MODELAMENTO DE INTERFACES DE APLICAÇÃO E EXEMPLO DE IMPLEMENTAÇÃO PARA O PROTOCOLO MMS

Edmundo Roberto Mauro Madeira
Jayme Nicolato Correa
Verônica Lima Pimentel de Sousa
Manuel de Jesus Mendes

UNICAMP - Faculdade de Engenharia Elétrica
13081 - Campinas - SP

SUMÁRIO

Este artigo descreve o modelamento de Interfaces de Programas de Aplicação (API's) para diversos Protocolos de Aplicação construindo um Modelo Geral. É analisada a implementação de API para o Protocolo MMS no SISDI-MAP - Sistema Didático do Protocolo e Interface de Aplicação MMS do MAP - em desenvolvimento no Departamento de Computação e Automação Industrial da FEE - UNICAMP.

1. INTRODUÇÃO

O Projeto MAP - Manufacturing Automation Protocol define os protocolos para as diversas camadas do Modelo OSI da ISO [1] para a interconexão de equipamentos programáveis em automação industrial. Para a camada de aplicação, o MAP utiliza na parte de serviços específicos denominada SASE - "Specific Application Service Elements" - os serviços do Protocolo MMS - Manufacturing Message Specification. O Projeto MAP também define uma interface entre o Programa de Aplicação (AP) e o Protocolo de Aplicação denominada API - Application Program Interface.

Os serviços MMS são oferecidos para serem executados em dispositivos virtuais (VMD's) que possuem domínios, sistema de arquivos virtuais, estação do operador e função executiva. As classes de serviços MMS abrangem: gerenciamento de contexto, suporte de VMD, gerenciamento de domínio, gerenciamento de invocação de programa, acesso a variáveis, gerenciamento de semáforos, comunicação de operadores, gerenciamento de eventos, gerenciamento de jornal e gerenciamento de arquivos. Num ambiente real, nem sempre todos esses serviços são utilizados em todas as estações. Normalmente, os serviços de gerenciamento de contexto estão sempre presentes em qualquer aplicação, já que são os responsáveis pelo estabelecimento e término de associações.

O SISDI-MAP é um sistema didático para o Protocolo de Aplicação MMS e a sua respectiva API. O SISDI-MAP é um sistema multitarefa que permite a execução de vários AP's que se comunicam entre si, com o objetivo de emular aplicações reais. O SISDI-MAP foi implementado inicialmente para oferecer os serviços de gerenciamento de contexto e acesso a variáveis.

Na Seção 2 será apresentado um Modelo Geral de API, utilizado na implementação do SISDI-MAP que pode ser estendido para outros Protocolos de Aplicação. Na Seção 3 são descritos alguns aspectos da implementação da API para o SISDI-MAP e a Seção 4 apresenta as conclusões do trabalho.

2. MODELAMENTO DE INTERFACES DE APLICAÇÃO (MAP/TOP)

A Interface de Aplicação (API) para o projeto MAP/TOP é descrita em [2] e é uma interface entre o programa de aplicação e o sistema de comunicação. Seus objetivos, funcionalidades e definições podem ser vistas em detalhes em [3].

Em [3] definiu-se a API como tendo três partes: Biblioteca (LIB), Procedimentos de Controle e Provedor de Serviços de Primitivas (PSP). LIB contém as funções que são chamadas pelos programas aplicativos para requisitar os serviços desejados da rede. O PSP é responsável pelo envio/recebimento de primitivas para/da camada de aplicação.

Os procedimentos de controle são três: Provedor de Serviços de Alto Nível (HLSP), Provedor de Serviços Confirmados (CSP) e Filtro e Árbitro de Indicações (IFA). O HLSP trata das funções de alto nível (funções que fazem uso de várias funções de serviços de baixo nível para completarem suas tarefas), o CSP trata dos serviços confirmados e o IFA trata da recepção de primitivas de indicação.

2.1 - PROTOCOLO MMS E ESTRUTURA DE API

O Protocolo de Aplicação MMS [4], caracteriza-se num modelo "Cliente/Servidor" por inúmeras requisições complexas de serviços feito pelo cliente a estações simples (CNC, CLP's, etc.), o que pode levar a um esquema de resposta dos serviços solicitados (requisições que chegam como indicações) por parte da própria API. Portanto, deve-se adicionar ao Modelo da API uma nova unidade funcional para Processamento de Indicações (IP). Esta unidade tem as seguintes funções:

a) Processar os serviços solicitados pelas Primitivas de Indicação aceitas,

b) Acessar a memória principal e diretório local de arquivos quando necessário,

c) Solicitar ao CSP (se o serviço é confirmado) ou ao PSP (se o serviço não é confirmado) o envio de Primitivas de Resposta às indicações aceitas.

Os serviços MMS são oferecidos sobre Dispositivos Virtuais da Manufatura (VMD) que devem corresponder a Dispositivos Reais. Torna-se necessária uma função de Mapeamento Dispositivo Virtual/ Dispositivo Real e vice-versa. Introduz-se no Modelo da API a unidade funcional (Virtual Device/Real Device Binding) VDRDB. É utilizando esta unidade funcional que o IP pode realizar as funções de acesso à memória principal e Diretório de Arquivos.

O Modelo de API para o Protocolo MMS é o da Figura 1.

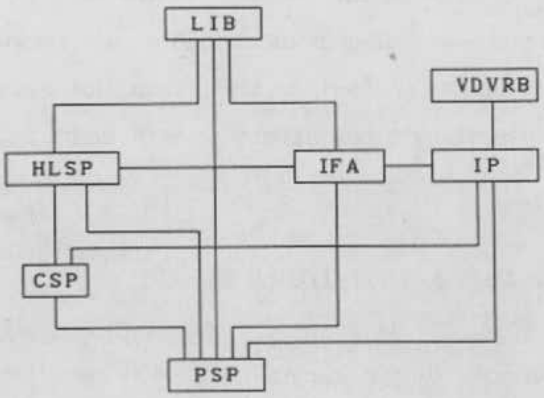

Figura 1 - Modelo de API para o MMS

2.2 - OUTROS EXEMPLOS DE API's

Protocolo FTAM

O Protocolo FTAM [5] oferece serviços de gerenciamento e acesso de transferência de arquivos sobre uma estrutura de Sistema de Arquivo Virtual. As estações podem possuir Sistemas de Arquivos diferentes, mas todas as estações se comunicarão utilizando uma "linguagem" comum que é o Sistema de Arquivos Virtual definido no

FTAM. Em cada estação, só é necessário fazer o mapeamento do Sistema de Arquivos Virtual que todas as estações conhecem para o Sistema Real de Arquivos que só a estação local conhece. Para o FTAM existem os conceitos de Estação Servidora e de Estação cliente. A Estação Servidora é uma estação complexa e isto pode levar a um esquema de resposta por parte de um Programa Aplicativo especializado. Contudo, as Estações Clientes podem ser simples, a nível de comunicação via FTAM, e podem ter um esquema de acesso ao Sistema de Arquivos por parte da própria API. Introduz-se a unidade funcional VDRDB para realizar este mapeamento.

Formatos de Troca de Dados

O Projeto TOP oferece três tipos de formatos de arquivos:

a) CGM - "Computer Graphics Metafile" para troca de arquivos gráficos,

b) PDI - "Product Data Interchange" para troca de dados de definição de produtos,

c) ODI - "Office Document Interchange" para troca de documentos de escritório com texto, gráficos geométricos e "raster".

Para a criação de arquivos nestes formatos, precisa-se de Interfaces que:

a) ofereçam uma biblioteca de funções para manipular esses formatos,

b) executem o mapeamento de arquivos com formatos convencionais para os formatos definidos e vice-versa.

Dessa forma é possível definir um Modelo Geral de API que suporte apenas um Protocolo de Aplicação que englobe todas as particularidades da API para o MAP/TOP, para o FTAM e para formatos de troca TOP (Figura 2).

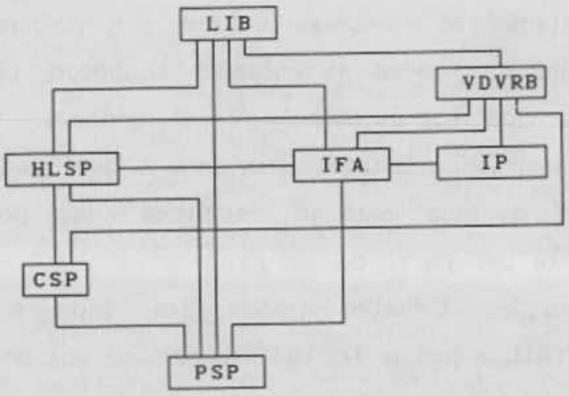

Figura 2 - Modelo Geral da API para suportar um Protocolo de Aplicação

2.3 - MODELO GERAL

Suponha-se que uma Entidade de Aplicação possua dois ASE's (Elementos de Serviço de Aplicação), por exemplo MMS e ACSE-Association Control Service Element. Para obter-se uma implementação conceitualmente correta, deve-se considerar que a API possua logicamente duas partes: uma relacionada com o Gerenciamento de Contexto (devido ao ACSE) e outra específica do ASE (devido ao MMS), chamadas GC e ESP respectivamente.

Na fase de estabelecimento da associação, o programa aplicativo fornece informação ACSE e específica do ASE para a API-GC.

À nível de API, o GC e o ESP interagem entre si, gerando uma primitiva para a Camada de Aplicação. Esta primitiva contém informação para o ACSE gerar um A-ASSOCIATE que possua no seu campo de informação um A-INITIATE gerado pelo ASE (no caso pelo MMS), (Figura 3a). Depois da associação estabelecida, o programa aplicativo chama funções da API-ESP para realizar serviços específicos do ASE (Figura 3b). Nos dois casos citados, o tráfego de informação, no sentido contrário é similar.

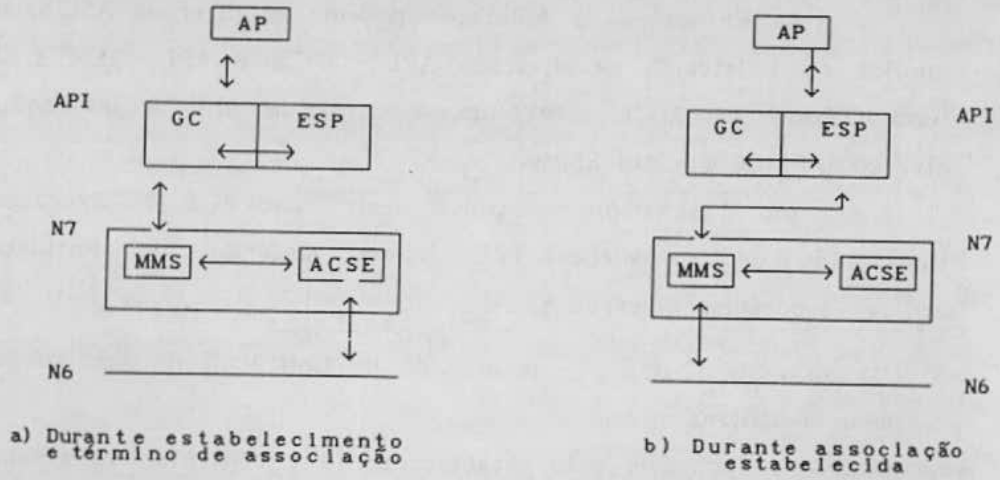

Figura 3 - Fluxo de informação

Como visto anteriormente, o GC e o ESP interagem entre si para gerarem uma primitiva para a Camada de de Aplicação. Cada um deles possui a sua LIB e o seu HLSP, CSP, IFA, IP e VDRDB. Duas funções de Controle de Objeto da Associação de Transmissão e Recepção (TAOCF e RAOCF, respectivamente) monitoram a interação (Figura 4).

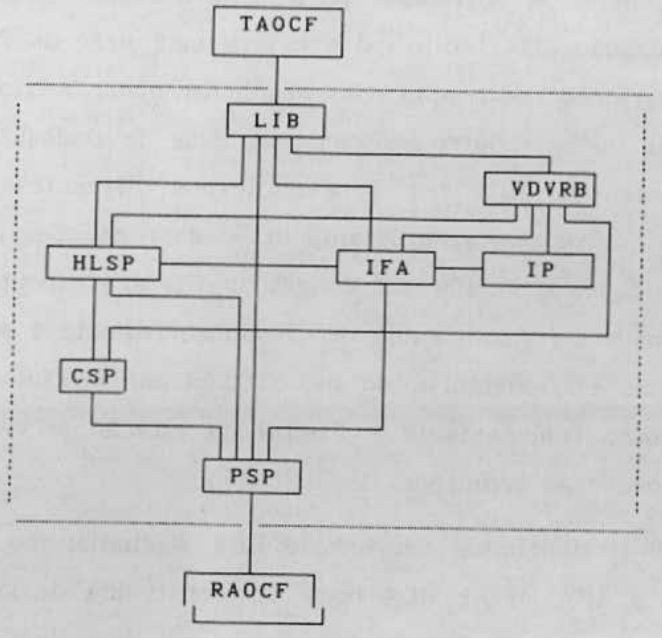

Figura 4 - Modelo Geral de API

As Entidades de Aplicação podem ter diversos ASE's, o que implica na existência de diversas API - GC's e API - ESP's. Neste caso, TAOCF e RAOCF controlam, à nível de API, a interação dos diversos API-GC's e API-ESP's.

Um exemplo típico deste caso é o de Processamento Distribuído de Transações(DTP). Nestes sistemas, as Entidades de Aplicação possuem diversos ASE's:

a) Um ou mais U-ASE's - protocolos de aplicação de serviços que o usuário utiliza;
b) ACSE - responsável pelo estabelecimento e controle das associações com os Sistemas Remotos;
c) TP-ASE - responsável pelo Gerenciamento do Diálogo;
d) ROSE - responsável pela geração de A-PDU's pertencentes ao ASE do Usuário (U-ASE);
e) CCR - responsável pela geração de A-PDU para efetivação de duas fases.

Neste caso, o TAOCF e o RAOCF controlam as interações entre as API-UASE, API-ACSE, API-TPASE, API-ROSE e API-CCR.

A título de exemplo, cita-se um outro modelo geral de API, Figura 4, para ser utilizado para o RDA (Remote Database Access) [6]. A padronização RDA facilita o acesso a uma Base de Dados Remota por um programa de aplicação residente numa estação de trabalho inteligente ou em outro sistema de Base de Dados. As duas estações comunicantes em RDA tem funcionalidades diferentes: no cliente existe um AP que pergunta e manipula dados no Servidor. O Modelo de Funcionalidades é similar ao Modelo do FTAM, e portanto, a API para o RDA pode ser construída de modo equivalente à API para o FTAM o que implica na existência do bloco funcional VDRDB na estação cliente e dos blocos funcionais IP e VDRDB na estação servidora, se a própria API responder às primitivas de indicação.

O Modelo de Serviço do RDA é similar ao Modelo do TP, e portanto a API para o RDA pode ser construída de modo equivalente à API para o TP em relação aos serviços. Isto implica na existência dos blocos funcionais TAOCF e RAOCF nas Estações Cliente e Servidora.

Os protocolos MMS, FTAM e RDA apresentam assimetrias entre as estações no que se refere a sua arquitetura do "software de aplicação" e conseqüentemente do "software de comunicação", segundo um modelo de estações clientes e estações servidoras. Para o Protocolo MMS a estação cliente pode ser uma estação supervisora de célula da manufatura com um "software aplicativo" mais complexo do que o das estações servidoras podem ser CLP's, CNC's, etc. Para o Protocolo RDA, a estação cliente pode ser uma estação de trabalho e a estação servidora um Sistema de Base de Dados que possui um "software aplicativo" mais complexo do que a primeira. Isto implica na existência de "softwares de comunicação" mais simples e outros mais complexos. No que se refere à Interface de Aplicação, API's mais simples podem ser construídas utilizando o mesmo Modelo Geral apresentado com a eliminação de alguns blocos funcionais e/ou algumas comunicações entre blocos, de acordo com os aspectos específicos de cada estação.

3. ASPECTOS DA IMPLEMENTAÇÃO DE API NO SISDI-MAP

O SISDI-MAP [7] foi implementado com o suporte de um núcleo de tempo-real onde vários processos são executados num mesmo ambiente multitarefa, utilizando-se do mecanismo de troca de mensagens para a sua comunicação, implementada através de filas tipo FIFO (Figura 5).

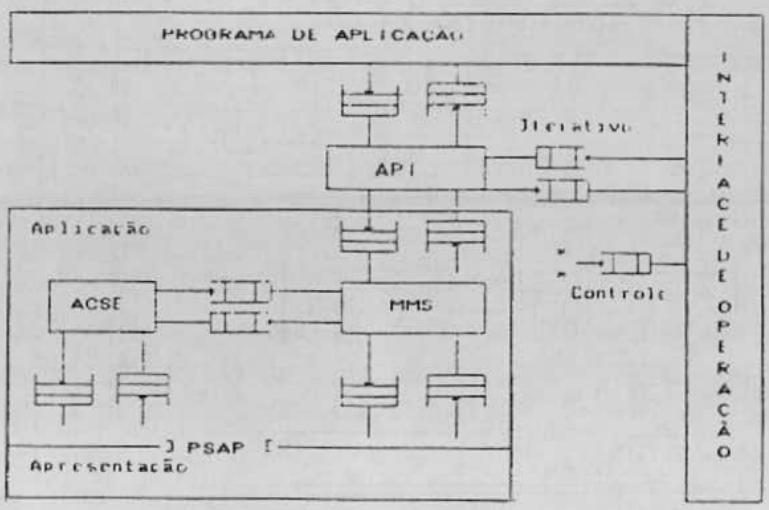

Figura 5 - Comunicação entre processos

Os protocolos MMS, FTAM e RDA apresentam assimetrias entre as estações no que se refere a sua arquitetura do "software de aplicação" e conseqüentemente do "software de comunicação", segundo um modelo de estações clientes e estações servidoras. Para o Protocolo MMS a estação cliente pode ser uma estação supervisora de célula da manufatura com um "software aplicativo" mais complexo do que o das estações servidoras podem ser CLP's, CNC's, etc. Para o Protocolo RDA, a estação cliente pode ser uma estação de trabalho e a estação servidora um Sistema de Base de Dados que possui um "software aplicativo" mais complexo do que a primeira. Isto implica na existência de "softwares de comunicação" mais simples e outros mais complexos. No que se refere à Interface de Aplicação, API's mais simples podem ser construídas utilizando o mesmo Modelo Geral apresentado com a eliminação de alguns blocos funcionais e/ou algumas comunicações entre blocos, de acordo com os aspectos específicos de cada estação.

3. ASPECTOS DA IMPLEMENTAÇÃO DE API NO SISDI-MAP

O SISDI-MAP [7] foi implementado com o suporte de um núcleo de tempo-real onde vários processos são executados num mesmo ambiente multitarefa, utilizando-se do mecanismo de troca de mensagens para a sua comunicação, implementada através de filas tipo FIFO (Figura 5).

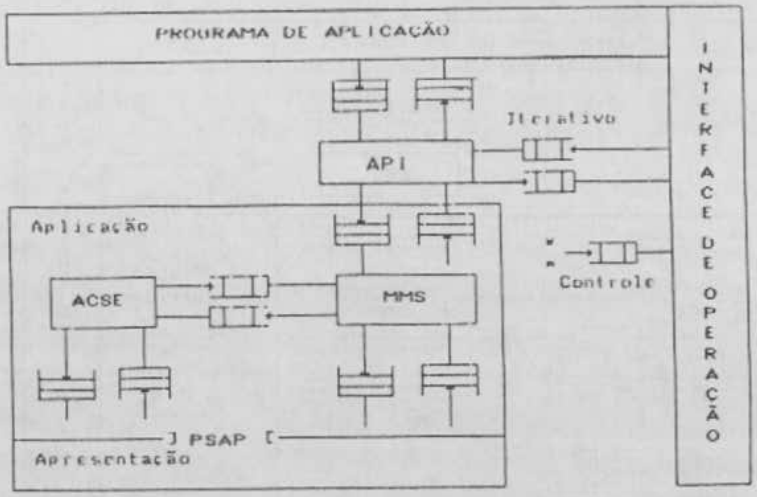

Figura 5 - Comunicação entre processos

O núcleo fornece primitivas para ativação e escalonamento dos processos (a pedido da Interface de Operação), criação de portas, tratamento de sincronização entre processos, gerenciamento de memória, etc.

A versão atual do SISDI-MAP 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).

O primeiro ponto abordado na implementação da API para o protocolo MMS foi o de compatibilizar as versões diferentes existentes no momento. A especificação do MMS encontra-se na versão DRAFT 6, enquanto que se dispunha da especificação da API dirigida à versão anterior do MMS-DRAFT 5.0 esforço foi no sentido de implementar as funções da API para atender às especificações do DRAFT 6.

A API para o SISDI-MAP foi construída utilizando o Modelo Geral de API apresentado. De acordo com o tipo de serviço solicitado pelo usuário (alto nível, baixo nível ou serviço respondedor), a API utiliza seus módulos funcionais (HLSP, CSP, PSP, IFA, IP e VDRDB), que se comunicam entre si para oferecerem os serviços correspondentes. Por outro lado, o usuário solicita serviços, ou de modo síncrono (fica bloqueado em seu processamento até a conclusão do serviço), ou de modo assíncrono (é liberado para outras atividades tão logo o pedido seja enfileirado, e ao término do serviço, é sinalizado).

A Tabela I mostra as rotinas internas à API de acordo com o tipo de chamada: funções auxiliares, confirmadas, não-confirnmadas, de alto nível e respondedoras.

FUNÇÕES AUXILIARES

1. recebe chamada de função
2. testa validade dos parâmetros
3. mapeia os parâmetros entrada/saída
4. retorna ao usuário

FUNÇÕES CONFIRMADAS

1. recebe chamada
2. testa validade dos parâmetros
3. cria identificação de chamada
4. mapeia parâmetros na primitiva de requisição
5. envia à Máquina de Protocolo
6. espera (com WAIT ou não)
7. recebe confirmação
8. mapeia primitiva em parâmetros de saída
9. retorna ao usuário

FUNÇÕES NÃO-CONFIRMADAS

1. recebe chamada
2. testa validade dos parâmetros
3. mapeia parâmetros em primitiva MMS
4. envia à máquina de Protocolo
5. Sinaliza envio ao usuário

FUNÇÕES DE ALTO NÍVEL

1. recebe chamada
2. testa validade dos parâmetros
3. cria identificação de chamada alto-nível
4. processa informações de alto nível
5. cria identificação de chamada
6. mapeia parte dos parâmetros em primitiva MMS
7. envia à Máquina de Protocolo
8. espera
9. recebe confirmação
10. mapeia primitiva em parâmetros de saída
11. verifica condição de término se não terminou, volta ao passo 4
12. mapeia parâmetros finais
13. retorna ao usuário

FUNÇÕES RESPONDEDORAS

1. Recebe autorização para aceitar indicações
2. Prepara função respondedora
3. Prepara o filtro (IFA)
4. Aguarda
5. Recebe indicação da MP
6. Mapeia primitiva de indicação
7. Determina a ação
8. Sinaliza usuário

Tabela I

A API é composta de duas rotinas: Trata_AP e Trata_MMS.

No procedimento trata_AP descrito na fig 6, mostra-se o tratamento das mensagens recebidas do programa de aplicação. Este procedimento recebe como parâmetro de entrada o endereço da mensagem recebida. Nesta mensagem existe um campo indicando qual o tipo de serviço que está sendo requisitado (tipo_serviço). Pela análise deste campo faz-se a chamada ao procedimento que realiza o serviço requisitado (AE_ACTIVATION, AE_DEACTIVATION, VREAD, VWRITE, 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. Se o serviço for executado assincronamente retorna-se ao AP uma mensagem dizendo que uma primitiva foi entregue ao "MMS", através de Envia_Mensagem (ap) e Note (ae_label).

```
VOID  TRATA_AP (ponteiro_interface_apl_ap) (
    SWITCH (tipo_serviço) ( /* serviços fornecidos ao AP */
      CASE mms_ae_activation:
        return_code     =  MM_AE_ACTIVATION( );
      CASE mms_connect:
          .
          .
          .
      CASE mms_vread:
        return_code     =  MM_VREAD( );
      CASE mms_vwrite:
        return_code     =  MM_VWRITE( );      )
IF    ( /*condição de erro */
   RESETA_RETURN_EVENT_NAME();
   ENVIA_MENSAGEM (ap); )
ELSE  (
   IF (    /* pedido assíncrono */
           /* confirmação de entrega da mensagem ao mms */
      ENVIA_MENSAGEM (ap);
      NOTE (ae_label)          ) )    /* libera semáforo do ap*/
```

Figura 6 - Procedimento TRATA_AP

No procedimento trata_MMS descrito pela Figura 7 mostram-se as mensagens recebidas do nó servidor. Este procedimento recebe como entrada o endereço da mensagem. Nesta mensagem existe um campo dizendo qual o tipo de primitiva envolvida (tipo_ primitiva). Novamente a análise de um campo de mensagem desencadeará um processamento particular (recepção de primitiva de confirmação - positiva ou negativa - ou de primitiva de indicação). Após o processamento da informação testa-se se a resposta é de um serviço síncrono ou assíncrono. Se a resposta é de um serviço síncrono o AP é imediatamente notificado sobre o término do serviço através de (3). Caso contrário, é necessário testar se o AP já realizou uma operação de espera de serviço. Se realizou, o AP é notificado sobre o término do serviço através de (1). Caso contrário o serviço é colocado numa fila de serviços prontos (2).

```
VOID TRATA_MMS (ponteiro_interface_api_mms)    (
   SWITCH (tipo_primitiva)          (
     CASE confirm_pos:
     CASE confirm_neg:
        ponteiro_interface_api_ap          = TRATA_CONFIRMAÇÃO (
                                             ponteiro_interface_api_mms);
     CASE indication:
        ponteiro_interface_api_ap          = TRATA_INDICATION (
                                            ponteiro_interface_api_mms);)
IF  (                  /* ponteiro_interface_api_ap e valido */
  IF  (                /* pedido assíncrono */
     IF  ( (1)         /* existe pedido de wait */
          ENVIA_MENSAGEM (ap); /* envia mensagem */
          NOTE (ae_label);          /* libera semáforo */          )
     ELSE ( (2)
        /* insere pedido na fila de serv prontos */
         /* seta return_event_name como pronto */           ) )
ELSE   ( (3)       /* serviço síncrono */
   ENVIA_MENSAGEM (ap); /* envia mensagem */
   NOTE (ae_label); ) ) )           /* libera semáforo */
```

Figura 7 - Procedimento TRATA_MMS

3.1 - FUNÇÕES DE GERENCIAMENTO DE CONTEXTO

Todo AP para utilizar os serviços de uma AE - Application Entity - deve ativá-la inicialmente através da chamada da função AE-ACTIVATION. Este serviço irá fornecer um identificador de invocação de AE (ae_label), a ser usado posteriormente nos pedidos de associação. A desativação da AE é realizada pela função AE_DEACTIVATION.

A camada de apresentação no MAP é orientada à conexão. Na camada de aplicação estabelecem-se associações entre AE's locais e remotas. Uma associação de aplicação utiliza uma conexão de apresentação e define um CEP no P_SAP da interface entre as camadas de aplicação e apresentação. À nível da API, a interação entre AP's deverá traduzir-se em associações dos ASE's MMS da camada de aplicação. Para estabelecer uma associação , o AP chama a função CONNECT da API. Quando a associação se estabelecer, a API retorna ao AP um identificador da associação (Connection_Id). Para esperar por pedido de associação por parte do AP remoto, o AP local chama a função LISTEN da API. Quando o pedido de associação chegar à API local, esta devolve ao AP o identificador da associação. O AP local aceita ou não a associação chamando a função ANSWER. O identificador da associação (Connection_Id) será utilizado como parâmetro de todos os serviços MMS solicitados pelo AP.

A API para monitorar as invocações de AE, as associações, os serviços confirmados pendentes e as invocações de AE que estão esperando por pedidos de associação possui as seguintes tabelas:

- AE_LABEL_MAP contêm o estado corrente dos AE_LABEL's válidos.

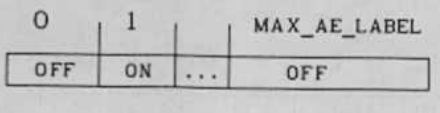

Figura 8 - AE_LABEL_MAP

- CONNECTION_ID_MAP, armazena o estado corrente dos números de associação 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: CONNECT, ANSWER e LISTEN.

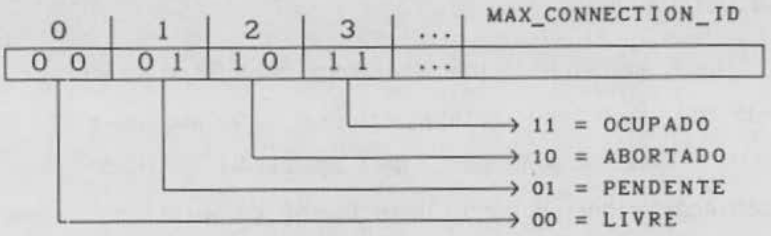

Figura 9 - CONNECTION_ID_MAP

- INVOKE_ID_MAP, 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 associação. Este número é único dentro do sistema.

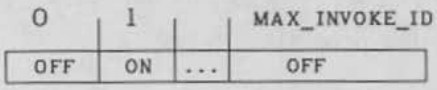

Figura 10 - INVOKE_ID_MAP

- INVOKE_AE_LISTEN_MAP, contém os ae_label's do sistema aptos a receber pedidos de abertura de associação. Esta Tabela é atualizada pela função LISTEN.

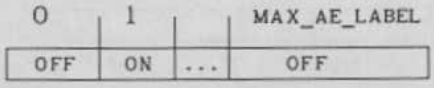

Figura 11 - AE_LABEL_LISTEN_MAP

A API possui também tabelas para controlar os nomes dos eventos associados às chamadas assíncronas de funções (para poder retornar ao AP quando do término do serviço solicitado) e para monitorar as invocações de AE que estão esperando pelo término de serviços remotos.

- RETURN_EVENT_NAME_MAP, é usada por todos os serviços fornecidos pela API e contém o estado corrente dos return_event_name's utilizados pelo sistema.

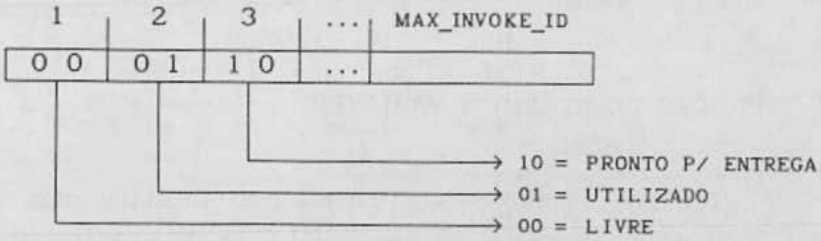

Figura 12 - RETURN_EVENT_NAME_MAP

- WAIT_MAP, indica se o AP está ou não no estado de "espera". Esta Tabela foi definida por motivos de sincronização entre os processos.

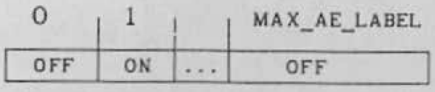

Figura 13 - WAIT_MAP

A API possui uma Tabela para indicar se o AP está apto ou não para o recebimento de uma indicação.

- **INDICATION_RECEIVED_MAP**, Esta Tabela é "setada" pela função INDICATION_RECEIVED e utilizada pelo processo quando do

recebimento de indicações do tipo ABORT, REJECT, CONCLUDE, CANCEL, UNSOLICITED STATUS e INFORMATION_REPORT.

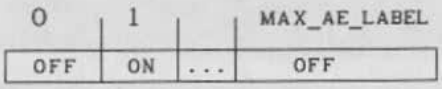

Figura 14 - INDICATION_RECEIVED_MAP

3.2 - FUNÇÕES DE ACESSO A VARIÁVEIS

Na implementação da API para o SISDI_MAP, além dos serviços de gerenciamento de conexão, foram implementados os serviços da classe de acesso a variáveis.

Dentre os serviços de acesso a variáveis, foi dado ênfase ao READ, WRITE (confirmados) e INFORMATION REPORT (não-confirmado). Uma aplicação cliente pode receber indicações de INFORMATION REPORT de outras aplicações. O usuário autoriza o recebimento através da função INDICATION RECEIVE, espera pelos dados e depois utiliza-se das funções auxiliares para interpretar os dados recebidos.

Para realizar pedidos de READ, WRITE e INFORMATION REPORT, funções auxiliares também são usadas antes das referidas funções principais. As funções auxiliares dão todo o suporte para o usuário compor as estruturas de dados complexas requeridas nas primitivas de requisição.

Um exemplo da seqüência de chamadas de funções auxiliares para posterior chamada da função READ, por exemplo, mostrando a simetria no par remoto é dado a seguir:

AP LOCAL	AP REMOTO
Antes da função principal:	Ao chegar a primitiva indication:
7. mm_new_asso	1. mm_interpret_asso
6. mm_makelist	2. mm_extract_list_element
5. mm_new_access	3. mm_interpret_access
4. mm_new_var_spec	4. mm_interpret_var_spec
3. mm_makelist	5. mm_extract_list_element
2. mm_new_partial access	6. mm_interpret_partial access
1. mm_new_restriction	7. mm_interpret_restriction

No AP remoto, são construídos os dados (mm_new_data) e, ao chegar ao AP local, na confirmação, esses dados são interpretados (mm_interpret_data).

Através dessas chamadas, as estruturas de dados complexas e aninhadas requeridas como parâmetros das primitivas de serviços podem ser construídas/desmontadas passo a passo e de modo recursivo, ou seja, definições são feitas a partir de definições anteriores.

Para evitar "overheads" adicionais de cópia de blocos de memória e transferência física dos blocos de uma chamada a outra, apenas um ponteiro é utilizado para referenciar às estruturas de dados já construídas nas chamadas anteriores.

Tomando como exemplo a construção da estrutura "Var_access_specification" requerida num Read_req, que especifique o acesso a várias variáveis, tendo algumas delas restrições de acesso (parte de um vetor ou subestrutura de um tipo estruturado), as seguintes chamadas devem ser feitas à API:

1: mm_new_restriction retorna a especificação da restrição no acesso de uma variável (faixa do vetor ou nome da subestrutura)

2: mm_new_partial_access forma a estrutura com as especificações de acessos parciais (uma só restrição ou uma lista de restrições), ou seja, uma definição de acesso alternado. Obs.: Uma nova definição

pode surgir para especificar o conjunto de acessos alternados.

3: mm_makelist constrói a lista de acessos parciais. São feitas tantas chamadas quantas forem os "acessos alternados",especificados na chamada 2. Essa lista pode ser usado em 2 para definir um novo acesso alternado.

4: mm_new_var_spec constrói uma especificação para cada uma das variáveis a ser acessada. Podem ser de quatro tipos: named (acesso por nome), unnamed (acesso por endereço), single (acesso por endereço e tipo) e scattered (acesso disperso). Obs: A construção do tipo scattered é feita com chamadas a mm_nscattaccess e mm_makelist.

5: mm_makelist agora, constrói a lista das especificações das várias variáveis a serem acessadas.

6: mm_new_access define a especificação do acesso, que pode ser através do nome da lista feita na chamada 5 ou da própria lista (seu endereço é passado).

7: mm_new_asso associa a especificação do acesso remoto (definida em 6) a endereços locais, onde serão armazenados os dados remotos (no caso de leitura).

4. CONCLUSÕES

É necessário apresentar aos Programas Aplicativos os serviços de comunicação de maneira amigável ("friendly"). Portanto, necessita-se ter Interfaces de Aplicação que garantam portabilidade dos programas aplicativos do usuário para ambientes diferentes e reduzam custos de treinamento de programadores que utilizam a interface em ambientes múltiplos, além de executar algumas tarefas que seriam de responsabilidade do usuário. Este conceito de Interface de Aplicação tem similaridades com os conceitos de Funções de Interação, UE-User Element e UA-User Agent da ISO.

As API's devem ser modeladas em função dos Protocolos de Aplicação utilizados e da complexidade da estação implementada. De

acordo com o Protocolo de Aplicação que se deseja implementar, a API deverá ou não implementar certos blocos funcionais, dependendo do tipo da estação, cliente ou servidora (modelo utilizado em muitos protocolos de aplicação) ou das particularidades operacionais. A complexidade da estação implica na implementação de todas ou de parte das funções suportadas pela API de acordo com o tipo de aplicação que essas estações permitirem. A implementação do SISDI-MAP é realizada a partir do modelamento e da modularidade propostos neste artigo.

REFERÊNCIAS

[1] ISO/DIS 7498 - "Information Processing System - Open System Interconnection - Basic Reference Model", Outubro/1982.

[2] GM-MAP 3.0 - "MMS Application Interface Specification" e Connection Management Interface Specification", Março/1986.

[3] MADEIRA, E.R.M.; e MENDES, M.J. Interface de Programas de Aplicação para o Protocolo MMS (RS-511), Anais do 3º CONAI, Setembro/1988.

[4] ISO/DIS 9506 - "Manufacturing Message Specification, Part I: Service Specification, Part 2: Protocol Specification", DRAFT 6, Maio/1987.

[5] ISO/DIS 8571 - "File Transfer Access and Management. Abril/1988.

[6] ECMA: Remote Database Access. Second Working Draft for a Standard, Dezembro/1986.

[7] PAGLIONI, A.J. e outros - SISDI-MAP: Sistema Didático do Protocolo e da Interface de Aplicação MMS do MAP - Seminário Franco-Brasileiro em Sistemas Distribuídos - Florianópolis - Setembro/1989.