e Ações de Eventos;

- **Serviço de Gerenciamento de Jornal (Arquivo LOG)**: Permite ao usuário MMS o registro e recuperação de informações ordenadas, incluindo eventos, variáveis relativas aos eventos e textos explicativos ou comentários. Os serviços incluem leitura, escrita, iniciação e informações sobre o estado do Jornal;

- **Serviço de gerenciamento de arquivos**: Provê as funções necessárias para leitura de arquivos contendo programas de controle e dados dos sistemas armazenadores de arquivos (nos equipamentos de controle e servidores de arquivo), e para gerenciar os sistemas de arquivos. Este serviço não forma uma parte integrante do padrão MMS, sendo considerado em Anexo no documento.

A maior parte destes serviços são serviços confirmados e estão baseados máquina de estado apresentada na figura 2.3 (vista tanto do usuário local quanto pelo usuário remoto).

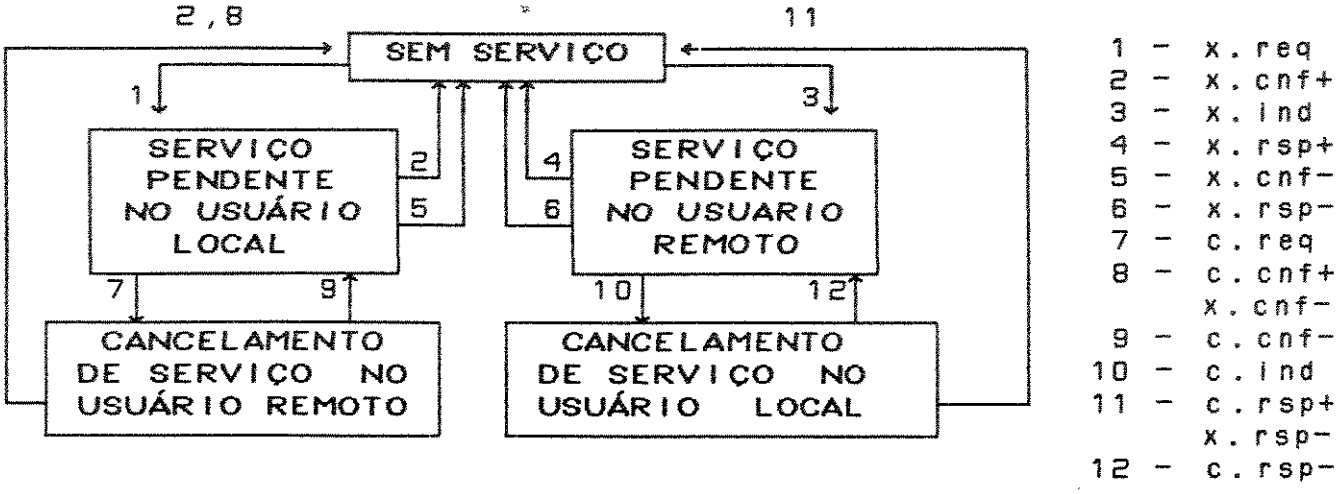


FIG. 2.3 - MÁQUINA DE ESTADO DOS SERVIÇOS CONFIRMADOS

2.1.5 - BLOCOS CONSTRUTIVOS DE CONFORMIDADE

Estes serviços são definidos de uma maneira genérica, sendo levados em consideração os mais diversos tipos e funções dos dispositivos programáveis da manufatura a que se destina o protocolo. Em virtude da grande quantidade de serviços oferecidos, prevê-se que, em implementações específicas, somente alguns sub-conjuntos poderão vir a ser considerados. Para tal introduzem-se agrupamentos de serviços, além das unidades funcionais propriamente ditas, chamados de Blocos Construtivos de Conformidade (CBB - "Conformance Building Blocks"). Os CBB's são designados de verticais ou horizontais. Os CBB's verticais correspondem a sub-conjuntos apropriados de serviços na mesma unidade funcional enquanto que os CBB's horizontais englobam serviços de unidades funcionais diferentes. Na figura 2.4 mostra-se os CBB's verticais por unidade funcional, para um subconjunto de unidades funcionais, com as suas respectivas primitivas MMS.

UNIDADE FUNCIONAL	CLASSES DE CONFORMIDADE	PRIMITIVAS
GERÊNCIA DE CONTEXTO	CON1, CON2	M_INITIATE
	CON1, CON2	M_CONCLUDE
	MMS1	M_ABORT
	CON3	M_CANCEL
	MMS1	M_REJECT
SUPORTE DE VMD	VMD1, VMD2	M_STATUS
	VMD1	M_UNSOLICITED_STATUS
	MMS2	M_GET_NAME_LIST
	MMS2	M_IDENTIFY
	VMD2	M_RENAME
GERÊNCIA DE DOMÍNIOS	DOM1, DOM3	M_INITIATE_DOWNLOAD_SEQUENCE
	DOM1, DOM3	M_DOWNLOAD_SEGMENT
	DOM1, DOM3	M_TERMINATE_DOWNLOAD_SEQUENCE
	DOM2, DOM4	M_INITIATE_UPLOAD_SEQUENCE
	DOM2, DOM4	M_UPLOAD_SEGMENT
	DOM2, DOM4	M_TERMINATE_UPLOAD_SEQUENCE
	DOM3	M_REQUEST_DOMAIN_DOWNLOAD
	DOM4	M_REQUEST_DOMAIN_UPLOAD
	DOM5	M_LOAD_DOMAIN_CONTENT
	DOM6	M_STORE_DOMAIN_CONTENT
	DOM7	M_DELETE_DOMAIN
	DOM8	M_GET_DOMAIN_ATRIBUTE
	FIL1	M_DOMAIN_FILE
GERÊNCIA DE INVOCAÇÃO DE PROGRAMAS	PRG1	M_CREATE_PROGRAM_INVOCTION
	PRG1	M_DELETE_PROGRAM_INVOCATION
	PRG2	M_START
	PRG3	M_STOP
	PRG4	M_RESUME
	PRG5	M_RESET
	PRG6	M_KILL
	PRG7	M_GET_PROGRAM_INVOCATION_ATTRIB.
ACESSO À VARIÁVEIS	VAR2	M_READ
	VAR3	M_WRITE
	VAR4	M_INFORMATION_REPORT
	VAR1	M_GET_VARIABLE_ACCESS_ATTRIBUTES
	VAR6,VNAM,VADR	M_DEFINED_NAMED_VARIABLE
	VAR7,VSCA,STR2	M_DEFINED_SCATTERED_ACCESS_ATTR.
	VAR6,VNAM	M_DELETE_VARIABLE_ACCESS
	VAR5,VNAM	M_DEFINE_NAME_VARIABLE_LIST
	VAR5	M_GET_NAMED_VARIABLE_LIST_ATTR.
	VAR5	M_DELETE_NAMED_VARIABLE_LIST
	VAR9,VADR,VNAM	M_DEFINE_NAMED_TYPE
	VAR8	M_GET_NAMED_TYPE_ATTRIBUTES
	VAR9	M_DELETE_NAMED_TYPE

FIG. 2.4 - UNIDADES FUNCIONAIS, CBB'S E PRIMITIVAS MMS

As classes horizontais são estabelecidas na descrição detalhada das primitivas, durante a definição dos parâmetros respectivos. Existem ao todo 10 classes horizontais de CBB's: RCV1, BTR1, STR2, NEST, VNAM, VADR, VALT, VSCA, SOCS.

Na definição das classes distinguem-se também três tipos de requisitos para as primitivas, resultantes do modelo básico Cliente-Servidor:

- papel de respondedor para o VMD (servidor) e de requisitante para o cliente;
- o contrário do caso anterior;
- ambos VMD/Cliente nos papéis de respondedor/requisitante.

2.2 - OS PADRÕES ASSOCIADOS

Como visto, o MMS fornece os serviços genéricos projetados para facilitar a interconexão de equipamentos programáveis da manufatura. Entretanto, o padrão não contém informações sobre aplicações específicas, informando somente como as mensagens devem ser montadas e enviadas.

Existem serviços MMS que possuem informações adicionais (não mostrados na especificação do MMS [04], [05]) e que, têm como objetivo fornecer funcionalidades específicas de um determinado equipamento, tais como: Comandos Numéricos Computadorizados, Controladores Programáveis, Controladores de Rôbos, etc. Estas informações específicas devem ser fornecidos por organizações apropriadas através dos documentos chamados de Padrões Associados ("Companion Standards") [18] , [19], [20].

Um Padrão Associado é formado das seguintes cláusulas:

- **Escopo e Introdução:** apresenta-se, nesta cláusula, uma explanação das funções do Padrão Associado e uma breve descrição das áreas de aplicação, e dos tipos de equipamentos aos quais o padrão está destinado. No estabelecimento do Escopo diferenciam-se as funções de cliente e servidor. Embora ambos, cliente e servidor, tomem parte na comunicação, o Padrão Associado é direcionado principalmente para a definição das funções do servidor, uma vez que esta função é considerada como a principal neste documento.

Na figura 2.5, apresentam-se modelos para interconexão entre estes equipamentos (servidores) e os Computadores de Controle/Supervisão (clientes), como definido no Padrão Associado para os Controladores de Robôs [20]. A primeira configuração consiste de um cliente controlando ou se comunicando com um robô. Em uma implementação simples desta configuração, será necessário estabelecer uma única associação MMS. Em implementações mais sofisticadas poderão coexistir associações múltiplas concorrentes. Na segunda configuração o robô é o único servidor, mas existem múltiplas associações. Num exemplo de aplicação desta configuração, um dos Computadores de Célula realizaria tarefas de controle e o outro tarefas de monitoração. Na terceira configuração o robô é cliente de um ou mais equipamentos. Embora o robô possa ser cliente de outros equipamentos, tais como Sistemas de Visão e Soldadores, o Padrão Associado não especifica classes de conformidade ou serviços ou protocolos necessários para um robô nesta função. Estes requisitos devem ser fornecidos pelo Padrão Associado do servidor em questão.

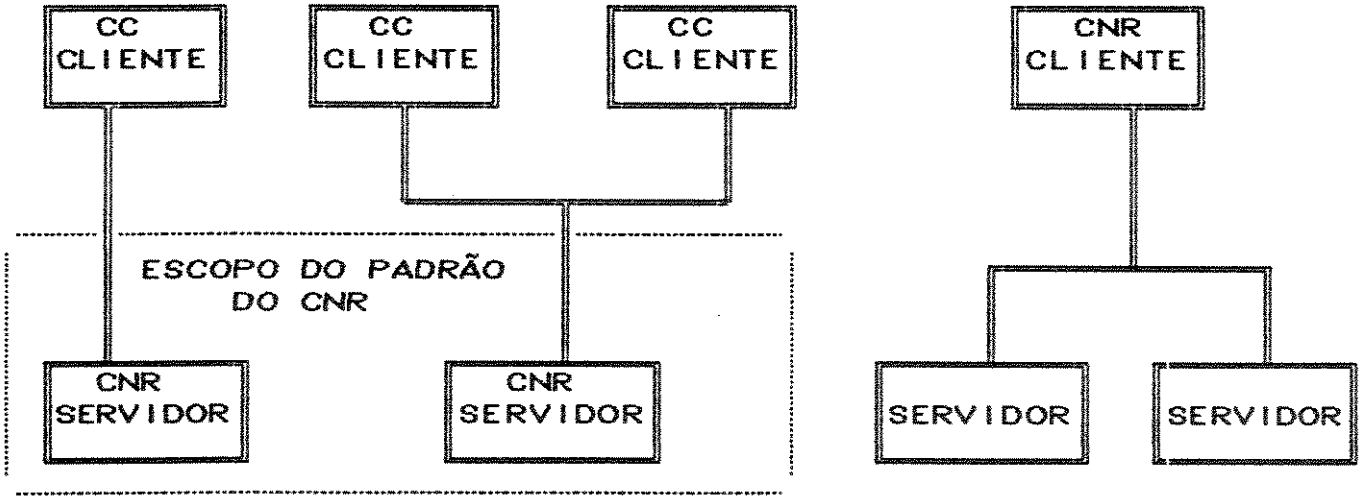


FIG 2.5 - VARIAÇÕES DO MODELO DE APLICAÇÃO DO CNR

- **Contexto e Funções**: define-se aqui o Contexto no qual o Padrão Associado é utilizado. Este contexto é principalmente descrito através de um modelo de aplicação do equipamento específico, que é composto do modelo funcional do equipamento e da sua máquina de estados. Esta cláusula deve conter também, uma descrição textual de todas as funcionalidades apresentadas pelo equipamento. Não obstante, um equipamento real pode apresentar características além das descritas pelo seu respectivo Padrão Associado.

  Tomando como exemplo o Padrão Associado para o CNC, mostra-se, na figura 2.6, o modelo funcional deste equipamento, onde as entradas, saídas, e os métodos de controle utilizados são explicitamente indicados no documento. Este modelo esclarece os objetos que o CNC faz uso para realizar suas tarefas.

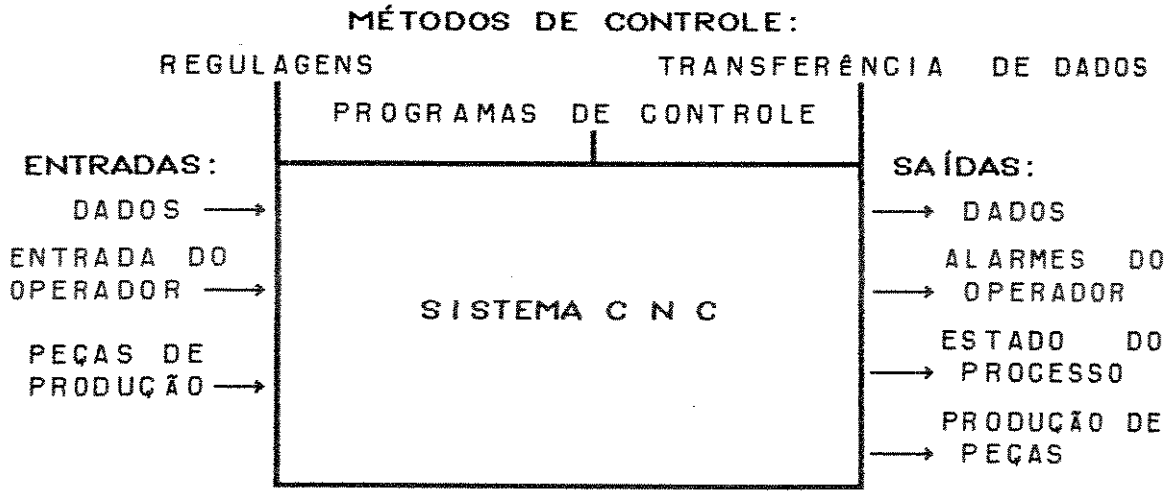


FIG 2.6 - O MODELO FUNCIONAL DO CNC

- **Mapeamento das Funções para o MMS**: especifica-se, nesta parte, o mapeamento do modelo do equipamento, com as suas funcionalidades específicas, em atributos do VMD e o mapeamento de Objetos, específicos de determinadas aplicações, em Domínios e Invocações de Programas. Outros Objetos MMS abstratos, tais como variáveis nomeadas, também podem ser especificados.

  Como exemplo, o NC-VMD, definido no Padrão Associado dos CNC's [19] e descrito na figura 2.7, consiste de uma coleção de Objetos e atributos específicos do MMS e dos CNC's. Esta representação identifica aqueles atributos que têm interpretação específica dentro do contexto dos Comandos Numéricos. O atributo NC State, a título de exemplo, define o estado corrente da relação cliente/servidor (controle local ou remoto) ou o estado operacional corrente do equipamento (ocioso, pronto, ativo, suspenso, etc...).

- **Definição de Serviços e Protocolos Específicos**: esta cláusula traz tanto alterações nos serviços especificados pelo MMS, como serviços adicionais para serem utilizados somente pelo equipamento específico. Quando uma função específica de um determinado equipamento é executada, pode ser necessário a inclusão de máquinas de estado para padronizar esta operação, neste caso, o Padrão Associado define esta máquina de protocolo. Operações simples, envolvendo somente um serviço MMS não necessitam de uma máquina de estado adicional no Padrão Associado, porque, neste caso, a máquina de estado definida no MMS é suficiente;

```
Object: NC-VMD
  Todos os Atributos do MMS
  Attribute: NC State
  Attribute: Current State
  Attribute: List of Tool Matrices
  Attribute: List of Machine Paths
  Attribute: List of Pallet Locations
  Attribute: List of Machine Axes
  Attribute: List of Machine Controls
  Attribute: List of Executable Programs
  Attribute: List of Parts
  Attribute: List of Pallets
  Attribute: List of Local Stores
  Attribute: List of Accessories
```

FIG 2.7 - ATRIBUTOS ADICIONAIS DO NC-VMD

Como exemplo, o serviço Initiate Download Sequence foi extendido no Padrão Associado dos CNC's, de tal maneira que se permita identificar o tipo do Domínio que está sendo referenciado. A adição dos novos parâmetros, mostrados na figura 2.8, foi feita através do parâmetro DomainDetail. O atributo Domain Type, a título de exemplo, identifica o tipo e o formato dos dados que o Domínio contém. Nas aplicações dos servidores, o conteúdo do Domínio consiste de programas executáveis ou de dados. Estes Domínios são associados a uma Invocação de Programa que, quando executada, se traduz em ações físicas realizadas pelo equipamento.

NOME DO PARÂMETRO	RSP	CNF
DOMAIN TYPE	M	M
DOMAIN SIZE	U	U
DOMAIN DATE	U	U
DOMAIN CONTENT FILENAME	U	U

FIG 2.8 - O PARÂMETRO DOMAINDETAIL

Os padrões associados especificam uma classe de conformidade básica que o equipamento deve suportar. Esta classe de conformidade básica forma um "kernel" que é incluído em todas as outras classes de conformidade.

2.3 - A INTERFACE DE PROGRAMAS DE APLICAÇÃO

Uma interface pode ser vista como um local onde sistemas independentes se comunicam. Um bom exemplo da aplicação de uma interface é o telefone. O telefone fornece ao seu usuário o serviço de comunicação de voz entre dois pontos de uma rede. Para poder se comunicar através desta rede o usuário deve estar familiarizado com um conjuto de sinais da interface. Mediante estes sinais, existe um conjunto padrão de regras a serem seguidas para o estabelecimento de uma comunicação. Por exemplo, o usuário chamante deve esperar por um tom constante antes de entrar com o número do usuário chamado. Este número deve ser entrado na ordem correta para que a ligação tenha sucesso.

Desta maneira, não é necessário que o usuário tenha conhecimento dos detalhes de funcionamento da rede ou da interface. O usuário somente necessita ter conhecimento das funcionalidades da interface para usar o telefone. Além do mais sendo a interface de telefone uniforme, o usuário pode acessar a rede sem tomar conhecimento do tipo do telefone ou da sua localização na rede.

A Interface de Programas de Aplicação (API), especificada em [14], deve ser transparente para o seu usuário da mesma forma que a interface descrita acima. Visto que a Camada de Aplicação contém uma variedade de protocolos diferentes que fornecem os serviços de rede utilizados pelo usuário do sistema de comunicação, é essencial que este usuário tenha conhecimento das funcionalidades da API para poder se comunicar usando a rede, mas não é necessário que ele tenha conhecimentos dos seus detalhes de funcionamento.

O Programa de Aplicacão (AP) utiliza a API como um mecanismo de acesso para os protocolos específicos da Camada de Aplicação

(SASE). Na figura 3.0 mostram-se as ligações entre o AP, a API e os SASES.

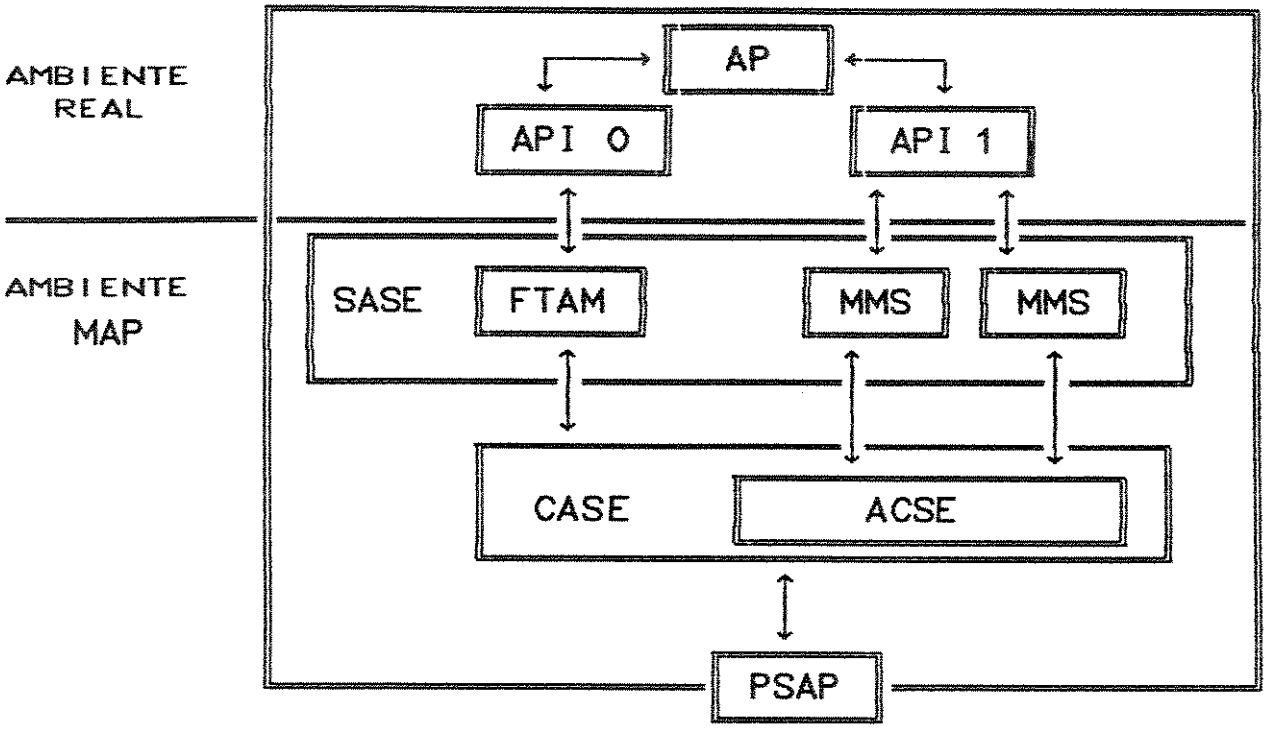


FIG 2.10 - A INTERFACE DO PROGRAMA DE APLICAÇÃO NO MAP

Algumas das vantagens de utilização da API são:

- O objetivo da portabilidade de programas de usuário para ambientes diferentes não pode ser alcançado usando-se somente a API. Contudo pode-se reduzir os custos de treinamento de programadores e do pessoal de manutenção que utilizam a interface em ambientes diferentes;

- Auxílio na migração de programas aplicativos para futuras versões da rede MAP;

- Execução de tarefas que são em geral de responsabilidade do usuário;

- Incentivo ao desenvolvimento de software aplicativo para as redes MAP;

2.3.1 - A ESPECIFICAÇÃO DA INTERFACE

A especificação da interface é independente do sistema operacional, contudo, pode-se prever que, se um padrão de sistemas operacionais vier a ser especificado pela ISO, as versões futuras da especificação desta interface deverão migrar na direção deste ambiente.

O Projeto MAP requer que tanto a arquitetura OSI quanto a arquitetura EPA sejam suportadas pela interface. Estes dois modelos são exemplos de comunicação "connectionless" e "connection-oriented". A principal diferença entre estes dois modelos é que no primeiro caso cada transferência de dados é completamente independente de todas as outras, não necessitando da manutenção de um contexto para a associação, como no segundo caso.

A interface deverá também suportar operações síncronas e assíncronas. A implementação de operações assíncronas restringe a utilização da interface a ambientes multitarefa que suportem a comunicação e a coordenação entre processos. Esta característica de ambiente multitarefa pode ser alcançada através do uso de um Kernel de tempo real, de tal forma que todas as tarefas compartilhem um espaço de endereçamento comum, e que cada uma, em separado, possua seu próprio contador de programa e sua própria pilha.

Numa operação assíncrona tão logo a mensagem seja entregue à tarefa do provedor de serviços, a tarefa do usuário retoma o controle e continua a sua execução. Sendo executada em paralelo com a tarefa do usuário que originou a chamada, a tarefa do provedor de serviços está constantemente monitorando suas portas de entrada à procura de mensagens para processar.

A emissão de uma chamada assíncrona requer a implementação de um mecanismo que permita ao provedor de serviços esperar por um evento específico ou por uma disjunção de eventos.

Define-se como um evento, a complementação de uma operação iniciada via chamada de funções assíncronas. Em todas as chamadas de funções assíncronas, o nome do evento deve ser incluído como parâmetro de entrada obrigatório, para que possa ser sinalizado quando do término do serviço.

Cada chamada de função está associada a um serviço fornecido pelo provedor de serviços da interface. Assim, em virtude dos diferentes tipos de serviços fornecidos ao usuário e da diversidade de parâmetros de entrada/saída envolvidos, esta interface é destinada a linguagens de programação que associem diferentes estruturas de dados ao mesmo espaço de dados na memória. Exemplos destas linguagens são C, PASCAL E PL/1. A aplicação desta interface para linguagens não procedurais, como por exemplo linguagens orientadas a objeto, ainda está indeterminado na especificação MAP.

A interface requer que dados de vários tamanhos sejam passados do usuário para o provedor de seviços e vice-versa. Todos estes dados devem ser passados em uma área de dados ou buffer controlado pelo usuário.

Em muitas funções da interface, a quantidade de dados varia a cada vez que uma função particular é requerida pelo usuário. Além

disto, em instâncias diferentes de uma mesma chamada de função, podemos encontrar duas situações diferentes com relação aos dados retornados. Na primeira situação, o usuário não teria meios de saber antecipadamente a quantidade de dados a ser retornada. Já na segunda situação, o usuário sabe exatamente o espaço requerido para armazenar os dados de retorno. Para tratar estas diferentes situações, tão flexível quanto possível, dois métodos para gerenciamento dos buffers são definidos:

- O programa do usuário pode alocar explicitamente o buffer para a troca de mensagens entre o programa do usuário e a interface do provedor de serviços.

- Alguns parâmetros de certas funções podem requerer alocação automática do buffer pela interface. Neste caso a interface aloca um dos buffers do pool de buffers do usuário.

Em ambos os casos descritos acima a responsabilidade de liberação da memória alocada é do usuário.

Inicialmente, foram especificadas no Projeto MAP/TOP API's para os protocolos MMS, FTAM e Comunicações Privadas.

2.3.2 - MODELO GERAL DA INTERFACE

Para o entendimento do modelo da interface, faz-se necessário apresentar as três classificações existentes para as funções da interface, sendo cada uma independente das outras. São elas:

- **Funções de Alto (HLS) ou de Baixo nível (LLS)**: como visto na figura 3.0, acima, os serviços fornecidos pela API estão associados a um protocolo específico da camada de

aplicação. Nos serviços de alto nível não se pode ter acesso a todas as funcionalidades da máquina de estados destes protocolos. Um serviço de alto nível típico faz uso de vários serviços de baixo nível, para completar a sua tarefa e pode ser interrompido pelo usuário que o iniciou. Os serviços de baixo nível, por sua vez, fornecem ao usuário acesso completo à máquina de protocolos, sendo bastante útil para testes do protocolo ou para programadores que desejem ter um controle mais próximo sobre todas as interações com a máquina de protocolos.

- **Modo Requisitante ou modo Respondedor**: as funções do primeiro modo o permitem ao usuário requisitar serviços da rede, sendo uma operação típica de aplicações FTAM. As funções do modo respondedor, por outro lado, existem no nó par à uma requisição e estabelecem mecanismos para a resposta automática das funções do modo requisitante. A interface para o MMS exige o tratamento de tais funções.

- **Sensíveis ao Contexto ou Livre de Contexto**: uma função livre de contexto não é influenciada por nenhuma função emitida antes, nem afeta qualquer função emitida após a sua emissão. A função sensível ao contexto é o oposto do caso anterior, sendo extremamente dependente do contexto previamente estabelecido.

O modelo da API é dividido em duas partes e constitui-se da região do programa do usuário e do provedor de serviços da interface.

Estas duas partes estão organizadas em forma de blocos funcionais, como pode ser visto na figura 2.11. No modelo, cada bloco funcional não implica necessariamente em uma tarefa separada em uma implementação real, eles têm como objetivo facilitar a descrição das funcionalidades necessárias à interface, quando da requisição de um serviço pelo usuário.

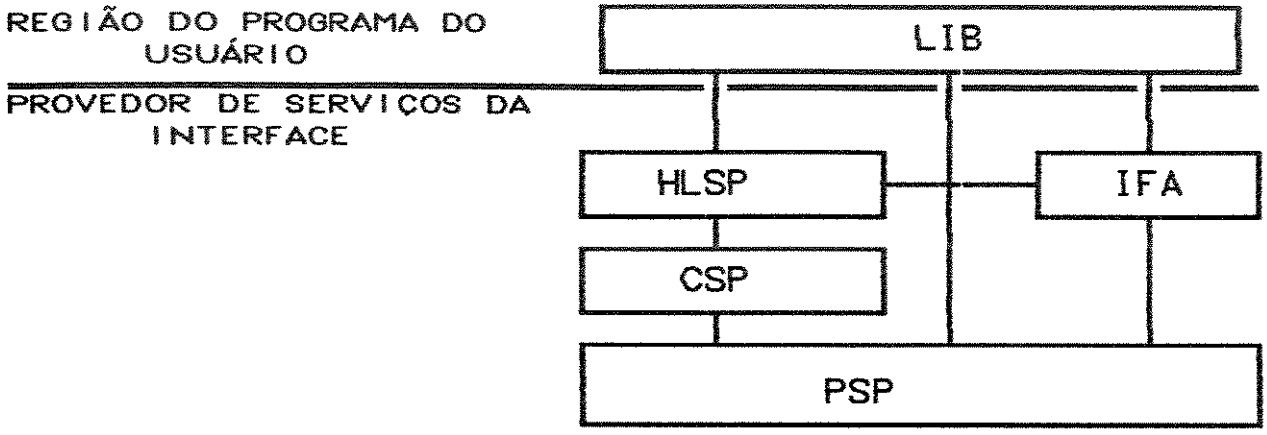


FIG. 2.11 - MODELO GERAL DA API

2.3.2.1 - A REGIÃO DO PROGRAMA DO USUÁRIO

Esta região contem basicamente uma biblioteca de funções que será chamada pelos programas aplicativos, para requisitar os serviços desejados da rede. Cada chamada à biblioteca é, no mínimo, responsável pela formatação da requisição de serviço, a ser processada pelo provedor de serviços da interface.

Existem duas classes de chamadas da biblioteca: serviços de baixo nível (LLS) e serviços de alto nível (HLS). Nos serviços de baixo nível consideram-se primitivas req/rsp/ind/conf isoladamente, já dos serviços de alto nível espera-se maior funcionalidade, de tal forma que uma chamada corresponda a uma sequência completa de chamadas LLS. Nesta região da interface encontra-se, também, o módulo dos Serviços Respondedores (RS), cuja função é a de examinar automaticamente a chegada de "indicações" de serviços não confirmados e de determinar um particular curso de ação.

2.3.2.2 - A REGIÃO DO PROVEDOR DE SERVIÇOS DA INTERFACE

Os serviços disponíveis do Provedor de Serviços da Interface são fornecidos por uma tarefa que não faz parte do Programa de Aplicação e sua interação com o usuário é sempre feita através da troca de mensagens. Na requisição do serviço, o usuário envia uma mensagem ao provedor de serviços que, por sua vez, enviará uma mensagem de resposta notificando sua terminação.

Pode-se dividir esta região em procedimentos de controle e em procedimentos de envio/recepção de primitivas. Os procedimentos de controle são três : Provedor de Serviços de Alto Nível (HSLP), Provedor de Serviços Confirmados (CSP), e o Árbitro e Filtro de Indicações (IFA). O HSLP é específico da Entidade de Aplicação, estando interligado com o CSP o IFA e o PSP. O CSP providencia o serviço de requisição e de confirmação como um todo, sendo específico de conexão. O IFA examina as indicações solicitadas ou não que chegam à API, sendo específico da conexão. A inteligência de uma função respondedora (RS), descrita anteriormente, é implementada no Arbitro e Filtro de Indicações (IFA). Com a ativação desta inteligência, o usuário poderá delegar recursos (tais como o acesso a variáveis compartilhadas pela rede) ou capacidades (tal como uma chamada de função que possua autorização para a execução de tarefas físicas em um robô). Uma vez ativada, esta inteligência irá filtrar a chegada de indicações não solicitadas de tal forma que só haja resposta para aquelas que o usuário, está programado para responder.O PSP providencia o serviço de envio de primitivas de requisição e resposta, e também o serviço de recebimento de primitivas de confirmação e indicação, para/da Máquina de Protocolo (PM). A Máquina de Protocolo contém a tabela de estados, o conjunto de regras do protocolo e todas as verificações de erro relacionadas. Esta Máquina de Protocolo é descrita externamente à interface de aplicação. O PSP terá também

como função, informar a todos os seus usuários o recebimento de indicações não solicitadas que possam afetar suas operações. Como, por exemplo, o recebimento de uma indicação de conexão abortada.

2.3.3 - MODELO DE API PARA O MMS

No SISDI-MAP, a API foi implementada para o protocolo MMS. Como visto anteriormente, este protocolo é destinado à troca de mensagens entre equipamentos da manufatura, tal como um computador e um comando número (CNC). Neste escopo, estabelece-se claramente uma relação de cliente/servidor entre o computador de controle e os demais equipamentos da manufatura, o que pode levar a um esquema de resposta dos serviços solicitados por parte da API. Portanto como descrito em [30], [31], [38], deve-se adicionar ao modelo da API uma nova unidade funcional para Processamento de Indicações (IP). Esta unidade tem como função processar os serviços solicitados pelas primitivas de indicação aceitas, acessar a memória e o diretório local de arquivos, quando necessário, e solicitar ao CSP ou ao PSP o envio de primitivas de resposta às indicações aceitas.

Para que esta nova unidade funcional possa realizar as funções de acesso à memória principal e diretório de arquivos torna-se necessário a introdução de mais uma unidade funcional,VDRDB (Virtual Device/Real Device Binding), que faça o mapeamento entre os Dispositivos Virtuais da Manufatura (VMD) e os Dispositivos Reais a que devem corresponder. Na figura 2.12, descreve-se o modelo da API para o MMS.

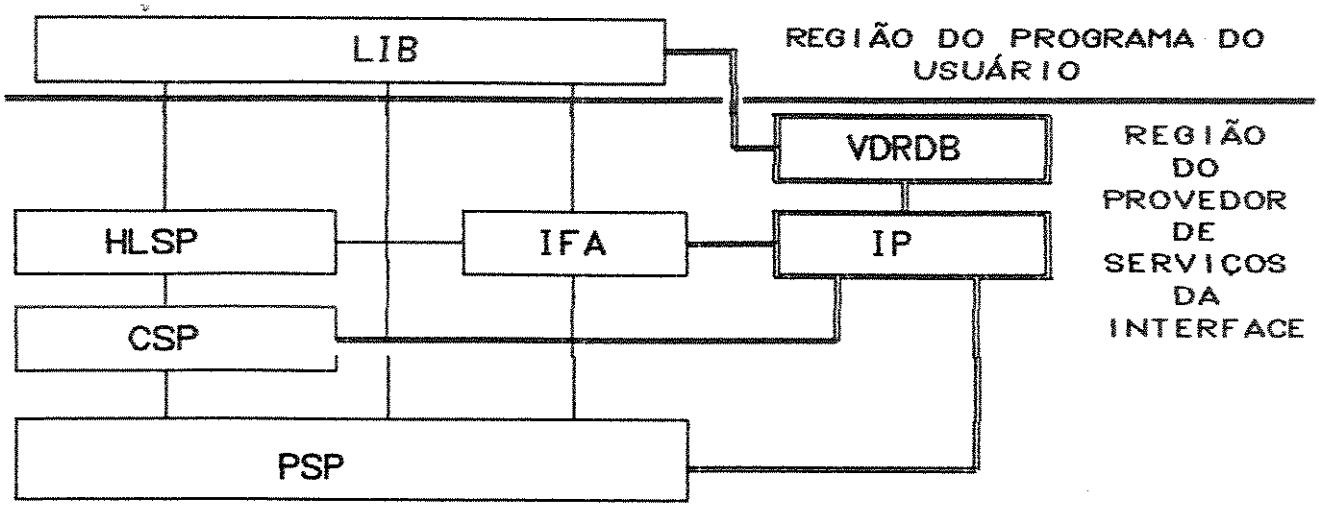


FIG. 2.12 - MODELO DA API PARA O MMS

2.3.4 - FUNÇÕES DE SUPORTE

Visando a operação global da interface, são especificadas no documento várias funções de apoio. Para que o programa de aplicação (AP) possa interagir assincronamente com os serviços da rede, a interface providenciará um serviço de Gerenciamento de Eventos que permite definir, notificar e esperar por eventos. Estas funções de gerenciamento são expostas ao usuário através de serviços "WAIT" e "NOTE". O mecanismo de evento generaliza o conceito de semáforos definidos por Dijkstra, de tal maneira que um processo possa esperar por uma disjunção de eventos múltiplos.

No tocante a troca de mensagens, a interface as processará nos mais diversos tamanhos e formatos. O tamanho das mensagens varia fortemente consoante os serviços invocados e o seu destino (usuário ou provedor). Em geral, lança-se mão de métodos de gerenciamento de "buffers", em que o usuário, ou a própria interface, poderão alocar dinamicamente o espaço necessário. No que diz respeito à passagem de parâmetros nas chamadas funcionais,

a API estabelece procedimentos precisos. Consideram-se as classes de parâmetros mandatórios e de parâmetros opcionais: os primeiros serão explicitamente especificados na chamada enquanto os parâmetros opcionais são colocados num Bloco de Controle de Dados (DCB). Existe associado a este conceito, um serviço de Inicialização Dinâmica de Blocos de Controle de Dados (DIDCB) que aloca e desaloca um DCB para um serviço particular.

O retorno de um serviço pode ser positivo ou negativo. Um retorno negativo está associado a um código de erro. Para que este código de erro possa ser apresentado ao usuário em um formato mais amigável, uma outra função de suporte, chamada "GET_PRINTABLE_ERROR", está definida para a tradução destes códigos de erro em "strings".

Existe na especificação um código de erro chamado "NO_CONNECTION_RESOURCE", que indica uma perda temporária de recursos. Para que o usuário possa ser notificado da liberação destes recursos, define-se na especificação a função "NOTEWHENCLEARED".

2.3.5 - SERVIÇOS DE GERENCIAMENTO DE CONEXÃO

A camada de aplicação é composta de uma ou mais Entidades de Aplicação. Dentro do Nível de aplicação, uma Entidade de Aplicação (AE), é o único objeto que pode ser localmente endereçado em um ambiente OSI. Uma AE pode ter várias invocações, ou seja, poderá estar sendo utilizada por vários Programas de Aplicação (AP's) ao mesmo tempo. Deverá também estar associada a um PSAP ("Presentation Service Access Point") onde poderão coexistir diversas conexões controladas por um parâmetro específico da rede.

Note-se que neste ponto, há necessidade de se esclarecer os conceitos de conexão e de associação: uma conexão está sempre ligada a existência dos SAP's, desta forma não existem conexões a nível da Camada de Aplicação e sim associações.

Todo Programa de Aplicação, que deseje fazer uso dos serviços de rede, deve iniciar o processo de estabelecimento de uma associação por meio de um pedido de Ativação da Entidade de Aplicação, através da função "AE-ACTIVATION". Este serviço de ativação de AE irá fornecer um "AE-LABEL" para o AP, indicando a qual invocação de AE ele estará associado e o registrará como um usuário autorizado da rede. Nas requisições de serviço subsequentes o AP será identificado por este parâmetro. O reverso deste procedimento de ativação é o procedimento de desativação, chamado "AE-DEACTIVATION", que tem como resultado, o cancelamento do "AE-LABEL" gerado anteriormente.

Após a ativação da AE, o próximo passo do AP é a requisição do serviço de Estabelecimento de Associação ("CONNECT"). Para a execução deste serviço, o "AE-LABEL", gerado pela função "AE-ACTIVATION", faz-se necessário, pois cada conexão está relacionada com uma determinada invocação. Sempre que um pedido de conexão é emitido, um Contexto de Aplicação deve ser selecionado de tal forma que a associação tenha sentido. Este contexto faz-se necessário para que o CASE (especificamente no caso do SISDMAP o ACSE) possa se entender com o SASE (especificamente no caso do SISDMAP o MMS), de tal forma que o AP do nó chamado (par remoto) não receberá uma indicação de associação, a menos que haja aprovação do CASE e do SASE do nó chamante. Quando uma indicação de associação emerge do Endereço de Apresentação, no nó chamado ,

uma função de gerenciamento da AE só entrega a indicação para a AE correspondente se houver uma permissão explicita para o seu recebimento. Tal permissão pode ser conseguida através de uma chamada da função "LISTEN", pelo AP do nó chamado (par remoto). Caso contrário, todas as indicações de associação para a referida AE serão automaticamente rejeitadas. Ao AP também é permitido a emissão de uma função "STOP-LISTEN" que indica uma alteração no seu desejo de receber indicações de conexão. A partir da requisição deste serviço, a intenção é de não receber mais qualquer pedido de associação.

Através da função "ANSWER" o AP poderá aceitar ou rejeitar a indicação de associação. Esta função faz uso do parâmetro "CONNECTION-ID", que é preenchido como parâmetro de saída pela função "LISTEN" e retornado ao AP . Tão logo uma resposta para o pedido de associação seja detectada no nó chamante, o SASE correspondente será informado. Parte da resposta é específica do SASE sendo que o restante deve ser apresentada ao usuário do ACSE. Se a resposta recebida é positiva, o serviço de estabelecimento de conexão será terminado com sucesso e o parâmetro "CONNECTION_ID", retornado ao AP. Este identificador de conexão corresponde ao CEP dentro do PSAP. No caso de uma resposta negativa, o ACSE e o SASE irão terminar a conexão pendente. Na figura 2.13 [24], mostra-se o modelo básico para estabelecimento da conexão.

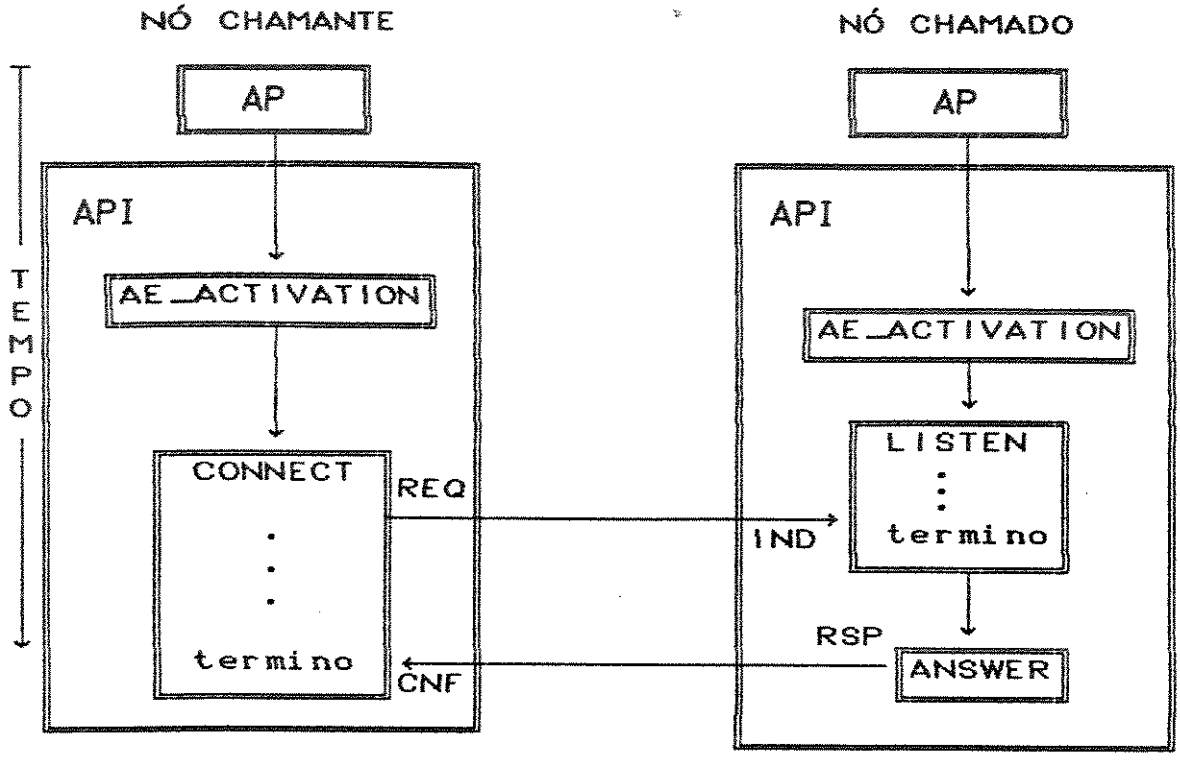


FIG 2.13 - ESTABELECIMENTO DE CONEXÃO

Com o uso da função "RELEASE", uma associação estabelecida pelas funções "CONNECT/LISTEN/ANSWER" poderá ser terminada. Segundo a especificação, uma associação somente é considerada como terminada, quando a área de saída do pedido de "RELEASE" é preenchido com os dados de confirmação. Desta forma, permite-se ao usuário o recebimento de indicações do nó chamado, após a emissão de um pedido de "RELEASE" pelo nó chamante. Para que uma associação seja terminada de maneira abrupta, o usuário deverá emitir a função "ABORT". Esta função tem prioridade sobre todas as outras atividades dentro do provedor de serviços.

Finalmente, a API fornece o serviço de "INDICATION RECEIVE" para receber indicações não solicitadas pelo nó chamante e que passem pelo IFA, tais como: Indicações dos serviços de Abort, Release, Information Report e Unsolicited Status. Se o usuário está

programando de uma maneira assíncrona, recomenda-se que ele tenha sempre uma chamada de Indication Receive pendente para que ele possa receber estas indicações não solicitadas. Esta prática deixa sempre disponíveis os recursos do sistema, e evita que o usuário fique em "deadlock". Esta é uma função de modo requisitador.

Como resumo, e para melhor visualização dos serviços de gerenciamento gerais de conexão, listam-se, na figura 2.14, as suas funções com os seus respectivos parâmetros.

2.3.6 - O GERENCIAMENTO DE CONEXÃO PARA A API DO MMS

A API é especificada em [13] para ser uma interface mestre (cliente) para o MMS, ou seja, uma interface destinada para aplicações de controle ou monitoração. Dentro deste escopo uma API mestre estaria fazendo requisições a estações simples tais como CLP's CNC's CNR's, etc e recebendo indicações através da função Indication Receive. Contudo, nos serviços da unidade de acesso a variáveis ou de gerenciamento de semáforos, são definidas funções da interface para requisição e resposta destes serviços. Por exemplo, na unidade de acesso a variáveis existem funções para a definição de variáveis locais, que são disponíveis de acesso para outros AP's. E uma vez definidas a interface poderá responder às indicações de variável "VREAD" ou "VWRITE" automaticamente. Para tal a função Indication Receive deve ter seu contexto ampliado para o MMS, de tal forma que ela também possa ser usada para o recebimento de indicações de serviços MMS. Se as resposta às indicações forem automáticas, elas estarão a cargo do bloco funcional IP do provedor de serviços da interface, representado na figura 3.2. Se não forem automáticas, estarão a cargo do usuário. O tipo de serviço recebido é definido pelo parâmetro de saída "IndicationName" e os dados recebidos pelo parâmetro de saída "IndicationSpecificInformation", da função "INDICATION-RECEIVE".

NOME DA FUNÇÃO	PARÂMETROS DE ENTRADA	PARÂMETROS DE SAÍDA
XX_AEACTIVATION	my_dir_name return_event_name input_dcb	inout_dcb ae_label return_code
XX_AEDEACTIVATION	ae_label return_event_name	inout_dcb return_code
XX_CONNECT	ae_label return_event_name called_dir_name input_dcb	inout_dcb connection_id return_code
XX_LISTEN	ae_label return_event_name	inout_dcb connection_id return_code
XX_SLISTEN	ae_label return_event_name	inout_dcb return_code
XX_ANSWER	connection_id return_event_name input_dcb	inout_dcb return_code
XX_RREQUEST	connection_id return_event_name input_dcb	inout_dcb return_code
XX_RRESPONSE	connection_id return_event_name input_dcb	inout_dcb return_code
XX_ABORT	connection_id return_event_name input_dcb	inout_dcb return_code
XX_IRECEIVE	connection_id return_event_name input_dcb	inout_dcb indication_name return_code
XX_AERESET	ae_label return_event_name	inout_dcb return_code

FIG. 2.14 - FUNÇÕES GERAIS DO GERENCIAMENTO DE CONEXÃO

Além dos serviços de gerenciamento de conexão descritos anteriormente, a Interface para o MMS apresenta novos serviços que são os seguintes: "VMD-ACTIVATE", "VMD-DEACTIVATE", "VMD-RESET", "REJECT-INDICATION" e "CANCEL". Os serviços de "RELEASE" foram substituídos pelos serviços de "CONCLUDE".

A função "VMD-ACTIVATE" é necessária se o usuário MMS atua como servidor dos serviços de Acesso à Variáveis ou de Gerenciamento de Semáforos. Nestes serviços, algumas das funções necessitam da definição de variáveis e semáforos locais que estarão associados ao VMD, cujo identificador é retornado pela função "VMD-ACTIVATE". É também através desta função que o usuário consegue registrar uma ou mais AE's à um VMD. Caso o usuário de uma Invocação de AE seja servidor dos objetos Váriavel Nomeada, Lista de Variáveis ou Semáforos, ele deverá requisitar o serviço de ativação do VMD, antes da emissão subsequente da função de ativação de AE.

Tão logo o usuário queira invalidar uma identificação de VMD, ele deverá emitir a função "VMD-DEACTIVATE", que só terá êxito se os objetos associados a este VMD estiverem desativados.

Caso o usuário queira realizar uma limpeza drástica de todos os objetos associados ao VMD, ele deverá realizar uma chamada à função "VMD-RESET".

A função "CONCLUDE", como o próprio nome sugere, é utilizada para concluir um contexto MMS, liberando a associação. Contudo, o "CONNECTION-ID" continuará sendo válido para as funções de "INDICATION-RECEIVE" e "ABORT". Como regra de utilização desta função, o usuário deverá esperar pelo término de todos os eventos assíncronos em uma conexão, antes da emissão de uma requisição de

"CONCLUDE". A razão por de trás desta restrição, está na própria natureza assíncrona dos componentes funcionais da interface, tornando possível que a requisição de "CONCLUDE" chegue à màquina de protocolos antes dos serviços assíncronos previamente requisitados pelo usuário. Nestes casos, as requisições dos serviços anteriores irão se perder.

Uma interface mestre pode receber indicações de "CONCLUDE" através da função "INDICATION-RECEIVE" e responder a estas indicações através da função "CONCLUDE-RESPONSE".

Quando da detecção de erros de protocolo pelo servidor MMS, este envia para a interface indicações de "REJECT" que serão recebidas através da função "INDICATION-RECEIVE".

Para cancelar uma requisição de serviço que tenha sido previamente enviada, mas que não tenha sido ainda completada, o usuário fará uma chamada à função "CANCEL", com a restrição de que existem requisições de serviço que não podem ser canceladas. Uma interface mestre receberá indicações de "CANCEL" através da função "INDICATION-RECEIVE" e as responderá através da função "CANCEL-RESPONSE".

Como resumo, e para melhor visualização dos serviços de gerenciamento do conexão, listam-se, na figura 2.14, as funções específicas do protocolo MMS, com os seus respectivos parâmetros.

NOME DA FUNÇÃO	PARÂMETROS DE ENTRADA	PARÂMETROS DE SAÍDA
MM_VMDACTIVATION	return_event_name input_dcb	inout_dcb vmd_id return_code
MM_VMDEACTIVATION	vmd_id return_event_name	inout_dcb return_code
MM_VMDRESET	vmd_id return_event_name	inout_dcb return_code
MM_CONCLUDE	connection_id return_event_name	inout_dcb return_code
MM_CORESPONSE	connection_id return_event_name input_dcb	inout_dcb return_code
MM_CANCEL	connection_id return_event_name input_dcb	inout_dcb return_code
MM_CARESPONSE	connection_id return_event_name input_dcb	inout_dcb return_code

FIG. 2.14 - FUNÇÕES DO G.CONEXÃO ESPECÍFICAS DO MMS

2.4 - CONCLUSÕES

Para que o Programa de Aplicação faça uso dos serviços MMS deverá estar a par dos conceitos definidos neste capítulo. Deverá, por exemplo, saber que o serviço "MM_ANSWER" não pode ser utilizado sem a execução anterior do serviço "LISTEN", ou que é necessário estabelecer um contexto entre os usuários comunicantes.

Note-se que foram introduzidos vários conceitos abstratos neste capítulo, e que mapeá-los para uma implementação não será uma tarefa facilitada.

3.0 - INTRODUÇÃO

Uma questão importante relacionada com a implementação de um sistema de comunicação é o seu interfaceamento (hardware e software) com o computador. Sabe-se que, devido aos vários protocolos presentes nas diversas camadas do modelo ISO/OSI, o "overhead" associado com a sua implementação não é insignificante. Entretanto, por tratar-se, aqui, de um sistema didático, optou-se pelo uso de um processador único para realizar o processamento das funções normais de comunicação e das funções normais de aplicação do sistema, em detrimento da quantidade de tempo disponível para estas últimas, especialmente em períodos de alta atividade da rede.

Na descrição da operação de um subsistema de comunicação, estruturado de acordo com o modelo de referência ISO/OSI, é essencial tratar cada camada de protocolo como uma entidade autônoma, que proporcione um conjunto definido de serviços para cada camada superior. E que, por sua vez, faça uso dos serviços fornecidos pela camada inferior para transportar as PDU's geradas para a camada similar no par remoto. Da mesma maneira, quando da implementação dos vários protocolos de camada em software, optou-se por manter a mesma abordagem, para que os benefícios da adoção de uma arquitetura em camadas não fossem perdidos.

3.1 - O SISTEMA DIDÁTICO DA APLICAÇÃO (SISDI - MAP)

A versão atual do Sistema Didático da Aplicação (SISDI-MAP) [29] baseia-se em um modelo de implementação simplificado, em que cada processo tem apenas uma instância e esses processos residem numa única máquina (IBM-PC/DOS), tendo como objetivo emular aplicações reais. A linguagem C foi utilizada como ferramenta de programação do SISDI-MAP, tanto do Software Aplicativo como do Software Básico, por ser adequada às necessidades da implementação.

O SISDI-MAP é implementado como um conjunto de tarefas (processos), uma por entidade de protocolo, com tarefas adicionais para gerenciamento e funções de temporização. A intercomunicação entre estas tarefas é coordenada por meio de um núcleo de tempo real, que é também responsável por funções de "schedule" das tarefas, pelo tratamento de interrupções (como por exemplo temporização) e, finalmente, pelas funções de sincronização (gerencimento de eventos). Os processos comunicam entre si através de filas do tipo FIFO. O uso de um novo núcleo (não comercial), justificou-se pela necessidade do fornecimento de um pacote mais customizado para comunicação, aberto a modificações e passível de adaptação de forma mais simplificada a diversos ambientes computacionais, podendo assim, evoluir junto com o software aplicativo.

Foram definidos seis processos, (figura 3.0), [29], [35], [36], [37], [38], [39], [41]:

- Processos da Interface de Operação do Usuário (Proc_OP);
- Processo da Interface de Aplicação (Proc_API);
- Processo MMS (Proc_MMS);

- Processo ACSE (Proc_ACSE);

- Processo de simulação da Camada de Apresentação (Proc_Presentation);

O processo da Interface de Operação do Usuário (Proc_OP), que trata da inicialização do sistema, cria todos os processos e todas as portas. Cada porta fornece uma identificação do processo a ela associado. Esta associação é feita de uma maneira estática, contudo, o sistema tem como característica ser reconfigurável. Para tanto, a Interface de Operação do Usuário deve criar, quando de uma nova inicialização do sistema, outros processos com suas portas respectivas. Desta forma, o sistema torna-se modular podendo-se introduzir novos protocolos, novos programas de aplicação e novas camadas inferiores. Neste último caso, o simulador será sucessivamente deslocado para as camadas de protocolos ainda não implementadas.

A comunicação entre os processos é feita utilizando-se duas estruturas de mensagens. Uma estrutura define um Bloco de Mensagens dos Protocolos da Camada de Aplicação e a outra define um Bloco de Mensagens entre a API e o AP.

Sempre que um processo deseje transferir a informação para outro processo, ele irá, primeiramente, escrever a informação em uma área de dados comum entre eles e então, enviar o apontador desta área para a fila de entrada do processo destino. O processo destino por sua vez, através do uso de funções fornecidas pelo núcleo, realiza um "polling" das filas de entrada, para retirar o apontador para a área de dados que contém a mensagem recebida. Neste processo de polling, determinam-se prioridades diferentes às filas entre os processos, estabelecendo-se uma maior prioridade para aquelas que recebem maior fluxo de mensagens. Processos que não realizam "polling" das suas portas de entrada possuem outro

mecanismo fornecido pelo núcleo para sincronização da retirada das mensagens: operações P (wait) e V (Note) sobre semáforos. Através do uso deste mecanismo de gerenciamento de eventos, os resultados de chamadas assíncronas são comunicados logo que estejam disponíveis.

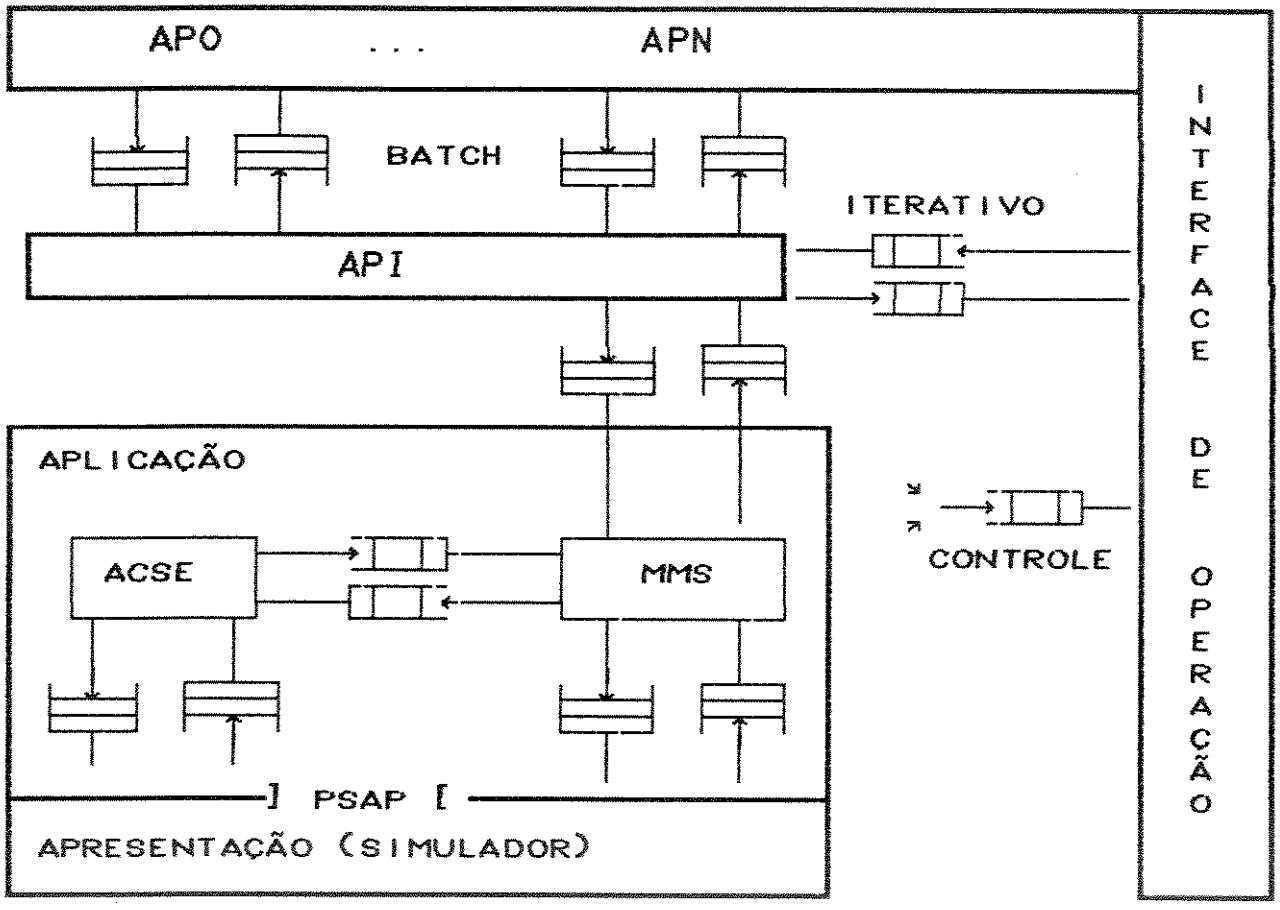


FIG 3.0 - MODELO DA IMPLEMENTAÇÃO DO SISDI-MAP

Este trabalho de dissertação engloba a implementação do Processo API e da Unidade Funcional de Gerenciamento de Conexão.

3.2 - O PROCESSO API

O processo API foi implementado de forma modular, permitindo flexibilidade no crescimento de sua funcionalidade, de acordo com as classes de conformidade de serviço necessárias para uma determinada aplicação.

A estruturação da implementação é separada em prólogos, bibliotecas e programa principal.

Os prólogos são os arquivos que devem ser incluídos por todos os procedimentos implementados. Possuem definição de tipos, declaração de variáveis globais e declaração dos protótipos das funções. Existem, na implementação, prólogos para os tipos globais a todo o SISDI-MAP, para os tipos globais à API, para as variáveis globais, para as variáveis externas e finalmente, para cada biblioteca, um prólogo distinto com a declaração dos protótipos das funções pertencentes àquela biblioteca.

Os prólogos de definição de tipos devem ser incluídos tanto pelo processo API, quanto por todos os outros procedimentos auxiliares que constituem a API.

O prólogo com a declaração das variáveis globais deve ser incluído somente pelo processo API e resulta na alocação da variável pelo compilador. Já o prólogo de declaração de variáveis externas deve ser incluído por todos os procedimentos auxiliares que fazem uso das variáveis globais, e resultará em um endereço externo ao procedimento, que será resolvido pelo ligador ("linkeditor").

Existem três tipos de bibliotecas diferentes. A primeira, visível ao usuário, é formada pelas funções "MM_AE_ACTIVATION",

"MM_CONNECT", "MM_LISTEN", etc. A outra, invísivel ao usuário, contem as funções auxiliares à implementação, como funções para manipulação de tabelas, para manipulação de filas, dentre outras. E, finalmente, a biblioteca do núcleo de tempo real que irá permitir o acesso às suas funcionalidades.

Essencialmente, o processo API manterá o controle das invocações feitas pelo Programa de Aplicação além de fornecer os serviços de sua biblioteca de funções. Estes serviços irão permitir, por exemplo, o estabelecimento dos recursos mínimos necessários à comunicação. Juntamente com as funções da gerência de conexão, a API, implementada no SISDI_MAP, põe à disposição do usuário as funções de suporte e as funções da unidade de acesso à variáveis - com as suas funções auxiliares específicas dos serviços da aplicação (MMS).

Para realizar alguns dos serviços do MMS, o usuário deve fazer chamadas à API, através das quais passa atributos em estruturas mais simples próximas da concepção do usuário (informações de mais alto nível). Com base nestas informações, a API constrói para ele, as estruturas mais complexas exigidas pelo protocolo MMS que, posteriormente, são passadas como atributos das primitivas da máquina de protocolo. Fica a cargo da API todo o controle do serviço (gerenciamento de associação entre os AP's), da alocação dos recursos necessários à realização do serviço (através da utilização de funções básicas transparentes ao AP como: controle de filas, tabelas de estado, etc.) e do retorno ao usuário da resposta do serviço solicitado (positiva ou negativa). O mesmo ocorre quando o usuário deseja interpretar a resposta recebida de um serviço solicitado. Ele utiliza as funções auxiliares da API para decompor a informação em estruturas mais simples e fáceis de serem manipuladas. A liberação do recurso alocado pela API é de responsabilidade do usuário.

A especificação da API permite assincronismo na comunicação interna, uma vez que os procedimentos para tratamento dos diversos tipos de serviços diferentes entre si, podem ser implementados como blocos funcionais da API, de forma a manterem-se independentes. Da mesma forma, a comunicação entre os blocos seria feita por intermédio de filas.

Nesta implementação adotou-se, por simplicidade, uma comunicação síncrona entre os blocos funcionais, nos quais os procedimentos são referenciados por meio de chamadas de funções.

Durante o desenrolar da descrição da implementação deve-se ter em mente que o processo API foi implementado para ser uma interface mestre, segundo determinação da especificação da interface para o MMS [17]. E como uma interface mestre, o processo API residirá no nó cliente, normalmente o controlador de célula. Neste escopo, O AP cliente irá fazer, na maioria das vezes, requisições de serviço ao AP servidor e esperar pela sua resposta. Além da resposta, esta API poderá também, receber dois tipos de indicações de serviços. Indicações de serviços não solicitados e também as indicações dos serviços confirmados cuja máquina de protocolos do MMS determine que sejam iniciados pelo servidor. Todas as indicações de serviço serão recebidas através da função "INDICATION_RECEIVE".

Para auxiliar o entendimento da implementação da API, descreve-se, a seguir, a sua estruturação na forma de um diagrama de blocos simplificado.

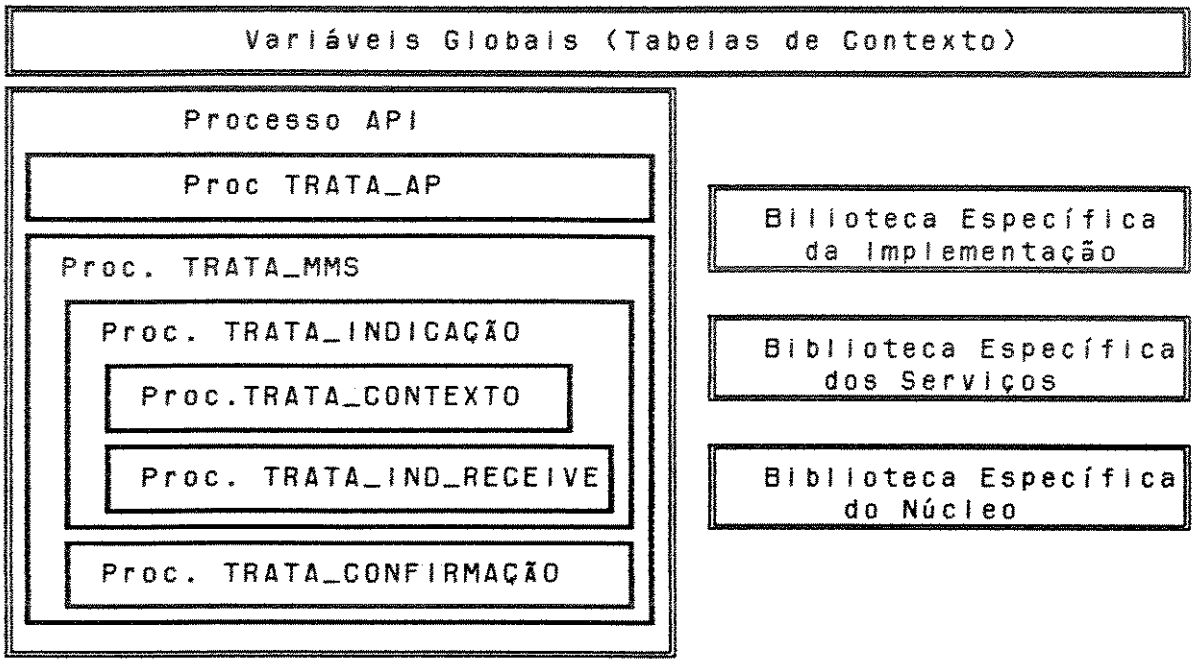


FIG. 3.1 - ESTRUTURA DA IMPLEMENTAÇÃO DO PROCESSO API

O bloco funcional chamado Biblioteca Específica do serviço é a única parte do procedimento visível ao usuário. Note-se que, não existe nenhum bloco desta estrutura de implementação que corresponda a um mapeamento direto dos blocos funcionais dos modelos da API, descritos no capítulo anterior, pois eles estão distribuídos ao longo de todos os bloco da estrutura da implementação.

Para uma descrição mais formal do processo API, residente no nó cliente, mostra-se, nas figuras subsequentes, todo o tratamento das mensagens recebidas e enviadas, na forma de um programa estruturado, com o objetivo de tornar o mais claro possível os detalhes da implementação.

```
#INCLUDE "\sisdimap\prologos\master.h"
#INCLUDE "\sisdimap\prologos\apitipos.h"
#INCLUDE "\sisdimap\prologos\varglob.h"
#INCLUDE "\sisdimap\prologos\apilib.h"
#INCLUDE "\sisdmap\prologos\auxlib.h"
VOID API( ponteiro_filas_entrada,
          ponteiro_filas_saida,
          ponteiro_fila_controle,
          ponteiro_semaforos ) {
     /* variaveis internas */
 STRUCT INTERFACE_API_AP *pt_interface_api_ap;
 STRUCT INTERFACE_API_MMS *pt_interface_api_mms;

       /* inicio do procedimento */
 FOR { /* inicializa variaveis globais */
   /* filas de entrada, filas de controle,
      filas de saida, semaforos,
      tabelas de contexto }
 FOR {    /* pooling das portas de entrada */
   IF(ESPERA_MENSAGEM( ap )); /* funcao do nucleo */
    TRATA_AP( &interface_api_ap );
   IF(ESPERA_MENSAGEM( mms )); /* funcao do nucleo */
    TRATA_MMS( &interface_api_mms ) } }
```

FIG. 3.2 - PROCESSO API

O processo API é criado quando da inicialização do SISDIMAP, recebendo como parâmetros de entrada o endereço das portas de entrada e de saída utilizadas pela API e o endereço dos semáforos utilizados para sincronização das mensagens trocadas com os programas aplicativos. Este processo tem como função inicializar as variáveis globais a toda a API e chamar, com o auxílio da função "ESPERA_MENSAGEM", fornecida pelo núcleo, procedimentos

diferentes, segundo a origem da mensagem recebida (AP ou MMS). Dentre as variáveis globais estão as tabelas de contexto que deverão ser analisadas no recebimento de qualquer mensagem pela API, para que se possa definir o processamento a ser executado. Este processo, como pode ser visto na figura 3.2, é composto de duas rotinas principais: TRATA_AP e TRATA_MMS. O primeiro deles trata das mensagens provenientes dos Programas de Aplicação e o segundo das mensagens provenientes do MMS. Note-se que, o processo de "polling" das portas de entrada estabelece prioridades iguais para o AP e para o MMS.

No procedimento TRATA_AP, descrito pela figura 3.3, mostra-se o tratamento das mensagens recebidas dos programas de aplicação existentes. Este procedimento tem como parâmetro de entrada o endereço da mensagem recebida. Nesta mensagem existe um campo indicando qual o tipo de serviço que esta sendo requisitado (tipo_serviço). Através da análise deste campo, faz_se a chamada ao procedimento da Biblioteca Específica de Serviços, correspondente ao serviço requisitado ("AE_ACTIVATION", "VMD_ACTIVATION", "CONNECT", "VREAD", etc.). Todos estes procedimentos retornam um código dizendo se existe ou não erro na passagem dos parâmetros. Após esta etapa analisa-se o campo RETURN_EVENT_NAME (nome do evento associado à chamada da função da biblioteca) para saber se o serviço deve ser executado síncrona ou assincronamente. Tanto síncrona como assincronamente o AP envia a mensagem para a API e espera sobre um semáforo específico. No tocante à API, se o serviço for executado assincronamente retorna-se uma mensagem e sinaliza-se sobre o semáforo do AP respectivo. Em procedimentos normais, esta mensagem indica ao AP que o serviço foi requisitado ao MMS. No caso de erro, esta mensagem indica ao AP o código específico do problema ocorrido. Os serviços de envio de mensagem e de sinalização do semáforo são realizados através das funções Envia_mensagem (ae_label) e Note (ae_label), respectivamente, que por sua vez utilizam-se da Biblioteca Específica do Núcleo.

```
VOID TRATA_AP( ponteiro_interface_api_ap) {
SWITCH(tipo_servico) {  /* servicos fornecidos ao AP */
  CASE mms_ae_activation:
     return_code = MM_AE_ACTIVATION( ) ;
     função requer primitiva de retorno = FALSE;
  CASE mms_ae_deactivation:
     return_code = MM_AE_DEACTIVATION( ) ;
     função requer primitiva de retorno = FALSE;
  CASE mms_connect:
     return_code = MM_CONNECT( ) ;
     função requer primitiva de retorno = TRUE;
           .
           .
           .
  CASE mms_vread:
     return_code = MM_VREAD( ) ;
     função requer primitiva de retorno = TRUE;
  CASE mms_vwrite:
     return_code = MM_VWRITE( ) ;   }
     função requer primitiva de retorno = TRUE;

IF  {  /* condicao de erro  ou
          funcão nao requer primitiva de retorno */
  RESETA_RETURN_EVENT_NAME();
  ENVIA_MENSAGEM( ap, ponteiro_interface_api_ap);
  SINALIZA( semáforo ); }
ELSE {
   IF {   /* pedido assincrono */
  /* confirmacao de entrega da mensagem ao mms */
     ENVIA_MENSAGEM( ap ) ;
     NOTE( ae_label )   /* libera semaforo do ap */
       } }
```

FIG 3.3 - PROCEDIMENTO TRATA_AP.

No procedimento TRATA_MMS, descrito pela figura 3.4, mostram-se as mensagens recebidas do nó servidor. Este procedimento, tal qual o procedimento anterior, recebe como parâmetro de entrada o endereço da mensagem . Nesta mensagem existe um campo dizendo qual o tipo de primitiva recebida (tipo_primitiva). Novamente, a análise de um campo da mensagem desencadeará um processamneto particular (recepção de primitivas de confirmação - positivas ou negativas - ou de primitivas de indicação). Após o processamento da informação testa-se se a resposta é de um serviço síncrono ou assíncrono. Se a resposta for de um serviço síncrono, a mensagem deverá ser enviada para a porta associada aquele AP e, posteriormente, dever-se-á fazer a notificação do término do serviço, liberando o AP para a recepção e processamento desta mensagem. Caso contrário, é necessário testar se o usuário já requisitou o serviço "WAIT", que permite ao AP esperar pelo evento de término de serviços assíncronos . Se o AP já requisitou o serviço "WAIT", enviam-se as mensagens de término de "WAIT" e do serviço cuja primitiva de confirmação está disponível, seguidas de uma sinalização do semáforo respectivo. Se não, o serviço é colocado na fila de serviços prontos.

A inserção dos serviços assíncronos, prontos, em uma fila de espera de pedido de "WAIT", faz-se necessária para evitar que a resposta da requisição do serviço chegue em um momento impróprio para o Programa de Aplicação.

Torna-se necessário mostrar também os procedimentos utilizados para o recebimento de primitivas de indicação. Uma vez que, sendo implementada como uma interface MMS "mestre", a API poderá receber indicações de primitivas de serviços de gerenciamento de conexão, serviços confirmados (exclusivamente os serviços de acesso a variáveis, os serviços de gerenciamento de semáforo e os serviços cuja máquina de protocolos do MMS define que sejam iniciados pelo servidor), ou finalmente de um serviço não confirmado.

O caso particular do tratamento de indicações deverá ser analisado com detalhe, pois é neste ponto que se implementará a "inteligência" do IFA.

```
VOID TRATA_MMS(ponteiro_interface_api_mms)  {
  SWITCH( tipo_primitiva )  {
    CASE confirm_pos:
    CASE confirm_neg:
      ponteiro_interface_api_ap =
        TRATA_CONFIRMACAO(ponteiro_interface_api_mms) ;
    CASE indication:
      ponteiro_interface_api_ap =
       TRATA_INDICATION(ponteiro_interface_api_mms); }
  IF{/* ponteiro_interface_api_ap  valido */
   IF {          /* pedido assincrono */
    IF {      /* existe pedido de wait */
 /* retira pedido de wait da fila de servicos pendentes */
     ENVIA_MENSAGEM( ap ); /* envia wait */
     ENVIA_MENSAGEM( ap ); /* envia servico */
     NOTE( ae_label ) :    /* libera semaforo */ }
    ELSE {
          /* insere pedido na fila de serv prontos */ }}
  ELSE {              /* servico sincrono */
     ENVIA_MENSAGEM( ap ); /* envia servico */
     NOTE( ae_label ) ; } } }   /* libera semaforo */
```

FIG. 3.4 - PROCEDIMENTO TRATA_MMS.

No procedimento TRATA_INDICAÇÃO, descrito pela figura 3.5, mostra-se o processamento executado quando do recebimento de uma primitiva de indicação. Neste procedimento, como nos procedimentos anteriores, chamam-se novos procedimentos a partir da análise do tipo de serviço.

```
STRUCT INTERFACE_API_AP *TRATA_INDICAÇÃO(
          ponteiro_interface_api_mms) {
  SWITCH( servico ) {
    CASE mms_serv_contexto    :
      ponteiro_interface_api_ap =
      TRATA_MMS_CONT_IND(ponteiro_interface_api_mms) ;
    CASE ELSE :
      ponteiro_interface_api_mms =
     TRATA_MMS_IND_RECEIVE(ponteiro_interface_api_mms) }
  RETURN(ponteiro_interface_api_ap); }
```

FIG. 3.5 - PROCEDIMENTO TRATA_INDICAÇÃO

No procedimento TRATA_MMS_CONT_IND, figura 3.6, pode-se ver que, a menos do servico de "CONNECT" ("INITIATE" para o MMS), todos os outros serviços têm sua indicação tratada pela rotina TRATA_MMS_IND_RECEIVE. A diferença está associada à "inteligência" do recebimento de indicações, pois o procedimento para a recepção de indicações de "INITIATE" é diferente do procediemnto para a recepção de todas as outras indicações do gerenciamento de conexão. A rigor, para que uma indicação seja tratada pelo usuário, este deveria ter requisitado , a priori, um serviço específico : "LISTEN" no caso de uma indicação de "CONNECT" e "INDICATION_RECEIVE" no caso de todas as outras indicações.

```
STRUCT INTERFACE_API_AP *TRATA_MMS_CONT_IND(
        ponteiro_interface_api_mms) {
 SWITCH( servico ) {
   CASE INITIATE :
     ponteiro_interface_api_ap =
     IFA_MMS_LISTEN( ponteiro_interface_api_mms) ;
   CASE MMS_CONCLUDE :
   CASE MMS_ABORT :
   CASE MMS_REJECT :
   CASE MMS_CANCEL :
     ponteiro_interface_api_mms =
    TRATA_MMS_IND_RECEIVE(ponteiro_interface_api_mms) }
   RETURN(ponteiro_interface_api_ap); }
```

FIG. 3.6 - PROCEDIMENTO TRATA_MMS_CONT_IND.

No procedimento TRATA_MMS_IND_RECEIVE, descrito pela figura 3.7, existem duas situações a serem tratados quando do recebimento de primitivas de indicação. Estas duas situações se diferenciam caso o usuário tenha solicitado , ou não, o serviço de "INDICATION_RECEIVE" nesta conexão. Caso sim, dependendo do tipo de serviço, a indicação recebida poderá ou ter uma resposta automática, fornecida pelo próprio provedor de serviços da API, ou poderá ser entregue ao usuário para que ele se encarregue da resposta. Caso não, se a indicação for do serviço de abort, a conexão será setada como abortada para que no momento do envio de novas primitivas pelo PSP, este tenha conhecimento da perda da conexão e retorne um código de erro ao usuário, impedindo novas tentativas de transmissão nesta conexão.

```
STRUCT  INTERFACE_API_AP *TRATA_MMS_IND_RECEIVE(
            ponteiro_interface_api_mms){
  IF(/*existe pedido de indication_receive na conexao*/)
   SWITCH( servico ) {
    CASE MMS_GERENCIAMENTO_SEMAFORO :
    CASE MMS_ACESSO_VARIAVEIS :
     IF(/* a variável ou o semáforo
            está associada a um VMD */)
            * resposta  automática */
      MMS_IP( ponteiro_interface_api_mms)
     ELSE { /* resp. a cargo do AP */
      ponteiro_interface_api_ap =
           IFA_MMS_IND_RECEIVE(
              ponteiro_interface_api_mms)
      RETORNA( ponteiro_interface_api_ap ) ;}}}
  ELSE {
     /* seta conexão como abortada */
     RETORNA( NULL ); } }
```

FIG. 3.7 - PROCEDIMENTO TRATA_MMS_IND_RECEIVE

3.3 - A UNIDADE FUNCIONAL DE GERENCIAMENTO DE CONEXÃO

A maior parte das funções da unidade de gerenciamento de conexão da API são divididas em duas partes: uma geral a toda a API e uma específica do protocolo, no nosso caso o MMS. Contudo existem funções que são locais à API (não resultam em uma PDU que será transmitida pela rede). Estas funções somente estão especificadas na parte geral do documento. A funções AE_ACTIVATION e AE_DEACTIVATION são exemplos de função local.

As funções da unidade de gerenciamento de conexão, juntamente com as funções pertencentes às demais unidades irão constituir a biblioteca da API. O primeiro passo para a implementação destas funções foi a construção de um algoritmo básico, baseado no método de implementação dos modelos funcionais da API apresentado em [30], [38]. Como exemplo de aplicação deste método, o algoritmo básico para o serviço de abertura de conexão ("MM_CONNECT") toma a seguinte estrutura:

1- Verifica parâmetros universais;
2- Verifica parâmetros específicos;
3- Transfere parâmetros do MMS para o HLSP_MMS;

} **Biblioteca_GC**

4- Envia Requisição de Initiate para o CSP

} **HLSP_MMS**

5 - Formata a requisição de Initiate com os parametros recebidos ou com "default";
6 - Associa nº de versão com connect_id;
7 - Retorna ao HLSP_MMS;

} **CSP_MMS**

8 - Recebe Initiate formatado;
9 - Inclui Initiate no campo de dados do A_ASSOCIATE;
10 - Retorna ao Gerenciamento de Conexão;
} HLSP_MMS

11 - Recebe primitiva de A_ASSOCIATE_REQ;
12 - Envia primitiva para o CSP e espera pelo retorno;
} HLSP_GC

13 - Verifica se o nº de associações por invocação de AE foi excedido;
14 - Passa um pedido de A_ASSOCIATE para o PSP;
15 - Espera pela resposta do PSP_GC;
} CSP_GC

16 - Envia Primitiva A_ASSOCIATE;
17 - Sinaliza CSP_GC;
} PSP_GC

18 - Solicita Confirmação do PSP_GC;
19 - Espera pela resposta do PSP_GC;
} CSP_GC

20 - Recebe Primitiva de resposta de A_ASSOCIATE;
21 - Passa a Primitiva para o CSP_GC;
22 - Informa Interface MMS da Confirmção;
} PSP_GC

23 - Recebe Confirmação de INITIATE do Gerenciamento de Conexão;
24 - Passa Primitiva Confirmação de INITIATE para o CSP_MMS;
} HLSP_MMS

25 - Recebe Primitiva de Confirmação de INITIATE e inicializa parâmetros da Interface MMS em função dos parâmetros da primitiva;
26 - Retorna ao HLSP_MMS;
} CSP_MMS

27 - Retorna ao Gerenciamento de Conexão;
} HLSP_MMS

28 - Coloca Informação de Saída na Área de Saída do usuário;
29 - Se a conexão é estabelecida com sucesso incrementa o nº de associações por invocação de Entidade de Aplicação;
30 - Retorna ao HLSP_GC;
} CSP_GC

31 - Se a informação de retorno indica sucesso, estabelece o Connection_id e sinaliza o ReturnEventName;

32 - Se não indica sucesso ou o parâmetro NumberOfRetry é nulo indica erro e sinaliza ReturnEventName;

33 - Se o ReturnEventName não foi sinalizado, RetryCounter ← NumberOfRetry e repete os passos seguintes até o ReturnEventName ser sinalizado:

- Espera DelayBetweenRetry;
- Envia pedido de A_ASSOCIATE.REQ para o CSP e Obtém informação de retorno;
- Se informação de retorno indica sucesso estabelece Connection_id;
- Decrementa RetryCounter;
- SE (informação de retorno não indica sucesso E RetryCounter = 0) OU condição de erro

  ENTÃO indica erro e sinaliza ReturnEventName;

} HLSP_GC

No algoritmo descrito, os blocos funcionais da API são distinguidos através do sufixo, caso sejam exclusivos do MMS (_MMS) ou caso sejam da especificação geral da API (_GC).

No capítulo anterior, descreve-se o modelo da API através dos blocos funcionais IFA, HLSP, CSP, PSP, LIB, etc... Note-se que, estes blocos funcionais estão distribuídos ao longo do algoritmo acima.

A partir deste ponto já pode se pensar na implementação do algoritmo real na linguagem C.

Todas as funções possuem como passo inicial a checagem de seus parâmetros de entrada. Caso algum erro seja detectado, a função é terminada e um código de erro é retornado ao processo API, que por

sua vez, tratará de enviá-lo ao usuário requisitante. Alguns parâmetros de contexto tais como: "RETURN_EVENT_NAME", "CONNECTION_ID" e "INVOKE_ID" possuem rotinas próprias para a sua checagem de tal forma que, caso o usuário requisite um valor de instânciação destes parâmetros que esteja em uso pela máquina de protocolos da API, um código de erro específico será retornado, e como consequência, ele terá seu serviço terminado.

Um outro procedimento comum à maioria das funções de biblioteca da API é a alocação de recursos (memória), dinamicamente. Existe na implementação um pool de recursos do usuário que é, na realidade, uma lista encadeada de apontadores para blocos de memória disponíveis para a aplicação. Cada conexão tem um limite de recursos disponíveis que são controlados pela máquina de estados da API. A responsabilidade da liberação destes recursos em procedimentos normais é toda do usuário com exceção do recebimento de uma primitiva de "ABORT" pela API que, neste caso particular, tratará de fazer uma limpeza drástica, liberando todos os recursos associados àquela conexão.

A função "MM_AE_ACTIVATION" possui uma característica de implementação importante que é a de associar o endereço da porta de entrada do AP, por ela recebido, ao "AE_LABEL", por ela gerado, que, por sua vez, está associado ao "CONNECTION_ID" . Estas associações são importantes porque toda primitiva de confirmação recebida do MMS possui sempre o "CONNECTION_ID", mas não possui nem o endereço da porta de entrada do AP, nem o "AE_LABEL". Uma outra característica de implementação importante é que, quando da requisição desta função, o AP poderá inicializar o parâmetro "RETURN_EVENT_NAME" com um valor aleatório. Mas a partir do recebimento de sua resposta, o AP já possui um "AE_LABEL" e, portanto, só poderá usar os "RETURN_EVENT_NAME" pertencentes à sua faixa.

A função "MM_AE_DEACTIVATION" só terá sucesso na desativação do "AE_LABEL" se não existirem nem serviços pendentes, nem pedidos de "LISTEN", nas conexões pertencentes àquele AP.

A função "MM_ABORT" realiza, para a máquina de protocolos da API, procedimentos de limpeza dos contextos ligados à conexão abortada. Estes procedimentos incluem: a retirada de mensagens associadas àquela conexão das filas internas da API, a liberação dos recursos associados e a alteração de determinadas tabelas de contexto.

As Funções que requerem primitivas de retorno, e que são utilizadas pelo usuário através de uma chamada única, se dividem na API em duas funções. Uma que realiza todos os passos necessários para o envio e outra que realiza os passos necessários para o recebimento de primitivas para/do MMS. Exemplos desta funções são "MM_CONNECT", "MM_LISTEN", "MM_IRECEIVE", etc. A função "MM_CONNECT", por exemplo, se divide na implementação da API em duas funções, cahamdas "API_MMS_CONNECT_ENVIA" e "API_MMS_CONNECT_RECEBE".

As funções que são diretamente responsáveis pela negociação do contexo de aplicação ("MM_CONNECT", "MM_LISTEN", "MM_ANSWER", "MM_CONCLUDE" e "MM_ABORT"), são responsáveis pela instanciação das tabelas de contexto da máquina de estado da API. Estas funções necessitam também de várias funções adicionais para mapeamento de primitivas e teste de parâmetros. A função "MM_ANSWER" necessitou da implementação de uma função auxiliar para teste dos CBB's negociados, pois se algumas das classes de conformidade enviadas como resposta forem maiores do que a requeridas pelo AP este serviço retornará um código de erro específico. Na implementação da função "MM_LISTEN" assume-se que o provedor MMS irá rejeitar, automaticamente, indicações de "ASSOCIATE" com números de versão incompatíveis. Por este motivo não foi necessário testar esta condição de erro, pois a API nunca a irá receber.

Uma função extremamente importante dentro da implementação é a função "MM_INDICATION_RECEIVE" que é responsável pelo recebimento de indicações. Esta função praticamente não instância tabelas de contexto nem manipula filas internas: seu trabalho principal está no mapeamento das respostas, uma vez que, de acordo com a especificação da API para o MMS, ela pode receber uma gama maior de primitivas.

Além das funções de gerenciamento de conexão foi necessário realizar a implementação das funções de suporte "WAIT", "NOTE" e "GET_PRINTABLE_ERROR".

As duas primeiras foram necessárias para a realização de pedidos assíncronos e a segunda para a tradução dos códigos de erro para o usuário.

A função "WAIT" permite que o resultado de chamadas assíncronas sejam comunicadas ao usuário. A mensagem de requisição desta função é colocada na fila de espera, se não houver nemhum serviço pronto associado ao AP que originou a chamada. Caso contrário, a mensagem de resposta do serviço é enviado para o AP, seguida da sinalização deste evento, através da função "NOTE".

3.4 - A ESTRUTURA DE DADOS DA IMPLEMENTAÇÃO

Embora o Sistema Didático seja estruturado em concordância com o Modelo de Referência ISO/OSI [1] - onde cada protocolo é tratado como uma entidade autônoma que fornece um conjunto definido de serviços para a entidade imediatamente superior na hierarquia do sistema e que, por conseguinte, recebe serviços da a entidade imediatamente inferior, permitindo que somente sejam acrescentados/ retirados cabeçalhos de controle das mensagens trocadas entre as camadas - esta sistemática não foi possível de ser implementada na definição da estrutura de dados entre AP, API e MMS, uma vez que a API, por não ser protocolo, possui como característica realizar mapeamento de parâmetros ao invés de acrescentar/retirar campos de controle. Desta forma, criou-se duas estruturas de dados de interfaces distintas, uma com o AP e outra com o MMS, onde na primeira existem estruturas de dados específicas para a requisição dos serviços da API, AP's e na segunda, da mesma forma, existem estruturas de dados específicas para as primitivas trocadas com o MMS. No início destas estruturas existem campos contendo parametros globais a todas as mensagens de uma determinada interface. Estes campos são seguidos de campos adicionais para todos os parâmetros possíveis, associados com o conjunto completo de primitivas de serviços da camada, parâmetros estes que podem ser em forma de uma lista de tipos de dados abstratos.

A título de exemplo, mostra-se na figura 3.8 uma parte da definição, em Linguagem C, da estrutura de dados utilizada para troca de mensagem entre a API e o AP.

```
typedef struct interface_api_ap {
    Return_event_name                   return_event_name;
    Return_code                         return_code;
    Tipo_de_funcao                      funcao;
    union {
      union Mms_gerencia_conexao        mm_contexto;
      typedef union Mms_gerencia_conexao {
        struct Ae_activation                    ae_activation;
        struct Ae_deactivation                  ae_deactivation;
        struct Mm_connect                       mm_connect;
        typedef struct Mm_connect {
          uint16                        ae_label;
          Ae_title                      called_ae_name;
          struct Connect_input          input_dcb;
          typedef struct Connect_input {
            Boolean       called_presentation_address_valid;
            P_address     called_presentation_address;
                          :
            Boolean       called_AP_title_valid;
            char          called_AP_title[ 5 ];
            Boolean       called_AP_invocation_id_valid;
            uint16        called_AP_invocation_id;
            Boolean       called_AE_invocation_id_valid;
            uint16        called_AE_invocation_id;
            Boolean       connection_resource_wait_time_valid;
            uint32        connection_resource_wait_time;
            Context       context_name;
            Mms_connect_req_info    connect_in_info;
            typedef struct {
              Boolean       prop_max_msg_valid;
              uint32        prop_max_msg_size;
              uint16        prop_max_serv_outst_calling;
              uint16        prop_max_serv_outst_called;
              Boolean       prop_data_str_nest_level_valid;
              uint8         prop_max_nesting_level;
              Horinzontal_cbb  prop_horizontal_cbb;
              Vertical_cbb  client_vert_cbb_calling;
              Vertical_cbb  server_vert_cbb_calling; } }
          struct Connect_output         inout_dcb;
          Connection_id                 connection_id; }
        struct Mm_listen                        mm_listen;
        struct Mm_slisten                       mm_slisten;
        struct Mm_answer                        mm_answer;
        struct Mm_abort                         mm_abort;
        struct Mm_ind_received                  mm_ind_received;
        struct Mm_conclude_req                  mm_conclude_req;
        struct Mm_conclude_res                  mm_conclude_res; }
      union Mms_funcoes_suporte         mm_suporte;
      union Mms_servicos_gerais         mm_servicos; } data;}
```

FIG. 3.8 - A INTERFACE API <-> AP

Com esta figura, procura-se caracterizar a complexidade da mensagem, que se apresenta com varios níveis de aninhamentos de estruturas, bem como a associação de diferentes estruturas ao mesmo espaço de dados.

Na definiçao da base de dados do Processo API do SISD-MAP foi necessário realizar o casamento entre os parâmetros definidos na especificação da API com os parâmetros definidos na especificação do MMS e do ACSE, pois as versões disponíveis dos documentos eram diferentes. O documento da API estava em concordância com o draft 5 do protocolo MMS e como já possuíamos a versão do MMS correspondente ao draft 6 optamos por adequar a especificação da API ao draft 6.

Tratando-se de um sistema didático, procurou-se limitar um pouco o escopo dos parâmetros. Assim cada invocação de AE terá um número máximo de RETURN_EVENT_NAME e de conexões a ela associados. Cada conexão terá por sua vez um número máximo de serviços pendentes a ela associado. Através da figura 3.9, mostra-se graficamente estas limitações.

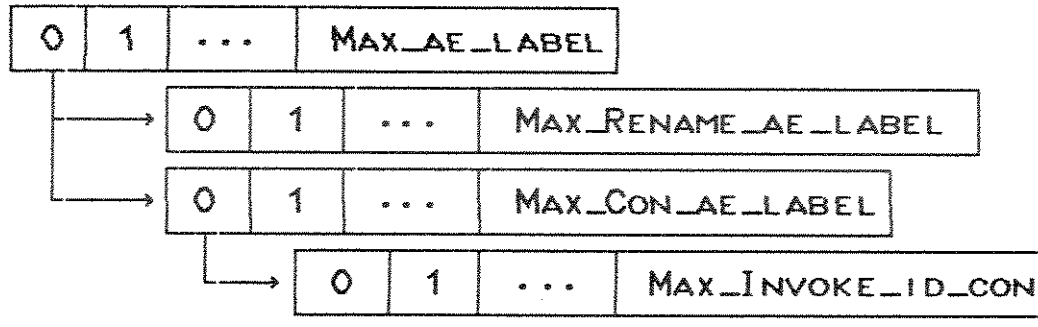


FIG 3.9 - LIMITAÇÕES PARAMETRIZADAS DA IMPLEMENTAÇÃO

Para que a API processe cada primitiva (request, indication, response ou confirmation) da forma correta é necessário que a Interface retenha o apontador para o bloco de dados de cada uma

das primitivas ainda não terminadas. Essa abordagem deve ter um caráter dinâmico e é melhor implementada através do uso de filas. Na implementação do SISDI-MAP o processo API manipula três filas internas que são: a fila de mensagens entre a API e o AP - nesta fila se encontram primitivas à espera de uma confirmação e primitivas assíncronas à espera de uma requisição de WAIT - , a fila de mensagens entre a API e o MMS - nesta fila se encontram as mensagens a serem retransmitidas por problemas de erro - e finalmente a fila de recursos (memória) - esta fila é utilizada quando da necessidade da alocação dinâmica de DCB'S pelo provedor de serviços da interface.

Semelhantemente à estratégia de implementação de máquinas de protocolos a Interface de Aplicação possui tabelas de estado, para certos parâmetros, que indicam o estado corrente da interface. Quando do recebimento de uma primitiva, estas tabelas de estado de eventos são analizadas para determinar o processamento a ser executado. Por problemas de memória, todas as tabelas armazenadas no SISDI-MAP são tabelas de bits. E tendo como objetivo facilitar futuras expansões do sistema todas as tabelas tiveram suas dimensões definidas por meio de "defines" da linguagem C.

Para que o processo API possa monitorar as invocações de AE, as associações, os serviços confirmados - que estão esperando pelo seu término - e as invocações de AE - que estão aptas a receber pedidos de associação definem-se na implementação as seguintes tabelas:

- **ae_label_map:** contém o estado corrente dos AE_LABEL's válidos. Esta tabela é checada por todos os serviços fornecidos pela API e setada somente pela função responsável pela ativação da AE. Como descrito anteriormente, define-se no sistema didático uma faixa particular de CONNECTION_ID para cada AE_LABEL.

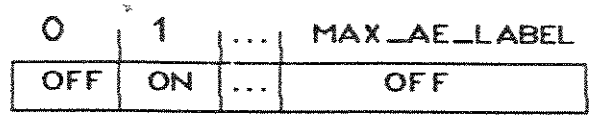


FIG. 3.10 - AE_LABEL_MAP

- **connection_id_map:** armazena o estado corrente dos números de conexão conhecidos pela API. Esta tabela pode ser setada diretamente pelo processo API no recebimento de uma primitiva de "ABORT" ou pelas seguintes funções de Gerenciamento de Conexão: "ANSWER", "CONNECT" e "LISTEN".

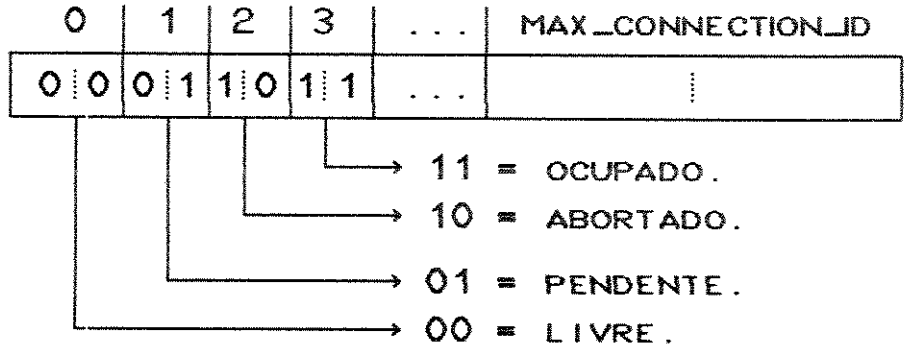


FIG 3.11 - CONNECTION_ID_MAP.

- **num_connect_mms:** Contém o número de conexão conhecido pelo MMS. Necessitou-se definir esta tabela porque, como dito anteriormente, o SISDI-MAP reside numa única estação. Desta forma, definem-se mais duas tabelas :

  - map_num_connect_api_mms;
  - map_numconnect_mms_api.

O mapeamento realizado pode ser visualizado através da figura 3.12 abaixo.

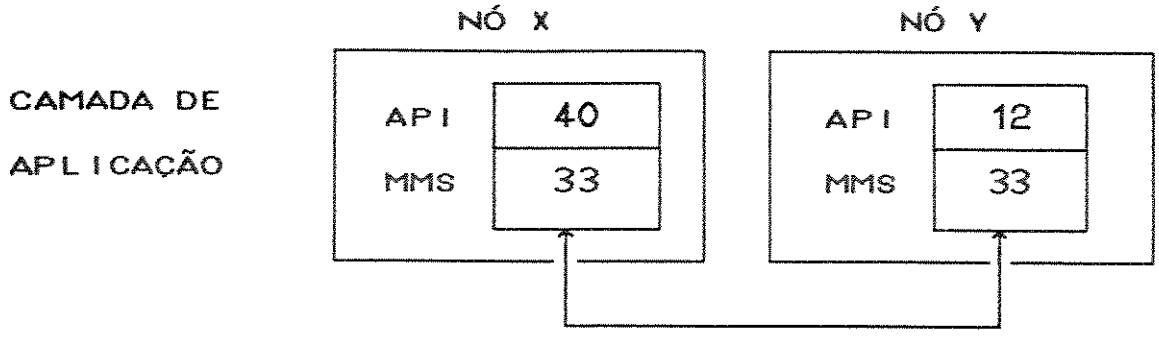


FIG 3.12 - MAPEAMENTO DOS NUMEROS DE CONEXÃO

- **invoke_id_map:** Esta tabela, descrita pela figura 3.13, é usada por todos os serviços confirmados, com exceção daqueles pertencentes ao gerenciamento de conexão e contém o estado corrente do número do serviço pendente (INVOKE_ID) dentro da conexão. Este número é único dentro do sistema. Seu valor é calculado através da seguinte expressão :

$$max\_invoke\_id * connection\_id < invoke\_id < max\_invoke\_id * (connection\_id - 1).$$

0	1	...	MAX_INVOKE_ID
OFF	ON	...	OFF

FIG. 3.13 - INVOKE_ID_MAP

- **invoke_ae_listen_map:** contém os ae_label's do sistema que estão aptos a receber pedidos de abertura de conexão. Esta tabela é atualizada pela função "LISTEN".

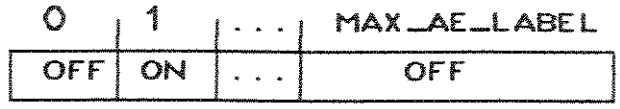


FIG. 3.14 - AE_LABEL_LISTEN_MAP

A API possui também, tabelas para controlar os nomes dos eventos associados às chamadas assíncronas de funções e para monitorar as invocações de AE que estão esperando pelo término de serviços remotos.

- **return_event_name_map:** Esta tabela é manipulada por todos os serviços fornecidos pela "API" e contém o estado corrente dos "RETURN_EVENT_NAME" utilizados pelo sistema. No SISDI-MAP cada "AP" possui uma faixa de RETUR_EVENTE_NAME_DISTINTA:

  AP0 ⟶ 0 < REN < 128;
  AP1 ⟶ 200 < REN < 328 ...

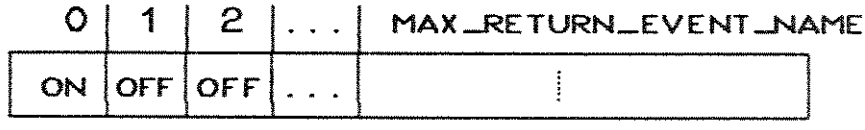


FIG. 3.15 - RETURN_EVENT_NAME_MAP

- **wait_map:** Esta tabela indica se o "AP" está, ou não, no estado de "WAIT" e foi definida por motivos de sincronização entre os processos.

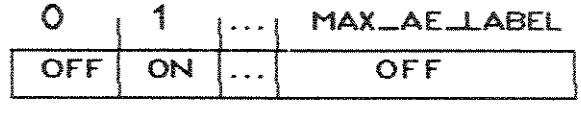


FIG. 3.16-WAIT MAP

A API possui uma tabela para indicar se o AP está apto, ou não, para o recebimento de uma indicação em uma determinada conexão.

- **indication_received_map:** indica se o AP está apto, ou não, para o recebimento de uma indicação naquela conexão. Esta tabela é setada pela função INDICATION_RECEIVED e utilizada pelo processo API quando do recebimento de inidicações como mostrado anteriormente na figura 3.11.

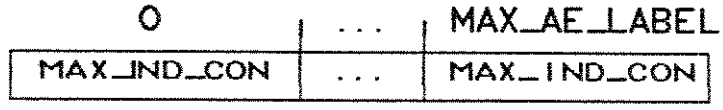


FIG. 3.17 - INDICATION_RECEIVE_MAP

E finalmente, a API possui tabelas para indicar os recursos (memória), que estão associados a uma determinada conexão.

- **alloc_resource_map:** esta tabela é setada pelo processo API durante a troca de mensagens, sendo específica da conexão. É resetada quando do recebimento de uma primitiva de "ABORT".

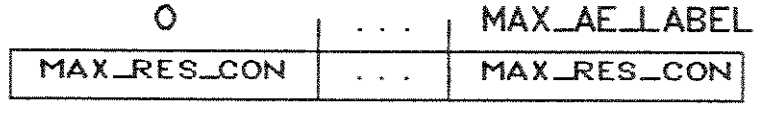


FIG. 3.18 - ALLOC_RESOURCE_MAP

3.5 - CONCLUSÕES

Além do esforço de entendimento da especificação, necessita-se de um esforço adicional para se passar da especificação para a implementação, pois com exceção das rotinas da Biblioteca de Serviços, todo o restante é considerado, no documento, como uma questão de implementação local.

Um outro ponto importante a destacar é que a especificação só trata do cliente, deixando a problemática do servidor em aberto.

4 - INTRODUÇÃO

A pressão econômica por melhorias na qualidade, produtividade e eficiência está levando à uma grande utilização de equipamentos programáveis no setor de automação industrial. Estes equipamentos programáveis tais como controladores de robô (CNR'S), controladores numéricos (CNC'S), controladores programáveis (CLP'S) e sistemas de visão estão sendo interligados em unidades de manufatura, chamadas de células da manufatura. As vantagens da utilização destes sistemas estão na sua flexibilidade, que permite a reprogramação da célula para a fabricação de diferentes produtos, e assim, a viabilização econômica dos investimentos a se realizar.

Uma Célula Flexível da Manufatura (FMC) é nada mais do que um conjunto totalmente automatizado de equipamentos industriais que são organizados espacialmente e interconectados apropriadamente, para facilitar a comunicação e a cooperação entre os componentes do sistema, durante a execução das operações necessárias para a realização de uma tarefa particular.

Os equipamentos da célula se comunicam através da troca de mensagens utilizando o protocolo MMS, especialmente projetado para tal. O usuário, residente no nó cliente e que queira utilizar as funcionalidades dos equipamentos desta célula, terá acesso ao sistema de comunicação através da API, ficando sobre sua responsabilidade a implementação dos Programas de Aplicação (AP's) para a execução dos serviços oferecidos pela rede. Os nós servidores que residem em estações mais simples como CLP'S, CNC'S, etc, não possuirão, em geral, uma API. Neles, implementam-se os serviços de rede até a camada de aplicação e AP's mais robustos que absorvam algumas características da máquina de estados da API

(principalmente aquelas relacionadas ao recebimento de indicações), e que armazenem os objetos do VMD. Estes AP's deverão ser fornecidos pelo fabricante do equipamento programável servidor, de tal forma que ele possa responder aos serviços requistados pelo cliente (figura 4.0.).

O programador deverá ter conhecimento do ambiente que está sendo utilizado para o sucesso da sua tarefa. Em particular, ele deverá estar a par dos serviços fornecidos pela rede (quais são e como utilizá-los), e conhecer os recursos e funcionalidades do equipamento da manufatura a ser utilizado. Conhecendo o equipamento servidor, o usuário poderá mapear seus serviços em serviços MMS.

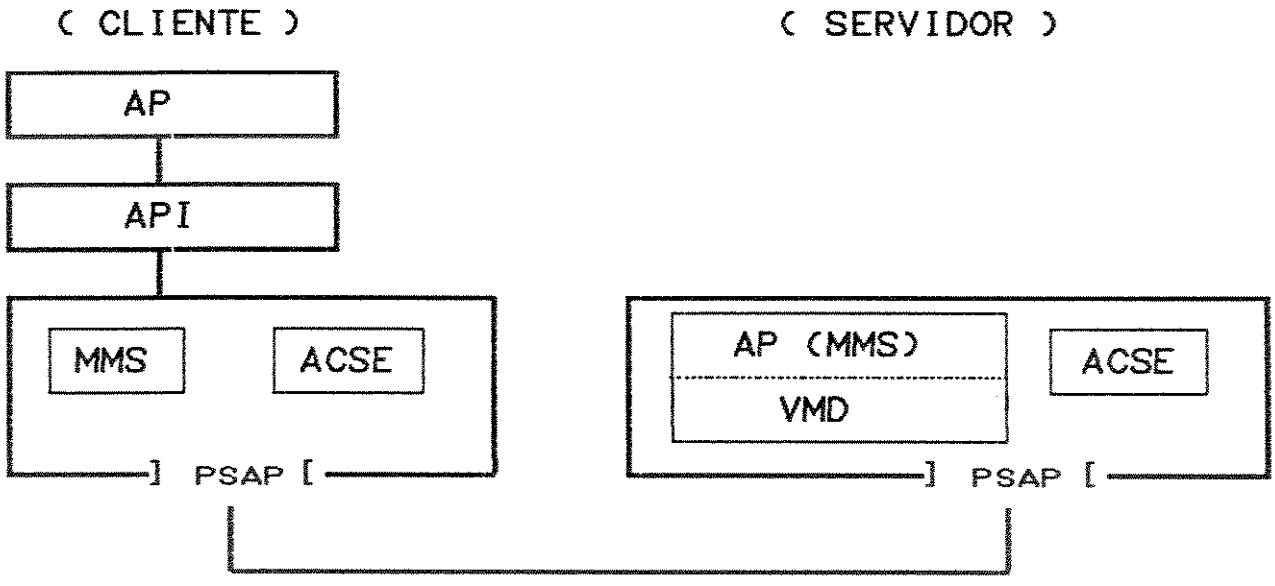


FIG. 4.0 - MODELO DE COMUNICAÇÃO ENTRE AP'S

Um MMS VMD é uma representação abstrata de um conjunto específico de recursos e funcionalidades de um equipamento real da manufatura. O VMD existirá dentro do AP do servidor MMS. Ele constitui a parte da tarefa de processamento de informação que torna disponível (para controle, monitoração ou ambos) um conjunto

de recursos e funções associadas ao equipamento real da manufatura. Assim, um AP pode definir zero ou mais VMD'S. Se nenhum VMD é definido, este não pode atuar como Servidor MMS.

Cada VMD representa um equipamento virtual da manufatura dentro de um AP, e é separado logicamente dos outros. Por exemplo, um sistema MMS que é conectado a um ambiente não MMS, contendo vários equipamentos de manufatura agregados, pode ser modelado como um único AP contendo um VMD para cada equipamento, ou vários AP'S, cada qual contendo um único VMD para cada equipamento.

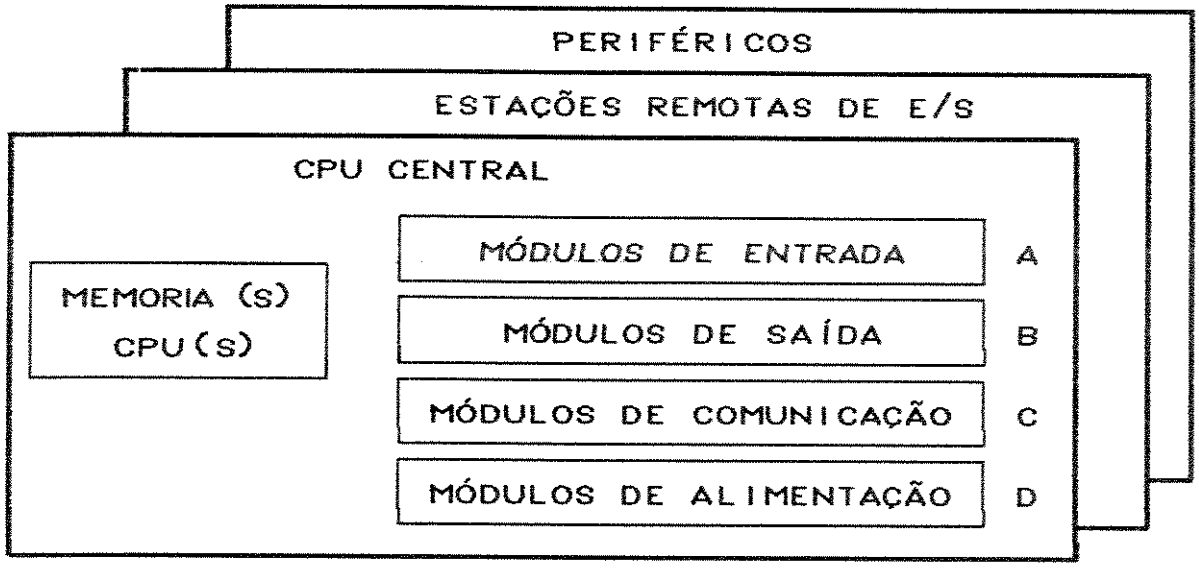
Geralmente, os recursos de um dado VMD são distintos e independentes dos recursos dos outros. Assim, quando um recurso virtual de dois ou mais VMD's é mapeado sobre o mesmo recurso físico, o(s) AP(s) deve(m) prover mecanismos de controle de acesso ao recurso físico, e torná-los acessíveis aos vários VMD's, de tal forma que os clientes dos VMD's possam coordenar seus acessos. Esta coordenação pode ser modelada através dos serviços de Gerenciamento de Semáforos definidos pelo MMS.

Se bem que o MMS seja aplicável a uma grande variedade de equipamentos industriais, a especificação do MMS por si só não é suficiente para tal tarefa, e o mapeamento de um equipamento específico deve ser feito em concordância com os Padrões Associados ao MMS ("Companion Standards"), elaborados por organizações especializadas [18], [19], [20].

Para melhor ilustrar a importância dos Padrões Associados, será apresentado um exemplo de Programa de Aplicação baseado no Padrão Associado para os controladores programáveis (CLP'S), desenvolvido pela NEMA ("National Electrical Association"). Mas antes disto, faz-se necessário descrever detalhadamente este equipamento.

4.1 - O CONTROLADOR PROGRAMÁVEL

Um controlador programável conectado a uma rede e comunicando-se como uma estação OSI em um ambiente MMS, constitui-se de uma ou mais unidades de processamento (CPU'S), memória, fonte de alimentação, I/O, interfaces de comunicação e, no mínimo, de uma interface de comunicação para o ambiente MMS, figura 4.1.



LEGENDA : A = ENTRADA DE SINAIS ANALÓGICOS E BINÁRIOS
B = SAÍDA DE SINAIS ANALÓGICOS E BINÁRIOS
C = INTERFACE SERIAL/PARELALA
D = INTERFACE COM ALIMENTAÇÃO

FIG. 4.1 - MODELO (IEC) DO CONTROLADOR PROGRAMÁVEL

Estes recursos irão constituir a Função Executiva do VMD, tornando-os visíveis para os clientes/usuários, e serão usados para descrever os elementos constituintes do Domínio. Além dos recursos, o Domínio deverá conter todo o código do programa

aplicativo e os dados necessários para a sua execução.

Um programa aplicativo para o controlador programável, representado de uma forma estruturada, compõe-se da declaração de variáveis globais, da definição de caminhos de acesso para estas variáveis, de entradas, saídas, declaração de atividades, chamadas de funções, etc...,(figura 4.2).

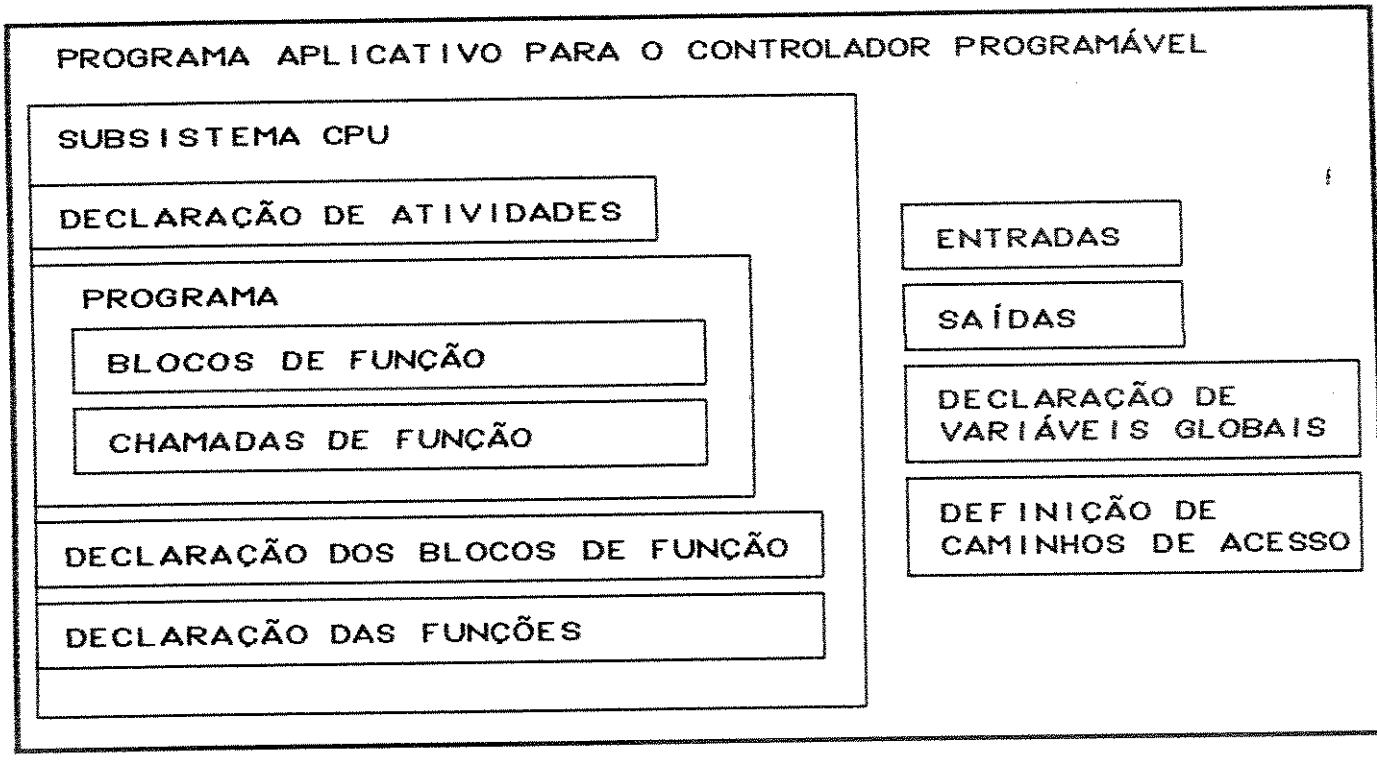


FIG.4.2 - ESTRUTURA DE UM PROGRAMA APLICATIVO DO CLP

O programa aplicativo é a ferramenta através da qual podemos realizar tarefas de controle e de monitoração em um controlador programável.

O controlador programável requer o carregamento ("download") e o descarregamento ("upload") por completo de todo o programa

aplicativo, bem como do programa aplicativo de seus subsistemas.

O controlador programável armazena e compila o programa aplicativo por meio do que chamamos "Unidades de Compilação". Estas Unidades de Compilação contêm todo o programa aplicativo ou parte dele e são mapeadas em Domínios MMS.

Os elementos da linguagem que estão sempre contidos em uma Unidade de Compilação são:

- Declaração de Atividades;
- Declaração de Blocos de Função;
- Declaração de variáveis globais;
- Declaração de caminhos de acesso.

O programa aplicativo do CLP é carregado através dos serviços de Gerenciamento de Domínio. Existem dois tipos de Domínios: Dinâmicos e Estáticos. Os Domínios Dinâmicos poderão ser criados e terem seus atributos inicializados remotamente. Já os Domínios estáticos deverão ser pré-definidos pelo fabricante e estarem distribuídos da seguinte forma :

- Um Domínio principal, por VMD, contendo todo o programa aplicativo do CLP;
- Um Domínio contendo a declaração das variáveis globais;
- Um Domínio contendo as definições dos caminhos de acesso para as variáveis do CLP;
- Um Domínio, por subsistema CPU, contendo a parte do programa de aplicação que corresponde àquele subsistema;

- Um Domínio por conjunto dos elementos da linguagem contidos em uma Unidade de Compilação.

Em correspondência a estes Domínios deverão estar associadas as seguintes Invocações de Programas:

- Uma Invocação de Programa associada ao Domínio principal;
- Uma Invocação de Programa por subsistema CPU;
- Uma Invocação de Programa por Domínio que possua uma Declaração das Atividades;

Na figura 2.4, mostra-se a asociação entre os Domínios e as Invocações de Programas, e também, a referências entre as Invocações de Programas.

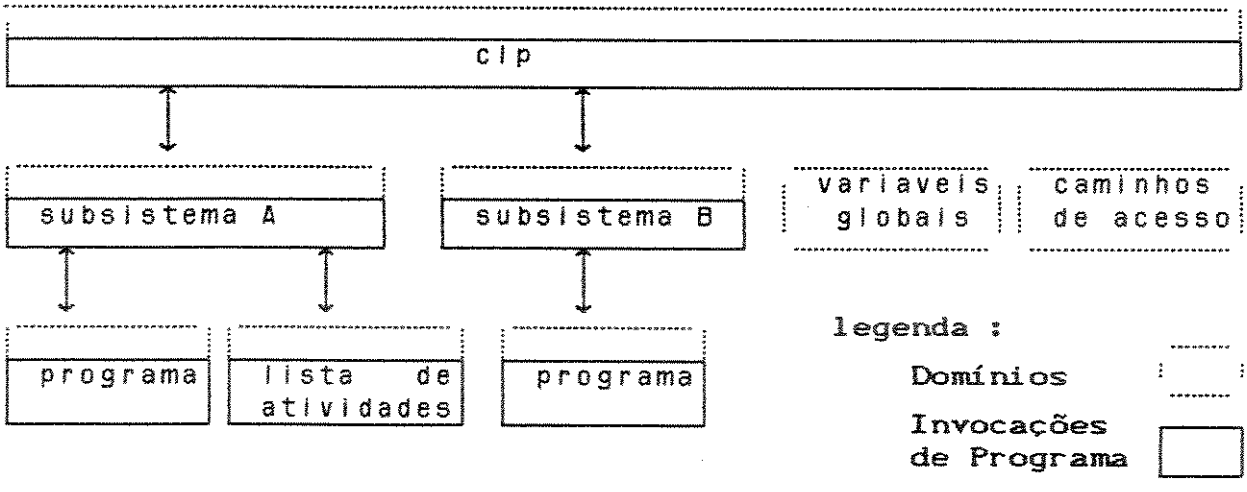


FIG 4.3 - ASSOCIAÇÕES ENTRE DOMÍNIOS E INVOCAÇÕES DE PROGRAMA

4.1.1 - FUNCIONALIDADES

Do ponto de vista de suas funcionalidades, um CLP é um equipamento utilizado em automação industrial que pode ser servidor de vários outros equipamentos em uma célula da manufatura (figura 4.4).

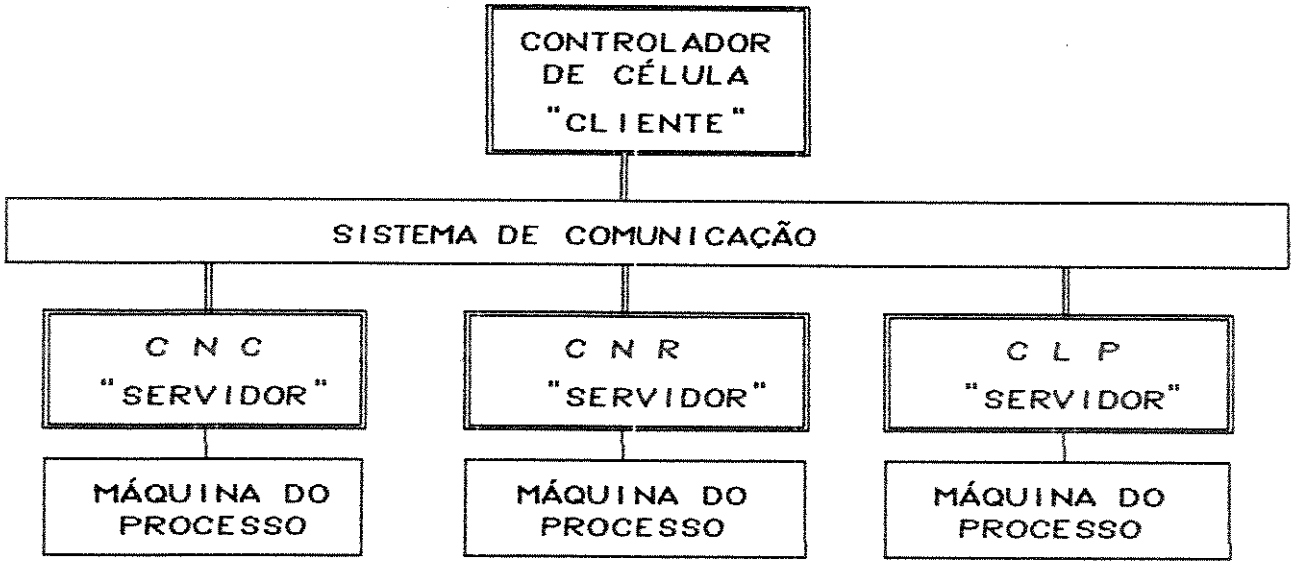


FIG. 4.4 - MODELO DE APLICAÇÃO DE UM CLP EM UMA FMC

As principais funções executadas pelo controlador programável são:

- **Verificação do estado do equipamento**: este serviço é fornecido para permitir que outros equipamentos consigam determinar se o controlador programável está, ou não, apto para executar sua função no sistema.

  Existem três tópicos básicos no Universo da verificação do estado do equipamento que são:

- Informações de estado;
- Detecção de falhas;
- Isolamento de falhas;

O CLP pode fornecer estas informações através de um pedido explícito do cliente, ou iniciar um relatório de informações não solicitadas usando um serviço específico para tal, fornecido pelo MMS.

- **Aquisição de dados**: os dados no controlador programável podem ter as mais diversas fontes (processos, máquinas, ...). Estes dados podem ser resultados de uma leitura direta ou de cálculos realizados. O cliente pode obter estes dados das mais diversas maneiras:

  - Por varredura;
  - Por intermédio de uma aquisição não solicitada (ação do servidor);
  - Por intermédio de uma aquisição programada pelo próprio cliente. Este tipo de aquisição de dados utiliza recursos da linguagem de programação próprios do controlador programável, estando implementados no programa aplicativo;
  - Por meio de uma aquisição de dados configurada. Este tipo de aquisição de dados utiliza os serviços de gerenciamento de eventos definidos no MMS.

- **Controle**: os métodos de controle fornecidos são chamados de paramétricos e por intertravamento. No paramétrico, determinam-se valores para as variáveis de controle residentes no VMD do CLP, através dos serviços de acesso a variáveis do MMS. Variáveis de "acesso direto" são pré-definidas pelo CLP e serão mapeadas no objeto Variável Não Nomeada. Já as variáveis acessadas por nome serão

mapeadas no objeto Variável Nomeada. Na definição do caminho de acesso das variáveis nomeadas, pode-se restringir o seu escopo, permitindo-se que elas sejam somente referenciadas para leitura. No segundo método de controle, por intertravamento, o cliente requisita ao servidor a execução de um serviço e fica esperando pela resposta da operação. Neste método de controle por intertravamento a troca de mensagens de dados ocorre em pontos de sincronização do programa de aplicação. Para tal finalidade este Padrão Associado define serviços adicionais ao MMS chamados "SEND" e "RECEIVE".

- **Alarmes**: No caso da ocorrência de condições predeterminadas o CLP servidor poderá enviar ao cliente mensagens de alarme não solicitadas (ação do servidor). Estas mensagens serão reenviadas até o recebimento de um "acknowlegedment" do cliente.

  O cliente poderá obter as mensagens de três modos:

  - Através de uma Detecção de alarme programada pelo próprio cliente. Este tipo de detecção de alarme utiliza recursos da linguagem de programação próprios do controlador programável, estando implementados no programa aplicativo;

  - Detecção de alarme configurada através dos serviços de gerenciamento de eventos do MMS;

  - Através de um serviço específico que informa um resumo estatístico dos alarmes ocorridos;

- **Interface com operador**: os controladores programáveis podem permitir a utilização de equipamentos de interface com o operador. Estes equipamentos são utilizados por um operador para monitorar e/ou modificar o processo controlado.

- **Gerenciamento de controle de programas**: através desta função o cliente poderá carregar ("download") a memória do controlador programável com programas aplicativos, iniciar e parar a execução destes aplicativos, bem como requisitar uma informação sobre o estado deste programa. Poderá também descarregar ("upload") a memória do servidor para armazenamento ou verificação de resultados.

4.1.2 - O MAPEAMENTO PARA O MMS

Em virtude da diversidade de tipos de controladores programáveis e da grande quantidade de serviços oferecidos, prevê-se que, em implementações específicas, só alguns subconjuntos poderão vir a ser considerados. Para tal, introduzem-se classes de conformidade diferenciadas pelas funcionalidades oferecidas. Estas classes são hierárquicas, isto é, cada classe engloba todos os serviços fornecidos pela classe inferior. Na figura 4.5, mostra-se como estas classes de conformidade estão divididas.

CLASSE	FUNÇÕES DO CONTROLADOR
1	AQUISIÇÃO DE DADOS, CONTROLE PARAMÉTRICO.
2	GERENCIAMENTO DE PROGRAMA.
3	AQUISIÇÃO DE DADOS NÃO SOLICITADA, CONTROLE POR INTERTRAVAVEMTO.
4	AQUISIÇÃO DE DADOS PROGRAMADA, SEMÁFOROS PREDEFINIDOS PELO CLP, ALARMES.
5	DEFINIÇÃO REMOTA DE SEMÁFOROS

FIG. 4.5 - RESUMO DAS CLASSES DE CONFORMIDADE DOS CLP'S

Independentemente da classe de conformidade, existe um conjunto básico de um ou mais serviços MMS que são obrigatórios para todos os controladores programáveis que operem neste ambiente. Estes serviços, utilizados para abertura de conexão, são os seguintes:

- Initiate_(modo_respondedor);
- Conclude_(modo_respondedor);
- Abort;
- Identify;

O próximo passo do programador, após se ambientar com o equipamento a ser utilizado, conhecendo seus recursos e funcionalidades, deverá ser o de mapear estas funcionalidades em serviços MMS. Estes serviços são fornecidos de acordo com a classe de conformidade do equipamento. A definição dos serviços MMS utilizados para fornecer as funcionalidades do CLP é a seguinte:

- **Verificação do estado do equipamento:**

  Classes 1 e 2: { Status

  Classes 3 a 5: { Status
  Unsolicited Status

  O serviço de Unsolicited Status deverá ser enviado a todas as associações, obviamente que suportem esta classe de conformidade, sempre que o estado do equipamento sofra alguma modificação.

- **Aquisição de dados por varredura:**

  Todas as classes: { Read

- **Aquisição de dados programada:**

  Classes 3 a 5: { Unsolicited Status
  Information Report

  Através do serviço de Unsolicited Status o servidor informa ao cliente que houve uma alteração no seu estado e que a informação esperada provavelmente não chegará.

- **Aquisição de dados configurada:**

  Classe 4: {
  Get Name List
  Unsolicited Status
  Read
  Report Event Condition Status
  Alter Event Condition Monitoring
  Event Notification
  Acknowledge Event Notification
  Report Event Action Status
  Report Event Enrollment Status

Classe 5:
- Get Name List
- Unsolicited Status
- Read
- Define Event Condition
- Delete Event Condition
- Report Event Condition Status
- Alter Event Condition Monitoring
- Get Event Condition Attributes
- Event Notification
- Acknowledge Event Notification
- Define Event Action
- Delete Event Action
- Get Event Action Attributes
- Report Event Action Status
- Define Event Enrollment
- Delete Event Enrollment
- Alter Event Enrollment
- Get Event Enrollment Attributes
- Report Event Enrollment Status

A requisição de leitura é feita através do serviço Read, mas a resposta enviada pelo servidor é iniciada através de um evento MMS. O serviço Unsolicited Status desempenha a mesma função da leitura anterior. O serviço Get Name List é utilizado, aqui, para obter-se o nome das condições de eventos, das ações de evento e do registro de eventos associados ao VMD.

- **Controle Paramétrico:**

Todas as classes:
- Write

O serviço Write é utilizado pelo cliente para alterar os valores das variáveis que parametrizam o processo no servidor. Após sua utilização pelo menos um serviço de leitura deve ser utilizado para que se obtenha a realimentação do controle.

- **Controle por intertravamento:**

Classes 3 a 5:
- Get Name List
- Data Exchange

O serviço de Data Exchange não faz parte do protocolo MMS sendo uma definição específica do Padrão Associado dos controladores programáveis.

- **Sincronização entre aplicações do cliente:**

Classe 4:
- Get Name List
- Cancel
- Take Control
- Relinquish Control
- Report Semaphore Status
- Report Semaphore Entry Status

Classe 5:
- Get Name List
- Cancel
- Take Control
- Relinquish Control
- Report Semaphore Status
- Report Semaphore Entry Status
- Define Semaphore
- Delete Semaphore

O serviço Get Name List é utilizado, aqui, para obter-se o nome dos semáforos fornecidos pelo controlador programável.

- **Alarmes programados:**

Classes 4 e 5:
- Event Notification
- Acknowledge Event Notification
- Get Alarm Summary

- **Alarmes configurados:**

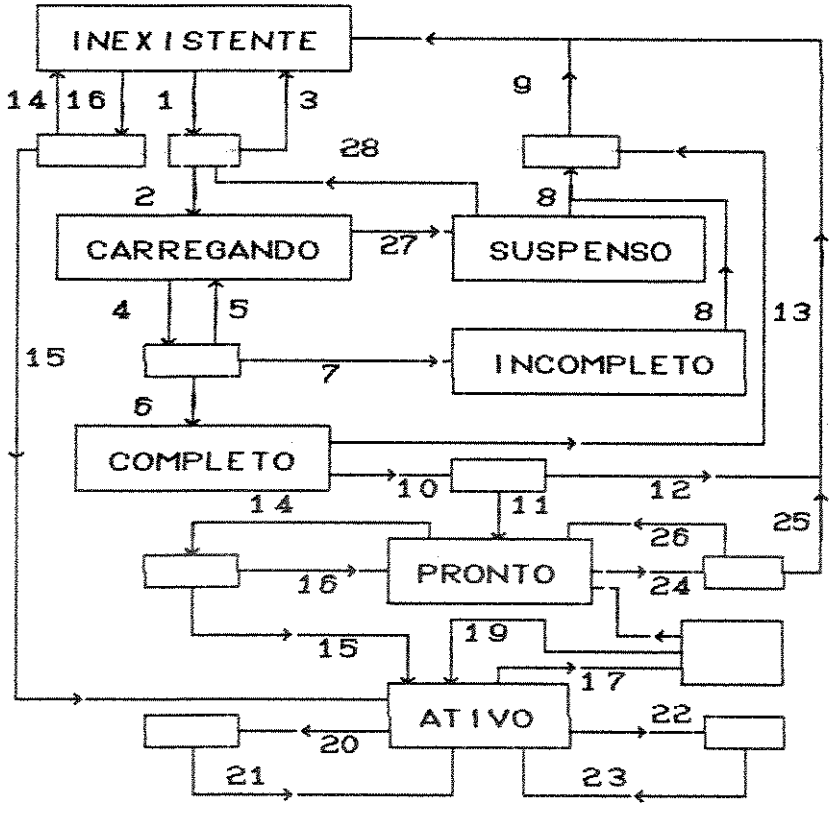
Classe 5:
- Get Name List
- Define Event Condition
- Delete Event Conditon
- Get Event Condition Attributes
- Report Event Condition Status
- Alter Event Condition Monitoring
- Event Notification
- Acknowlwdge Event Notification
- Define Event Action
- Delete Event Action
- Get Event Actions Attribute
- Report Event Action Status
- Define Event Enrollment
- Delete Event Enrollment
- Alter Event Enrollment
- Get Event Enrollment Attributes
- Report Event Enrollment Status
- Get Alarm Summary

- **Gerenciamento de Programa:**

Classe 2 a 5:
- Get Name List
- Initiate Download Segment
- Download Segment
- Terminate Download Segmnet
- Initiate Upload Segment
- Upload Segment
- Terminate Upload Sequence
- Delete Domain
- Get Domain Attributes
- Create Program Invocation
- Delete Program Invocation
- Get Program Invocation Attributes
- Start
- Stop
- Resume
- Reset

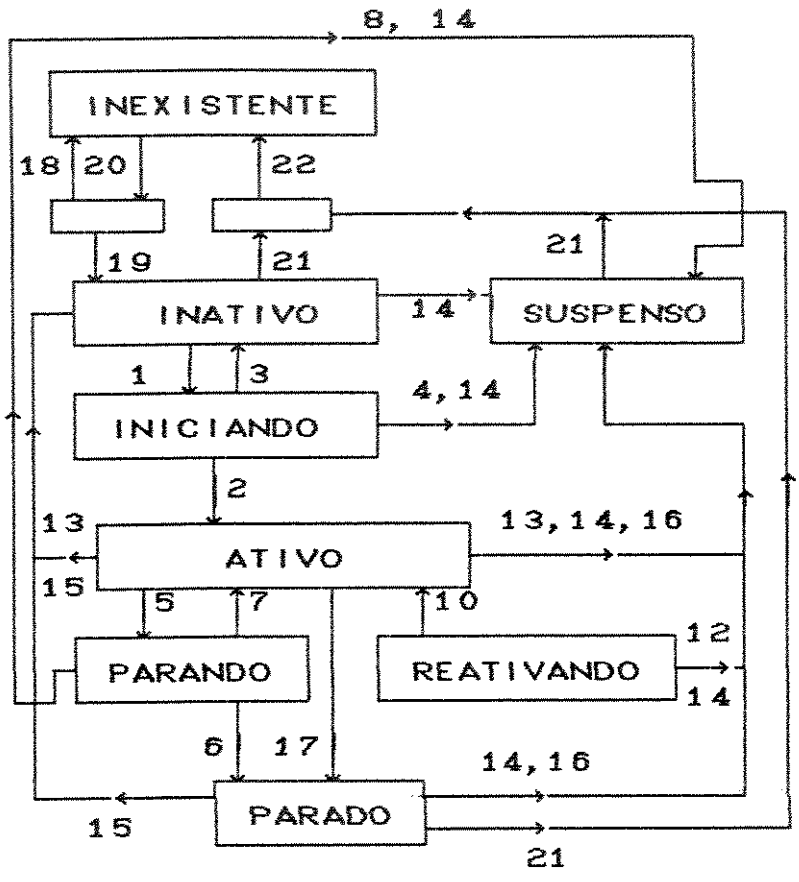
Tendo adquirido conhecimento sobre o mapeamento das funcionalidades do CLP para serviços MMS, requisitam-se agora, do

programador do AP cliente, conhecimentos específicos do protocolo MMS. Ele deverá, por exemplo, conhecer os serviços definidos para operar sobre Domínios e também, os serviços que permitem associar Domínios à Invocações de Programa. Uma vez que, é através dos serviços de Gerenciamento de Domínios que o cliente poderá carregar/descarregar o programa aplicativo do servidor e suas respectivas base de dados, e é através dos serviços de Invocação de Programas que ele poderá executar os programas aplicativos carregados no servidor. Para que o programador tenha sucesso nesta tarefa, ele deverá estar a par do funcionamento das máquinas de estado para os serviços de Gerenciamento de Domínios e de Gerenciamento de Programas (figuras 4.6 e 4.7).



1.IDLS_ind
2.IDLS_rsp (+)
3.IDLS_rsp (+)
4.DS_req
5.DS_cnf (+) MF = true
6.DS_cnf (+) MF = false
7.DS_cnf (-)
8.TDS_req
9.TDS_cnf (+) ou (-)
10.TDS_req
11.TDS_cnf (+)
12.TDS_cnf (-)
13.TDS_req
14.CPI_ind
15.CPI_rsp (+)
16.CPI_rsp (-)
17.DPI_ind
18.DPI_rsp (+)
19.DPI_rsp (-)
20.CPI_ind
21.CPI_rsp (+) ou (-)
22.DPI_ind
23.DPI_rsp (+) ou (-)
24.DD_ind
25.DD_rsp (+)
26.DD_rsp (-)
28.AL
29.IDLS_ind

FIG 4.6 - TABELA DE ESTADOS DO GERENCIAMENTO DE DOMÍNIOS



1.Start_ind
2.Start_rsp (+)
3.Start_rsp (-)
4.Start_rsp (-)
5.Stop_ind
6.Stop_rsp (+)
7.Stop_rsp (-)
8.Stop_rsp (-)
9.Resume_ind
10.Resume_rsp (+)
11.Resume_rsp (-)
12.Resume_rsp (-)
13.(término do programa)
14.Kill_ind
15.Reset.ind
Reset_rsp(+)
16.Reset_rsp (-)
17.(programa parado)
18.CPI_ind
19.CPI_rsp (+)
20.CPI_rsp (-)
21.DPI_ind
22.DPI_rsp (+) ou (-)

FIG 4.7 - TABELA DE ESTADOS DO GERENCIAMENTO DE PROGRAMA

Com o conhecimento das tabelas de estado destes serviços, o programador estará evitando o recebimento de primitivas de Reject.

A partir deste ponto, o cliente já possui as informações suficientes para a implementação do Programa de Aplicação.

4.2 - EXEMPLO DE APLICAÇÃO

Como exemplo de aplicação, suponha-se um controlador programável, classe 3, controlando várias máquinas do processo que estão sendo utilizadas na montagem de motores elétricos.

A informação necessária ao controlador programável para a montagem dos motores será carregada em diversos Domínios. O tipo dos Domínios se diferencia pela informação armazenada. Existem dois tipos básicos.Os do primeiro tipo podem conter todo o código do programa aplicativo do CLP, por exemplo na linguagem LADDER, com a suas diversas subrotinas: motor_ca, motor_cc, etc. Já os do segundo tipo contêm os dados necessários para a montagem dos diferentes tipos de motores. Se o cliente desejar construir um mesmo tipo de motor, mas com características diferentes como, por exemplo, o número de espiras, ele irá realizar um controle paramétrico sobre o sistema, escrevendo nas variáveis que parametrizam o processo.

O AP, localizado no Controlador de Célula, realizará as seguintes funções:

- **Associar o AP a uma AE** (serviço "AE_ACTIVATION"). Com este serviço o AP será registrado como um usuário da rede e receberá um AE_LABEL, através do qual será identificado nas chamadas subsequentes;

- **Iniciar o estabelecimento da associação** (serviço "CONNECT"). Quando o cliente requisita este serviço ele estará na realidade enviando a PDU de A_ASSOCIATE.request. No caso de uma resposta positiva, um CONNECTION_ID será fornecido para identificação da conexão. Nesta etapa, o

Controlador de Célula irá negociar com o CLP servidor a classe de conformidade suportada. O Cliente e o Servidor devem ter como norma negociar sempre a classe de conformidade mais alta por eles suportada, de tal forma que o resultado final seja o mais abrangente possível para o par cliente-servidor;

- **Receber primitivas de indication do servidor** (serviço "INDICATION RECEIVE"). Sendo a API implementada, para o MMS, uma API mestre, ela poderá requisitar todos os serviços de Gerenciamento de Conexão e quase todos os serviços Confirmados do MMS, com exceção daqueles que sua máquina de protocolos não permita. A recepção de quase todos os serviços iniciados pelo servidor será feita pelo serviço de "INDICATION RECEIVE" e será específica da associação. Este primeiro pedido é deixado como pendente, para o recebimento de indicações não solicitadas de serviços como "ABORT", "INFORMATION REPORT", etc. evitando que estas situações de exceção deixem de ser tratadas;

- **Obter o estado do CLP** (serviço "STATUS"). A fim de ter certeza de que o CLP está em condições de funcionamento para iniciar a montagem do motores, o Controlador de Célula requisita o estado do CLP;

- **Criar o Domínio do Programa** (serviço "INITIATE DOWN LOAD SEQUENCE"). Através deste serviço o cliente, Controlador de Célula, irá requisitar a criação do Domínio. O serviço irá conter o nome do Domínio, por exemplo "Montagem", e os recursos do equipamento, por exemplo "memória 7000-7500, portas de entrada 1 e 2, portas de saída 3 E 4". O Domínio, por sua vez, irá conter todo o código e dados do aplicativo escrito na linguagem LADDER. Após o recebimento da PDU de resposta, o estado do Domínio será mudado de INEXISTENTE para CARREGANDO (figura 4.6.). A título de exemplo, este

serviço teria a seguinte notação padrão, de acordo com o anexo B da especificação ASN.1 [02]:

```
|| Notação padrão
    { confirmed_RequestPDU
      { invokeID                                26,
        ConfirmedServiceRequest
        { initiateDownloadSequence
          { DomainName                          "Montagem"
            listOfCapabilities                  "MEMORY 5000-7FFF,
                                                 CODE 0000 - 3FFF,
                                                 DATA 4000 - 4FFF,
                                                 INPUT PORTS 1-2,
                                                 OUTPUT PORTS 3-4"
             sharable                           TRUE
          }
        }
      }
    }
```

- **Esperar pelo pedido do CLP de início do carregamento do Domínio** (serviços "INDICATION RECEIVE" e "DOWNLOAD SEGMENT RESPONSE"). Tendo conhecimento da máquina de estado do Gerenciamento de Domínios o cliente irá requisitar o serviço de "INDICATION RECEIVE" para receber a indicação do serviço de "DOWNLOAD SEGMENT REQUEST" iniciado pelo servidor, o recebimento desta indicação moverá o estado do Domínio de CARREGANDO para COMPLETO e, por conseguinte, o cliente irá fazer quantas requisições ao serviço de "DOWNLOAD SEGMENT RESPONSE" forem necessárias para o carregamento dos dados no Domínio;

- **Esperar pelo pedido do CLP de término do carregamento do Domínio** (serviços "INDICATION RECEIVE" e "TERMINATE DOWNLOAD SEQUENCE RESPONSE"). Novamente o cliente faz uso do serviço de "INDICATION RECEIVE" para o recebimento de um serviço iniciado pelo servidor: "TERMINATE DOWNLOAD SEQUENCE REQUEST". O recebimento desta indicação irá mover o estado do Domínio de COMPLETO para PRONTO. O serviço

"TERMINATE DOWNLOAD SEQUENCE RESPONSE" será requisitado pelo cliente para a emissão da resposta;

- **Criar o Domínio dos Dados** (serviço "INITIATE DOWN LOAD SEQUENCE"). Este primeiro Domínio de Dados conterá os dados necessários para a fabricação de motores CA. Para a criação de um Domínio de dados necessita-se somente especificar, como recurso, um endereço de memória, por exemplo 7501-75ff. Este Domínio irá receber o nome de "Motor_CA";

- **Esperar pelo pedido do CLP para carregamento do Domínio** (serviços "INDICATION RECEIVE" e "DOWNLOAD SEGMENT RESPONSE"). Estes serviços serão utilizados da mesma forma já anteriormente apresentada;

- **Esperar pelo pedido do CLP de término do carregamento do Domínio** (serviços "INDICATION RECEIVE" e "TERMINATE DOWNLOAD SEQUENCE RESPONSE"). Estes serviços serão utilizados da mesma forma já anteriormente apresentada;

- **Criar uma Invocação de Programa** (serviço "CREATE PROGRAM INVOCATION"). O Controlador de Célula já criou os Domínios necessários para a fabricaçào de motores CA mas ele ainda necessita criar uma Invocação de Programa que possa ser iniciada e parada, a fim de que o CLP possa, realmente, fazer uso destes Domínios. Esta Invocação de Programa receberá o nome de "Faça_Motores_CA" e conterá os Domínios "Montagem" e "Motor_CA", criados anteriormente. Este serviço irá mudar o estado dos Domínios de PRONTO para ATIVO e mudar o estado da Invocação de Programa de INEXISTENTE para OCIOSO (figura 4.7);

- **Iniciar a montagem dos motores (CA)** (serviço "START"). Após o Controlador de Célula ter criado os Domínios e definido a Invocação de Programa que inclui os Domínios criados, ele

requisita o serviço "START" para iniciar a fabricação regular de motores CA. Se o CLP iniciou a montagem dos motores ele irá enviar uma resposta positiva para o serviço "START" e o estado da Invocação de Programa mudará para ATIVO;

- **Parar a montagem dos motores (CA)** (serviço "STOP"). Após um certo número de motores CA padrão terem sido montados, o Controlador de Célula decide fazer novos motores CA com um número diferente de espiras. Para tal a Invocação de Programa deve ser parada e a variável que define o número de espiras do motor deve ser alterada. Através da requisição do serviço STOP ele pede que o CLP pare a montagem dos motores. Quando a montagem chega ao fim o CLP envia uma resposta ao serviço "STOP" que mudará o estado da Invocação de Programa para PARADO;

- **Alterar uma variável de controle paramétrico** (serviço "WRITE"). O Controlador de Célula muda o valor do número de espiras do motor escrevendo na váriavel "NumeroEspiras" do Domínio "Motor_CA". Este é um exemplo de aplicação de controle paramétrico;

- **Reinicializar a montagem dos motores (CA)** (serviço "RESUME"). Neste ponto iniciar-se-á a fabricação de motores CA com um número diferente de espiras. O estado da Invocação de Programa mudará de PARADO para ATIVO;

- **Criar um novo Domínio dos Dados** (serviço "INITIATE DOWN LOAD SEQUENCE"). O Controlador de Célula decide que já existe um número suficiente de motores CA montados, então, decide montar motores CC. Como primeiro passo para a montagem de motores CC, o Controlador de Célula cria o Domínio de Dados chamado "Motor_CC". Não será preciso parar a execução da Invocação de Programa "Faça_Motores_CA" para

a criação deste novo Domínio, pois este não pertence à Invocação de Programa referida;

- **Esperar pelo pedido do CLP para carregamento do Domínio** (serviços "INDICATION RECEIVE" e "DOWNLOAD SEGMENT RESPONSE"). Estes serviços serão utilizados da mesma forma já anteriormente mostrada;

- **Criar uma nova Invocação de Programa** (serviço "CREATE PROGRAM INVOCATION"). Este serviço irá requisitar a criação da Invocação de Programa chamada "Faça_Motores_CC" que conterá os Domínios "Montagem" e "Motor_CC";

- **Parar a montagem dos motores (CA)** (serviço "STOP"). O Controlador de Célula requisita ao CLP que pare a fabricação de motores CA. Quando a montagem chega ao fim o CLP envia uma resposta ao serviço "STOP" que mudará o estado da Invocação de Programa para PARADO;

- **Iniciar a montagem de um novo tipo de motor (CC)** (Serviço "START"). Tão logo a Invocação de Programa "Faça_Motores_CA" tenha parado, o controlador programável inicia a montagem dos motores CC através do serviço "START" que irá ativar a Invocação de Programa "Faça_Motores_CC";

- **Parar a montagem dos motores (CC)** (Serviço "STOP"). O Controlador da Célula requisita ao CLP que pare a montagem dos motores CC. Quando a montagem chega ao fim o CLP envia uma resposta ao serviço "STOP" que mudará o estado da Invocação de Programa para PARADO;

- **Liberar a associação** (Serviço "CONCLUDE"). O Controlador de Célula requisista a liberação da conexão ao CLP, caso este responda positivamente a conexão será liberada e ele poderá requisitar o serviço seguinte;

- **Liberar a Invocação de AE** (Serviço "AE_DEACTIVATION"). Através deste serviço o AE_LABEL anteriormente associado ao AP é liberado para o utilização da API;

A realização destas funções pelo AP traduziu-se no programa mostrado na figura 4.8. Este programa é implementado através de chamadas de funções, atingindo um dos objetivos da API que é o de ser uma interface amigável para o usuário. Tendo este objetivo como meta foi necessário, então, um esforço adicional de implementação para a criação de uma biblioteca que faça o mapeamento dos parâmetros das funções para o formato da mensagem trocada entre AP e a API no SISDI-MAP. Cada função desta biblioteca possui como procedimentos básicos, além do mapeamento descrito, enviar a mensagem mapeada para a porta de entrada da API e esperar pela mensagem de resposta das chamadas, através da utilização de semáforos fornecidos pelo núcleo. Estes semáforos são únicos para cada AP. Se o parâmetro RETURN_EVENT_NAME, presente em todas as chamadas de função, corresponder à uma chamada assíncrona, a API enviará uma mensagem de resposta indicando que a mensagem foi entregue ao MMS e fará uma sinalização sobre o semáforo do AP de tal forma que este possa continuar a sua execução. No caso de uma chamada síncrona o AP só será liberado pela API quando do envio da mensagem de resposta ao serviço solicitado, com a sua posterior sinalização deste evento sobre o semáforo do AP.

```
/* PROGRAMA DE APLICAÇÃO DO CLIENTE
                (CONTROLADOR DA CELULA) */
#INCLUDE "\sidimap\master.h"
#INCLUDE "\sisdimap\apitipos.h"
#INCLUDE "\sisdimap\aplib.h"
P_APLICACAO(){
/* ALOCA DCB´S */
/* INICIALIZA VARIAVEIS */
        /* ae activation */  /* sincrono */
IF(MM_AE_ACTIVATION(my_dir_name, return_event_name,
            input_dcb, inout_dcb, ae_label) == ERRO);
 GET_PRINTABLE_ERROR(ERRO);
ELSE
       /* connect */ /* sincrono */
 IF (MM_CONNECT( ae_label, return_event_name,
called_dir_name, input_dcb, inout_dcb, connection_id)
                                          == ERRO);
 ELSE
   GET_PRINTABLE_ERROR(ERRO);
       /* indication receive */ /* assincrono */
  IF(MM_IRECEIVE( connecton_id, return_event_name,
                  inout_dcb, indicaton_name) == ERRO);
   GET_PRINTABLE_ERROR(ERRO);
  ELSE
        /* status */  /* assincrono */
   IF (MM_STATUS( connection_id, return_event_name,
                  input_dcb, inout_dcb) == ERRO);
    GET_PRINTABLE_ERROR(ERRO);
   ELSE
 /* initiate down load sequence */  /* assincrono */
    IF (MM_IDSEQUENCE( connection_id, return_event_name,
                       input_dcb, inout_dcb) == ERRO);
     GET_PRINTABLE_ERROR(ERRO);
    ELSE
          /* indication receive */   /* assincrono */
     IF(MM_IRECEIVE( connecton_id, return_event_name,
                  inout_dcb, indicaton_name) == ERRO);
      GET_PRINTABLE_ERROR(ERRO);
     ELSE
           /* wait */ /* sincrono */
      WAIT(connection_id,return_event_name,
                        inout_dcb, FOREVER );
      IF( indication_name == DownLoadSegmentRequest);
           /* down load segment response */
       IF(MM_DSRESPONSE( connection_id, return_event_name
                       input_dcb, inout_dcb) == ERRO);
        GET_PRINTABLE_ERROR(ERRO);
                  :
                  :
```

FIG.4.8 - PROGRAMA DE APLICAÇÃO DO CLIENTE

Note-se que o código do Programa de Aplicação, descrito acima, contem, após cada requisição de serviço, um teste para verificação de erros em tempo de execução. Estes testes são necessários porque mesmo que o programador passe os parâmetros sintaxicamente corretos, existem alguns parâmetros que são controlados pela API tais como número da invocação de AE, número de conexão, número do return_event_name etc. e que a priori, o usuário não tem meios de saber se estão sendo utilizados por outros AP's. Além do controle destes parâmetros, existe um erro que indica o esgotamento temporário de recursos (memória) e que impede a continuação imediata da execução do Programa de Aplicação. Este erro é definido na especificação como NO_CONNECTION_RESOURCE.

Como sistemática de implementação, note-se, também, que este programa possui sempre uma Requisição de Indication_Receive pendente, para evitar que a chegada de indicações de Abort coloquem o AP em "deadlock".

4.3 - SUGESTÃO DE IMPLEMENTAÇÃO NO NÓ SERVIDOR

Mostra-se na figura 4.9, como seria a estrutura de uma implementação no nó servidor.

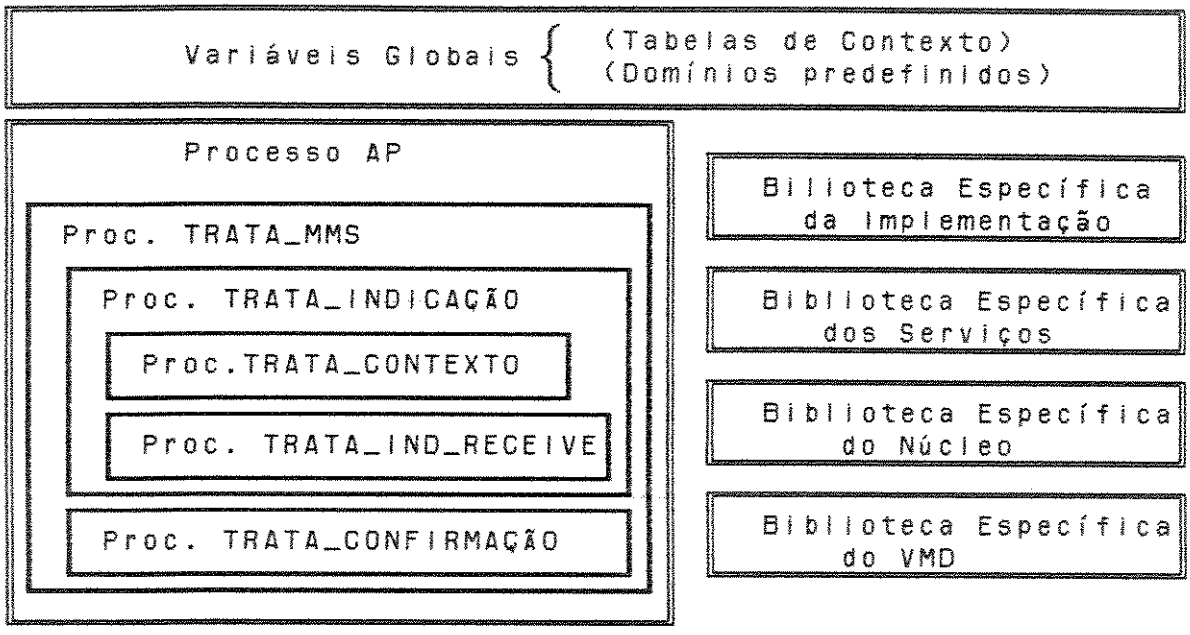


FIG. 4.9 - ESTRUTURA DA IMPLEMENTAÇÃO NO NÓ SERVIDOR

Note-se que existem grandes diferenças entre esta estrutura de implementação e a estrutura da implementação no nó cliente, descrita pela figura 3.1.

No que diz respeito às variáveis globais, acrescentou-se a alocação dos domínios predefinidos. Estes domínios deverão conter os objetos do VMD como, por exemplo, o objeto Manipulador do Rôbo, descrito pela figura 2.1.

Os Domínios, tanto predefinidos, quanto remotamente definidos, deverão ser manipulados através de serviços específicos da

Biblioteca Específica do VMD. Esta biblioteca deverá, também, fornecer os serviços necessários para adminstrar a inicialização dos segmentos de código, dos segmentos de dados, das portas de entrada e das portas de saída do VMD.

Já o bloco funcional denominado Processo AP absorveu do Processo API, descrito na figura 3.1, todo o tratamento das mensagens recebidas do MMS , mas não o tratamento das mensagens recebidas do AP, pois espera-se que o AP no servidor seja único, ou que , no máximo existam várias tarefas com a estrutura do processo AP acima.

Finalmente, a Biblioteca dos Serviços específicos deve ser bastante simplificada, quando comparada com o mesmo tipo de biblioteca do nó cliente, pois esta biblioteca conterá somente: um subconjunto dos serviços de gerenciamento de conexão acrescido dos serviços de resposta, dos serviços confirmados cuja máquina de estado do MMS define que sejam iniciados pelo servidor e, finalmente, dos serviços não confirmados.

4.4 - CONCLUSÕES

O Problema da Aplicação, título do capítulo, teve como finalidade ressaltar que mesmo com as facilidades expostas ao AP cliente, através da API, a implementação de um Programa de Aplicação torna-se, no mínimo, complexa devido ao número de conhecimentos que o programador deverá ter do protocolo MMS e dos conhecimentos específicos do equipamento. Neste escopo, os Padrões Associados são de suma importância para a implementação do AP.

Da mesma forma, a implementação de um AP no nó servidor também é uma tarefa complexa, como pode ser visto da estrutura de implementação proposta para o nó servidor.

5.0 - ESPECIFICAÇÃO

A existência de um padrão concreto não conduz à implementação de uma maneira direta. Ao contrário, esta é uma tarefa desafiante devido à natureza aberta da especificação e dos inúmeros pontos de ambiguidade existentes. Dentre estes pontos, podem-se citar os SAP's, CEP's, Invocações de AE, Invocações de AP, Associações, Conexões, Objetos, etc...

Para que se passe da especificação para a implementação, dois aspectos principais devem ser considerados. Primeiro, a implementação deve ser correta: deve estar em conformidade com o seu ambiente - tratando-se da implementação da API para o MMS, ela deve estar em conformidade com o Programa de Aplicação e com o Protocolo MMS. Este objetivo requer um investimento substancial no entendimento de sua especificação. Segundo, a implementação deve ser eficiente.

Contudo, somente uma definição precisa da semântica dos conceitos não é suficiente para resolver os problemas de implementação de um sistema baseado no modelo RM-OSI. Primeiro, é necessário planejar como será feito o mapeamento do modelo abstrato para a implementação real. Segundo, a especificação deve ser extendida para incluir o gerenciamento de recursos internos e outras funções de supervisão para uma operação efetiva do sistema. No caso particular da API, necessitou-se adotar uma estratégia de implementação que armazenasse variáveis de contexto para cada transação ativa que indicasse o estado corrente da interface e que, desta forma, determinasse o processamento a ser executado, quando do recebimento de primitivas.

Deve-se ressaltar que toda esta carga adicional de implementação não faz parte da especificação.

No SISDI-MAP, por tratar-se de um sistema didático, não se investiu em medidas que pudessem analizar o seu desempenho como software de comunicação, mas sim como um sistema didático cujo objetivo é auxiliar o entendimento do ambiente OSI e adquirir experiência para novas implementações. Neste sentido, fez-se uma análise de diferentes tipos de implementação, que dentre as mais importantes destacam-se:

- a criação de uma única tarefa API;
- a criação de uma tarefa API "filha" para cada invocação de AE;
- a impementação dos blocos funcionais, descritos nos modelos apresentados no capítulo dois, como tarefas diferentes;
- a criação de uma tarefa API "filha" para cada Associação.

Por motivos de simplicidade criou-se, para esta primeira versão, uma única tarefa API.

5.1 - IMPLEMENTAÇÃO

A escolha apropriada do tipo de serviço, das unidades funcionais e demais aspectos do protocolo não requer somente um entendimento por completo do padrão, mas também uma interação eficiente com os outros implementadores do software de comunicação.

Um problema particular encontrado, quando do início da implementação, foi a diferença de versões disponíveis para a API e para o MMS. Nestas circunstâncias, necessitou-se fazer um trabalho cuidadoso junto com o grupo responsável pela implementação do MMS e do ACSE, que resultou na criação, cancelamento e modificação de serviços fornecidos pela API [10], [11], [12], [14] para adequá-los ao draft 6 da especificação do MMS [04], [05]. Por exemplo, toda a estrutura de dados dos serviços foi adequada à nova especificação. Por exemplo, os serviços de gerenciamento de conexão que não eram suportados na versão anterior, tais como "CANCEL" e "REJECT", tiveram de ser criados.

Devido ao caráter assíncrono da troca de mensagens entre AP⇔API e API⇔MMS, necessitou-se adotar uma estratégia para o controle do fluxo destas mensagens, que evitasse o "gargalo" do processo API. A solução adotada foi a alocação de recursos (memória), tanto pela API quanto pelos AP's, de um pool de buffers. Nesta abordagem, cada AP terá somente um número limitado de quantidade de memória para que ele possa enviar uma mensagem para a API, similarmente à implementação de filas finitas. Em contrapartida, esta opção atrasa a execução de um determinado AP, pois este fica limitado a uma quantidade constante de memória para troca de mensagens.

Com a experiência adquirida durante a implementação, pôde-se verificar que, apesar da API ser modelada através de vários blocos

funcionais, estes não devem ser traduzidos em processos diferentes, porque esta abordagem os colocaria como servidores de todos os processos implementados pela API e certamente haveria um "overhead" na troca de mensagens entre estes processos.

Como citado na página 5.2, esta versão atual do SISDI-MAP possui um único processo API que controla toda a troca de mensagens entre as associações estabelecidas. Terminado o processo de implementação, deve-se destacar que este modelo não mostrou-se o mais adequado às características de um ambiente de tempo real. Pôde-se constatar que o modelo mais adequado para a implementação, seria a criação de um processo API "pai", responsável pela criação de vários processos "filhos", onde cada processo filho estaria relacionado à uma determinada associação. Este modelo de implementação é resultado de uma necessidade percebida durante o desenvolvimento do trabalho que, quando implementada, permitirá estabelecer prioridades diferentes para cada processo, de tal forma que se possa, por exemplo, privilegiar uma tarefa de alarme em relação à uma tarefa comum. Além do que, esta nova estrutura, retiraria o excesso de troca de mensagens com uma único processo, previlegiando a característica multitarefa do ambiente de programação.

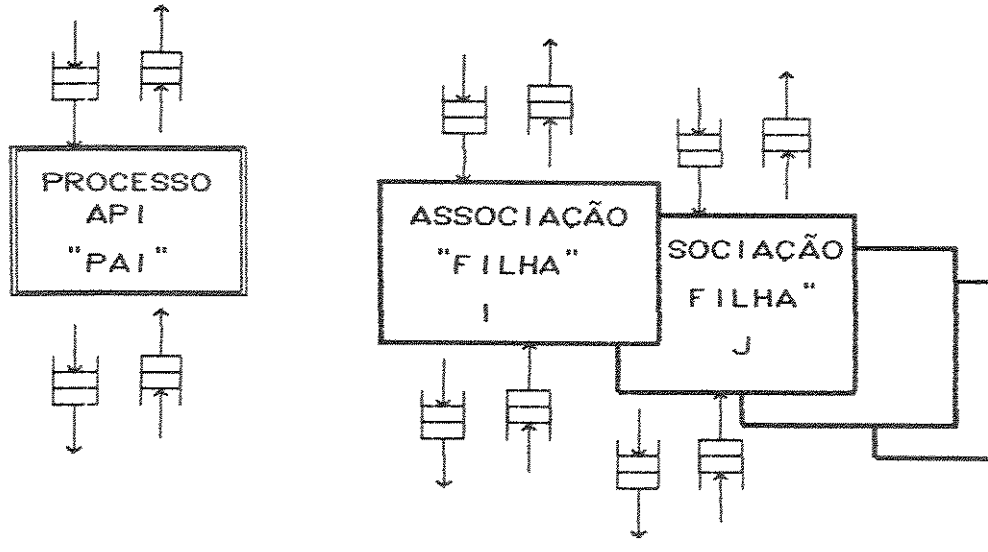


FIG. 5.0 - MODELO DE IMPLEMENTAÇÃO DE UMA API COM MULTIPROCESSOS

Devido a relação cliente/servidor estabelecida entre os equipamentos comunicantes em um ambiente MMS, pôde-se verificar que a existência da API é extremamente necessária em um nó cuja função seja claramente de cliente, como por exemplo, um controlador de célula. No controlador de célula, a existência da API irá permitir uma implementação facilitada de um Programa de Aplicação cujo algorítimo será basicamente uma sequência de chamadas de funções. Desta forma, toda a carga da máquina de protocolos da API é transparente para o usuário. A sua existência auxiliada a um sistema operacional padrão traria grande reusabilidade do Programa de Aplicação.

Deve-se ressaltar a importância dos Padrões Associados para a elaboração do Programa de Aplicação. Sem a utilização destes documentos esta tarefa não teria sucesso algum.

Nos nós cuja função seja exclusivamente de servidor, a implementação por completa da API não se faz necessário, pois o servidor funciona basicamente como um respondedor de serviços. Uma opção viável para a ausência da API, seria a solução proposta no capítulo 4, onde a criação de um AP mais robusto que absorviria algumas funções da máquina de protocolos da API e que, possuiria tabelas mapeando todos os atributos dos objetos do VMD e o contexto decorrente da classe de conformidade da associação. Note que neste caso o AP seria único por servidor.

Como ferramenta de implementação, a Linguagem C mostrou-se bem adequada à implementação devido à necessidade de se trabalhar com estruturas e apontadores e também devido às características de tempo real da implementação.

5.2 - FUTURO

Espera-se que com o desenvolvimento apresentado pelas Técnicas de Descrição Formal (FDT's): LOTOS [33] e ESTELLE [34], o problema da ambiguidade das especificações sejam resolvidos, uma vez que estas ferramentas devem permitir uma especificação precisa e definitiva que evite diferentes interpretações e, consequentemente, implementações incompatíveis.

Esta primeira versão do SISDI-MAP está voltada para um tipo particular de aplicação, estando instalada em um único microcomputador IBM-PC compatível. Este ambiente é mais adequado para uma API Mestre, devendo-se, portanto, migrar o sistema para um sistema operacional UNIX compatível. Esta mudança permitará uma maior portabilidade e reusabilidade do Programa de Aplicação e da API. Já no nó servidor, espera-se que o AP mais robusto, descrito na tese, seja implementado diretamente na placa de comunicação. Neste caso, a adoção de um núcleo de tempo real particular torna-se adequada, atingindo também para este caso, os objetivos de portabilidade e reusabilidade do software.

REFERÊNCIAS BIBLIOGRÁFICAS

[01] ISO/DIS 7498 - "Information Processing Systems - Open Systems Interconnection: Basic Reference Model", Outubro 1982.

[02] ISO/DIS 8824 - "Specification of Abstract Syntax Notation One (ASN.1)", Agosto 1986.

[03] ISO/DIS8825 - "Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1), Agosto 1986.

[04] ISO/DIS 9506 - "Manufacturing Message Specification. Part 1: Service Specification", Draft 6, Maio 1987.

[05] ISO/DIS 9506 - "Manufacturing Message Specification. Part 2: Protocol Specification", Draft 6, Maio 1987.

[06] ISO/DIS 8649/2 - "Service Definition for Common Application Service Elements - Association Control", Abril 1986.

[07] ISO/DP - Proposed EIA RS-511 "Manufacturing Message Specification. Part 1: Service Definition", Draft 5, Junho 1986.

[08] ISO/DP - Proposed EIA RS-511 1393A "Manufacturing Message Specification. Part 2: Protocol Specification", Draft 5, Junho 1986.

[09] GM - "Manufacturing Automation Protocol", Versão 3.0, Julho 1987.

[10] GM-MAP/TOP - "Application Program Interface - Model Description and Interface Specification Requirements",Dezembro 1987.

[11] GM-MAP/TOP - "Application Program Interface - Connection Management Interface Specification", Novembro 1987.

[12] GM-MAP/TOP - "Application Program Interface - Application Interface Support Functions", Março 1987.

[13] GM-MAP/TOP - "Application Program Interface - MMS Application Interface Specification", Novembro 1987.

[14] GM-MAP/TOP - "Application Interface Model and Specification Requirements", Junho 1988.

[15] GM-MAP/TOP - "Connection Management Interface Specification", Junho 1988.

[16] GM-MAP/TOP - "Application Interface Support Functions", Junho 1988.

[17] GM-MAP/TOP - "MMS Application Interface Specification", Junho 1988.

[18] IEC/SC65A/WG6/TF7 - "Programmable Controller Message Specification", Fevereiro 1989.

[19] ISO/TC 184/SC1 - "Numerical Control Message Specification: A Companion Standard to ISO 9506/1 + 2", Fevereiro 1989.

[20] ISO/TC 184/SC 2 - "Robot Companion Standard to MMS", Julho 1989.

[21] Mendes, M.J. - "Comunicação Fabril e o Projeto MAP/TOP", Escola Brasileira Argentina de Informática - IV EBAI, 1989.

[22] Dwyer, John; Ioannou, Adrian - "MAP and TOP: Advanced Manufacturing Communications, Korgan Page, 1987.

[23] Halsall, Fre - "Data Communications, Computer Networks and OSI", Addison-Wesley, 1988.

[24] Tanenbaum, Andrew S - "Computer Networks", Prentice Hall, 1987.

[25] Giosa, W.F.; Araújo, J.F.M.; Moura, J.A.B.; Sauvé, J.P. - "Redes Locais de Computadores - Tecnologia e Aplicação", McGraw-Hill, 1986.

[26] Moura, J.A.B; Sauvé, J.P.; Glozza, W.F.; Araújo, J.F.M. - "Redes Locais de Computadores - Protocolos de Alto Nível. Avaliação de Desempenho", McGraw-Hill, 1986.

[27] Stacy, A.H. - "The MAP Book: An Introduction to Industrial Networking" , Industrial Network Incorporated, 1987.

[28] Madeira, E.R.M.; Mendes, M.J. - "Interface de Programas de Aplicação para o Protocolo MMS (RS-511)" Anais do 3º CONAI, São Paulo, Setembro 1988.

[29] Paglioni, A.; Jacintho, D.C.A.; Madeira, E.R.M.; Fernandes, I.A.; Mendes, M.J.; Correa, J.N.; Lima, J.M.S.; Zabeu, M.C.; Sousa, V.L.P.- "SISDI-MAP: Sistema Didático do Protocolo e da Interface de Aplicação MMS do MAP". - Seminário Franco-Brasileiro em Sistemas Distribuídos, Florianópolis, Setembro 1989.

[30] Madeira, E.R.M; Correa, J.N.; Sousa, V.L.P.; Mendes, M.J.- "Modelamento de Interfaces de Aplicação e Exemplo de Implementação para protocolo MMS", Anais do SBRC, Campinas, Abril 1990.

[31] Madeira, E.R.M, Mendes, M.J. - "The Application Interface Model for Communication Software and MMS Protocol Implementation Example" - Productique & Integrations CIM - Integration Aspects - Bourdeaux , França , Junho 1990.

[32] Svobodova, L. - "Implementing OSI Systems", - IEE Journal on Selected Areas in Communication, Vol. 7. nº 7, Setembro 1989.

[33] Bolognesi, T.; Brinksma, E - "Introduction to the ISO specification language LOTOS", Comput. Networks ISDN Syst. vol. 14 nº 1, 1987.

[34] Budkowski, S.; Dembinski, P. - "An introduction to Estelle: A specification language for distributed systems", Comput. Networks ISDN Syst. vol. 14 nº 1, 1987.

[35] Jacintho, D.C.A.- "Aspectos de Especificação e Implementação da Estrutura de Mensagens de um Sistema Didático do Protocolo MMS"- Campinas - UNICAMP/FEE/DCA, Dezembro 1989.

[36] Zabeu, M.C.- "Um Modelo para Configuração de Núcleos para Tempo Real", Campinas - UNICAMP/FEE/DCA, Dezembro 1989.

[37] Sousa, V.L.P. - "Aspectos de Especificação e Implementação da Interface de Aplicação para o MMS: Serviços de Acesso a Variáveis", Tese de Mestrado em conclusão , UNICAMP/FEE/DCA.

[38] Madeira, E.R.M. - Tese de doutorado em andamento, UNICAMP/FEE/DCA.

[39] Páglioni, A.J.- Tese de mestrado em andamento, UNICAMP/FEE/DCA.

[40] Fernandes, I.A. - Tese de mestrado em andamento, UNICAMP/FEE/DCA.

[41] Lima, J.M.S. - Tese de mestrado em andamento, UNICAMP/FEE/DCA.