Antenor Paglione Junior

Engenheiro Eletricista - Modalidade Eletrônica

Faculdade de Engenharia Elétrica - FEE

Universidade Estadual de Campinas - UNICAMP

Graduação em Dezembro de 1984

Este exemplar corresponde à redação final da tese defendida por Antenor Paglione Junior e aprovada pela Comissão Julgadora em 3 / 6 / 91.

Orientador

ASPECTOS DE ESPECIFICAÇÃO E IMPLEMENTAÇÃO DA ESTRUTURA DE MENSAGENS DA MÁQUINA DE PROTOCOLOS ACSE PARA O SISDI_MAP.

Dissertação apresentada à Faculdade de Engenharia Elétrica da UNICAMP como requisito parcial para obtenção do título de "Mestre em Engenharia Elétrica".

ORIENTADOR : Prof Dr. Manuel de Jesus Mendes

UNVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA

DEPARTAMENTO DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

JULHO DE 1991

Dedico à Helena e meus pais, com carinho, pela compreensão e incentivo nos momentos difíceis.

Agradeço ao Durval e Ivo, cuja paciência e obstinação me fizeram chegar ao final desta tese.

ÍNDICE

ÍNDICE

ÍNDICE

ÍNDICE DE TABELAS E FIGURAS

[TABELAS]

[FIGURAS]

ÍNDICE DE TABELAS E FIGURAS

1 - INTRODUÇÃO.

O projeto **MAP** ("Manufacturing Automation Protocol")<1> pode ser visto como um perfil funcional do Modelo *OSI* <2>, representando assim um corte vertical na norma , que tem por objetivo a interligação de sistemas computacionais de automação industrial ditos abertos , isto é , sistemas que independem de fabricante e tipo de equipamento para poderem se interconectar.

1.1 - Objetivo.

O objetivo principal desta tese é apresentar uma proposta de implementação prática do protocolo **ACSE** (**Association Control Service Element**") <3> dentro da filosofia adotada pelo grupo de desenvolvimento do "**Sistema Didático**" da Camada de Aplicação e da Interface de Aplicação do protocolo "**MAP**" <4>.

1.2 - Histórico.

O protocolo "*MAP*" encontra-se atualmente na versão 3.0 <4> e define em sua Camada de Aplicação , na parte específica denominada *ASE* ("*Application Service Element*") , os serviços e protocolos **RS-511** <5> , chamados serviços *MMS* ("*Manufacturing Message Specification*") <6>.

O protocolo "*MAP*"<1> utiliza na parte comum da Camada de Aplicação o protocolo *ACSE* <3> , e define, na interface com o usuário, o Protocolo de Interface de Aplicação - *API* ("*Application Process*

Interface") (7).

1.3 - SISDI_MAP.

O sistema didático *SISDI_MAP* (8) é caracterizado como um sistema multitarefa na execução de vários *"Application Process"* (*APs*) que se comunicam entre si.

Esses *APs* executam tarefas definidas para a Camada de Aplicação e suas interfaces no protocolo *MAP*.

O *SISDI_MAP* permite a comunicação entre os usuários (*APs* simulando equipamentos de controle de processos no ambiente de manufatura) através dos serviços *MMS* e tem por objetivos, facilitar a compreensão e auxiliar no estudo de implementação de protocolos, através de exercícios práticos no ensino de comunicação fabril.

O sistema completo está estruturado em software básico e aplicativo cuja arquitetura funcional está descrita em (8) e pode ser vista na figura 1.1.

1.4 - O Software Básico do Sistema Didático.

O software básico do sistema didático é composto de um núcleo que oferece características multi-tarefas e tempo real sobre uma máquina *IBM-PC*.

O núcleo (9) oferece uma interface padrão para acesso a seus serviços que, por sua vez podem ser construídos e compostos

conforme necessidades da aplicação no tocante a tamanho de código , tempo de execução , presença de serviços , etc.

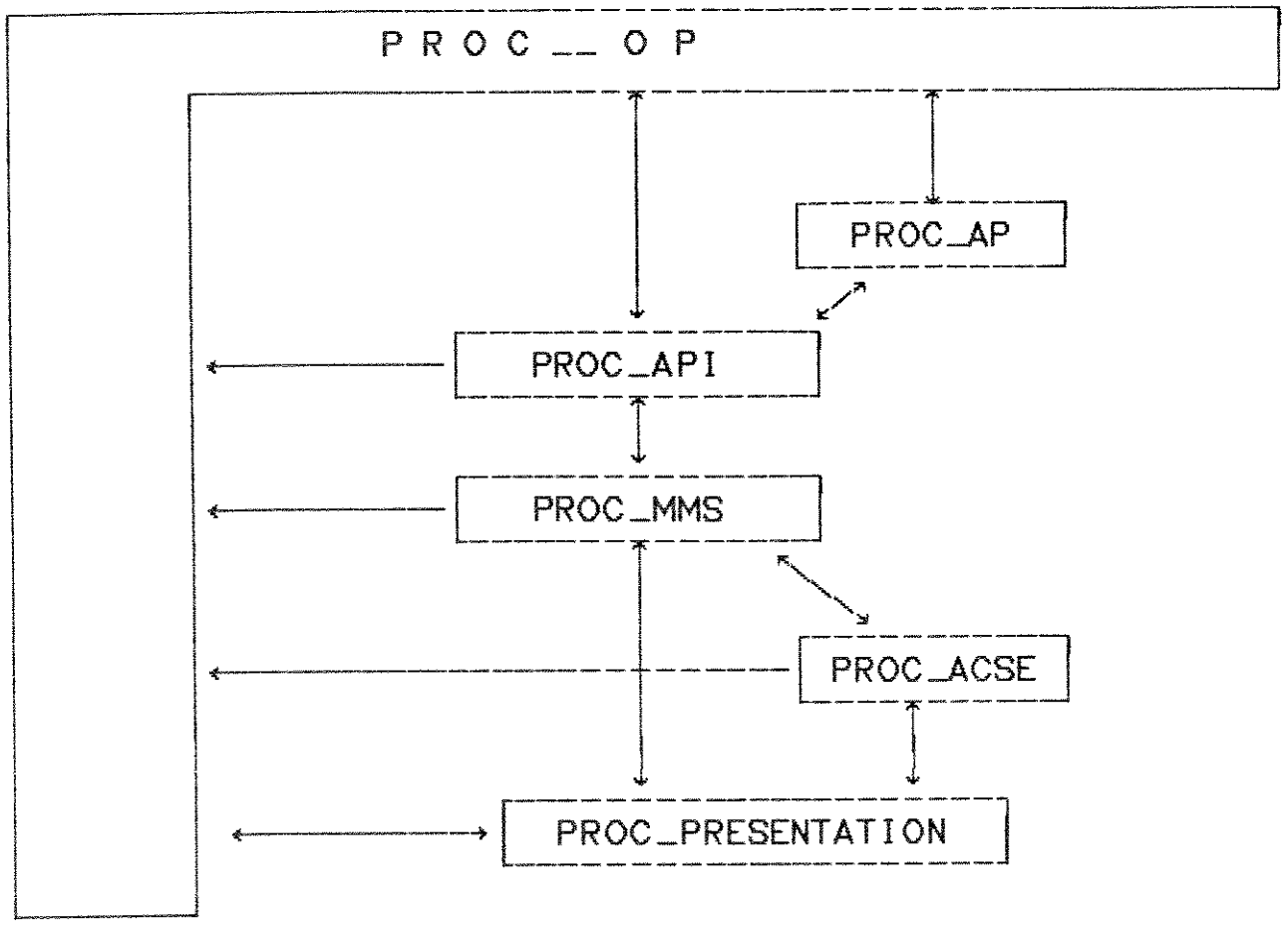


Figura 1.1 - Processos do SISDI_MAP

Também, são disponíveis no núcleo, serviços adicionais possibilitados pelo tipo de "hardware" (*PC*) utilizado.São eles que fazem o controle de acesso a serviços do sistema *"MS-DOS"* da máquina e a operação em *"IO"* (vídeo e teclado) com características

que possibilitam uma interface "homem-máquina" <10> adequada a aplicação do sistema didático.

O núcleo apresenta-se à aplicação na forma de uma biblioteca de funções , sendo que o produto final - núcleo + aplicação - é construido a partir de ligação destes dois módulos , resultando em um único módulo executável.

Sua arquitetura funcional está dividida em processos funcionais (grupos de código que fornece serviços relacionados a uma mesma entidade) tais como descritos abaixo.

Processo de Escalonamento de tarefas : Responsável pela execução de tarefas no núcleo;

Processo de filas : Permite a manipulação de filas, de forma uniforme, por todos os serviços do núcleo;

Processo de memória : Permite a alocação à aplicação de áreas de memória, controlando posteriormente sua liberação;

Processo de interrupções : Reprograma o relógio da máquina e fornece funções para controle de concorrência e para início efetivo da operação;

Processo de sincronização e comunicação : Fornece primitivas que manipulam "portas" (áreas de manutenção de apontadores para elementos genéricos) , com temporização associada;

Processo de semáforos : Mecanismo para sincronização de eventos e controle no acesso a recursos que operam com exclusão mútua;

Processo de I/O : O núcleo não fornece interfaces para memória de massa.É possivel a incorporação de funções que permitam visualizar ligações de dados do núcleo ou informações da própria aplicação , para fins de comunicação homem/máquina (inclusive com recursos gráficos de janelas , cor , etc)

1.5 - O Software Aplicativo do Sistema Didático.

O software aplicativo do sistema didático implementa os protocolos da Camada de Aplicação do *MAP* e suas interfaces , e simula o elemento usuário e as camadas inferiores do modelo *OSI* , permitindo assim a comunicação fim-a-fim de dois ou mais usuários *MMS*.

1.5.1 - Arquitetura Funcional.

A divisão da arquitetura funcional é feita nos seguintes processos funcionais:

- Processo de Operação de Usuário;
- Processo da *API* ;
- Processo do Protocolo *MMS*;
- Processo do Protocolo *ACSE*;
- Processo de Simulação da Camada de Apresentação <11>.

1.5.1.1 - Processo de Operação de Usuário.

CAPÍTULO 1 - INTRODUÇÃO

O processo de operação de usuário Proc_OP destina-se a prover uma interface de comunicação entre o operador do sistema didático e o Protocolo *MMS*, facilitando o acesso a dados e primitivas de serviço de protocolo , via *API*.

As funções básicas deste processo são as seguintes:

.Prover interface amigável com o operador / usuário , através da manipulação de funcões de terminal (como movimentação de cursor , funções de "*scroll*" , escrita e leitura de vídeo , controle de teclado , etc), complementadas por funções de tratamento de janelas (criação , destruição , movimentação de janelas) , menus e gráficos , simplificando o acesso do operador / usuário aos serviços oferecidos pelo sistema;

.Controlar o acesso à *API* , fazendo o papel de programa usuário do protocolo (*API*) , coordenando dados e primitivas necessárias a execução de um serviço oferecido pelo sistema;

.Gerenciar os sinais de controle de sinalização entre os elementos da Camada de Aplicação.Este pode ser considerado o programa principal da interface do usuário , pois é ele que executa todo o tratamento de sinalização , usando o pacote gráfico para informar o usuário sobre os eventos detectados no tratamento do protocolo.Este programapossui interface com todos os processos do *SISDI_MAP* , que estão encarregados de enviar sinais indicando o tipo de processamento que está sendo executado em cada um deles.De

posse destes dados , o processo Proc_OP mostra ao usuário os dados necessários , de acordo com as opções pré-definidas de visualização e controle.

.Permitir a introdução de erros (por exemplo em *APDUs* - "Application-Protocol-Data-Unit") entre protocolos fim-a-fim , para testes de situação de excessão.

1.5.1.2 - Processo da API.

O processo da *API* é responsável pela interface do programa do usuário (Programa Aplicativo , embutido no Processo de Operação do Usuário) com o Processo do Protocolo *MMS*.

O objetivo deste processo é obter uma portabilidade maior do programa aplicativo , através da colocação , à disposição do usuário , de funções de suporte e funções auxiliares específicas dos Serviços da Aplicação *MMS*, funcões estas que vão constituir uma biblioteca de funções.

Com isso, facilita-se o trabalho do usuário , poi o processo da API realiza tarefas que a princípio seriam de responsabilidade do programa aplicativo.

O processo da *API* é implementado em três blocos funcionais:

1) - Uma biblioteca de funções, chamada indiretamente pelos programas aplicativos, através de mensagens enviadas pelo Processo de Operação do Usuário, para requisitar serviços

desejados da rede;

2) - Um provedor de serviços de primitivas para enviar / receber primitivas para / do Processo do Protocolo *MMS*;

3) - Um provedor de serviços de alto nível e de serviços confirmados , que funcionam também como filtro para as primitivas de indicação.

O processo da *API* implementa o controle das invocações e suas correspondentes associações, através de tabelas indexadas pelos números das associações.

A execução das funções deste processo é monitorada pelo usuário do *SISDI_MAP* através do Processo de Operação de Usuário.

1.5.1.3 - Processo do Protocolo MMS.

O processo *MMS* é responsável pela execução das funções do provedor *MMS* , conforme <6>.

As funções deste processo estão relacionadas com as máquinas de estado definidas no Protocolo *MMS* , e são as seguintes:

.Tratar as primitivas de serviço do usuário , mapeando-as em *APDUs* a serem enviadas ao usuário remoto via primitivas do *ACSE* e da Camada de Apresentação;

.Tratar as *APDUs* corretas, recebidas do usuário remoto, via primitivas do *ACSE* e da Camada de Apresentacão

.Tratar as *APDUs* inválidas recebidas;

.Tratar as primitivas recebidas do *ACSE*, relativas ao contexto de aplicação.

A execução das funções desempenhadas pelo Processo do Protocolo *MMS* é monitorada pelo operador do sistema de usuário na recepção de eventos (primitivas) pelo protocolo do *MMS*.

O Processo do Protocolo *MMS* controla as associações estabelecidas e os serviços pendentes em cada associação, através de tabelas de contexto e tabelas de serviços pendentes em cada contexto.

Esta tese especifica a implementação da estrutura de mensagens de comunicação entre o **ACSE** , os protocolos **MMS** e o simulador da Camada de Apresentação , bem como discorre sobre o interrelacionamento com a **API**.O Capítulo 2 descreve a Estrutura da Camada de Aplicação e o ACSE, fornecendo conceitos básicos sobre a Camada de Aplicação segundo o padrão **ISO/IEC DIS 9545** <12> e conceitos sobre o **ACSE** (serviços e protocolo).

No Capítulo 3 é apresentada a Especificação de Implementação do **ACSE**. Neste Capítulo será detalhada a estrutura das mensagens.

O Capítulo 4 trata da interação do **ACSE** com o **MMS**, mostrando um exemplo prático no projeto **SISDI_MAP**.

No Capítulo 5 estão as conclusões desta tese. No Capítulo 6

estão referências bibliográficas.

CAPÍTULO 2 : ESTRUTURA DA CAMADA DE APLICAÇÃO E O ACSE

2 - Estrutura da Camada de Aplicacão e o ACSE.

Os padrões *OSI* definidos pela *ISO* pretendem dar suporte às aplicações que requerem atividades de processamento em dois ou mais *SISTEMAS ABERTOS REAIS*.Os padrões *ISO* para a Camada de Aplicação definem procedimentos para dar suporte ao processamento de informação distribuída.

A Camada de Aplicação é suportada pelas camadas inferiores , destacando-se a Camada de Apresentação que contém facilidades para representar a informação trocada entre as Entidades de Aplicação (*AEs*), e a Camada de Sessão que contém mecanismos para controlar a interação entre *AEs* (sincronismo , modo de comunicação , etc).

A Camada de Aplicação difere das outras do modelo *OSI* sob vários aspectos.As entidades *(AEs)* da Camada de Aplicação são formadas pela coleção de Elementos de Serviço de Aplicação *(ASEs)* - MMS , FTAM ,JTM , MHS , etc - ; sendo que cada qual é definido como um conjunto de serviços e protocolos padrão.Na figura 2.1 mostra-se uma *AE* formado pelos *ASEs MMS , FTAM ,MHS.*

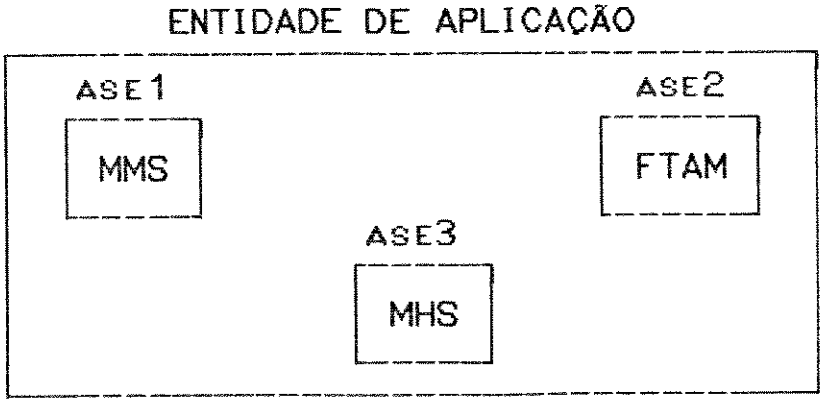


FIGURA 2.1 - Exemplo de uma composição de ASEs formando uma AE

Esses *ASEs* são combinados em várias formas , para formar vários tipos distintos de *AEs*.Como a Camada de Aplicação , sendo a de nível mais alto no modelo *OSI* , não provê serviços a uma camada mais alta , não existe nela o conceito de conexão mas sim o de Associação de Aplicação.Conceitualmente uma associação de aplicação , ou uma associação somente , representa um relacionamento cooperativo entre duas *AEs* com o propósito de troca de informação e coordenação de sua operação conjunta.

CAPÍTULO 2 : ESTRUTURA DA CAMADA DE APLICAÇÃO E O ACSE

2.1 Conceitos Básicos

A cooperação entre *SISTEMAS ABERTOS REAIS* ,para o processamento de informações , é modelada em termos de interações entre processos de aplicação (*APs*) residentes nesses *SISTEMAS*. Um *AP* é uma representação abstrata daqueles elementos de um *SISTEMA ABERTO REAL* que executam processamento de informação para uma particular aplicação.Dependendo da natureza da aplicação , um *AP* pode somente necessitar de uma comunicação intermitente com outro *AP* .

2.1.1 Processo de Aplicação (AP)

Um *AP* representa um conjunto de recursos , incluindo recursos de processamento , dentro de um *SISTEMA REAL ABERTO* que pode ser usado para executar um determinado processamento de informação.

Um *AP* requer que suas interações com outros *APs* sejam organizadas em um determinado número de invocações.Essas invocações podem ser executadas ao mesmo tempo ou podem ser o resultado de várias invocações seguidas , ou seja , de forma serial .Invocações de forma serial significam que o resultado de uma serve como parâmetro de entrada para a outra.

A atividade de um dado *AP* é representada por uma invocação de AP("*AP-invocation*").A cooperação entre *APs* , localizados em sistemas distintos ,é obtida pelo relacionamento entre os "*AP-invocations*".

Em um dado instante, um *AP* pode estar representado por nenhuma , uma ou mais "*AP-invocations*". Isto significa que um *AP* pode ,em um determinado instante de tempo ,não ter nenhum processamento de informação com outro *AP* , ou ter diferentes processamentos de informação com um mesmo *AP* ou com *APs* diferentes.

Uma "*AP-invocation*" que está interagindo com duas ou mais "*AP-invocations*" , isto é , dois ou mais sistemas abertos distintos ,é responsável pela coordenação das interações, se necessário.A figura 2.2 mostra o relacionamento de *APs* em sistemas reais distintos, podendo-se notar a existência de zero ou mais invocações de *APs*.

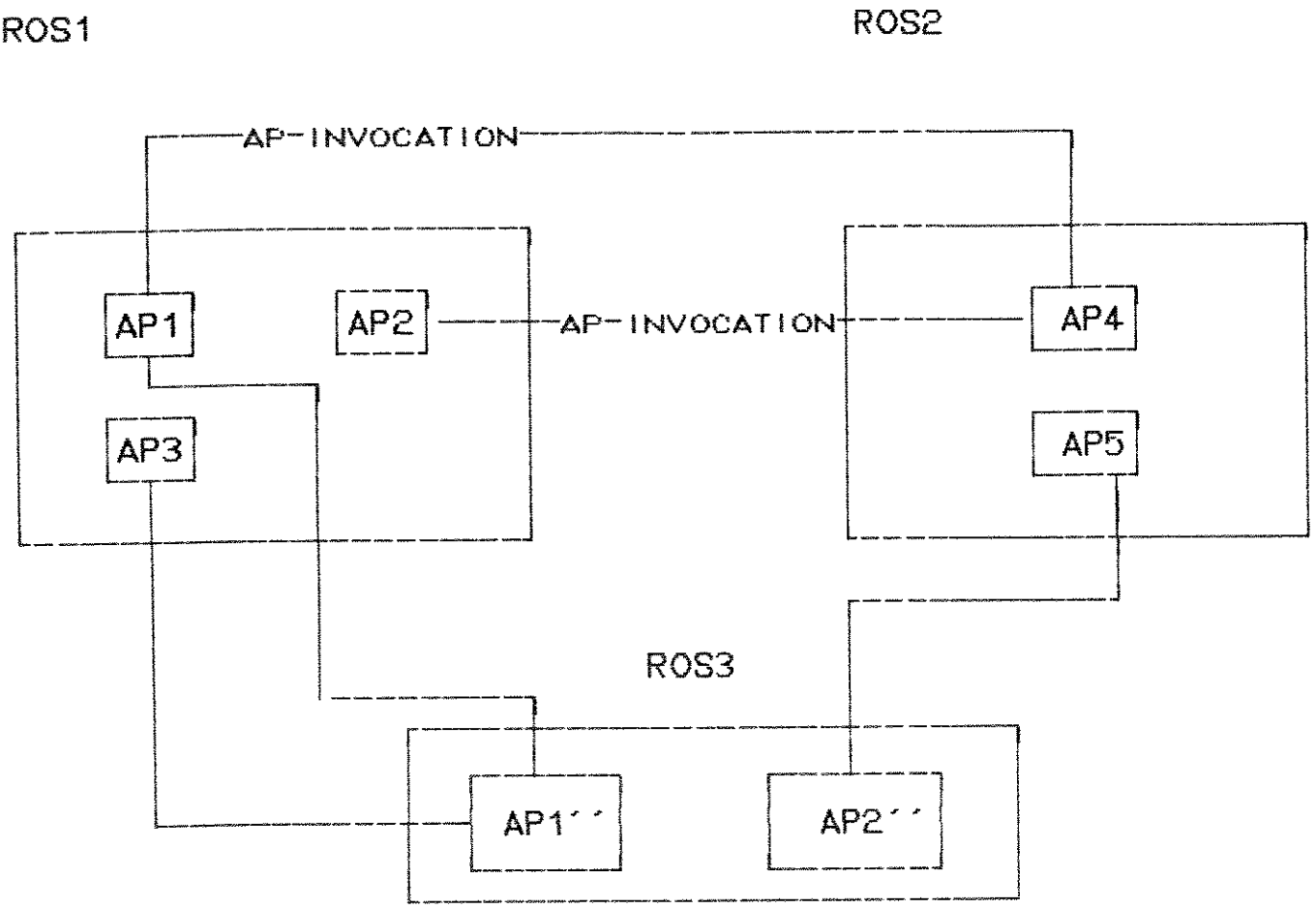


Figura 2.2 - Representação de invocações de APs entre ROSs ("Real Open Systems")

2.1.2 Entidades de Aplicação (AEs)

Os aspectos de um *AP* que devem ser considerados para o propósito de um processamento de informação segundo o modelo *OSI/ISO*, são representados por uma ou mais *AEs*. Uma *AE* representa um conjunto de capacidades de comunicação *OSI* de um dado *AP*. Essas "capacidades" de comunicação estão contidas nos vários *ASEs* (*MMS* , *MHS* , *FTAM* , *etc*) que, por sua vez, compõem uma dada *AE*.

Na figura 2.3, pode-se observar que diferentes *APs* (AP1 e AP2) podem ser representados por *AEs* do mesmo tipo (neste caso AE1), e também que um *AP* pode ser representado por um conjunto de *AEs* que não são todas necessariamente do mesmo tipo (neste caso, AE2 para o AP1 e AE3 para o AP2).Entretanto, uma *AE* representa um e somente um *AP* no ambiente *OSI*

Figura 2.3 - Representação de APs

A interação entre *AEs* do mesmo tipo (de modo a executarem um processamento de informação) é representada por uma "*AE-invocation*" .Isto envolve atividades específicas de comunicação entre *APs* distintos através de uma "*AP-invocation*". Em um sistema de interconexão aberto , uma ou mais "*AE-invocations*" representam os aspectos básicos de uma "*AP-invocation*".Na figura 2.4 mostra-se que, em uma única invocação de *AP*, pode-se ter diferentes *AEs* (em diferentes sistemas reais abertos) processando informação através de invocações de *AE*.Um exemplo prático seria dois "mainframes" processando informações de tratamento de mensagem e troca de documentos eletrônicos (AE1) e MAP (AE2), simultaneamente.

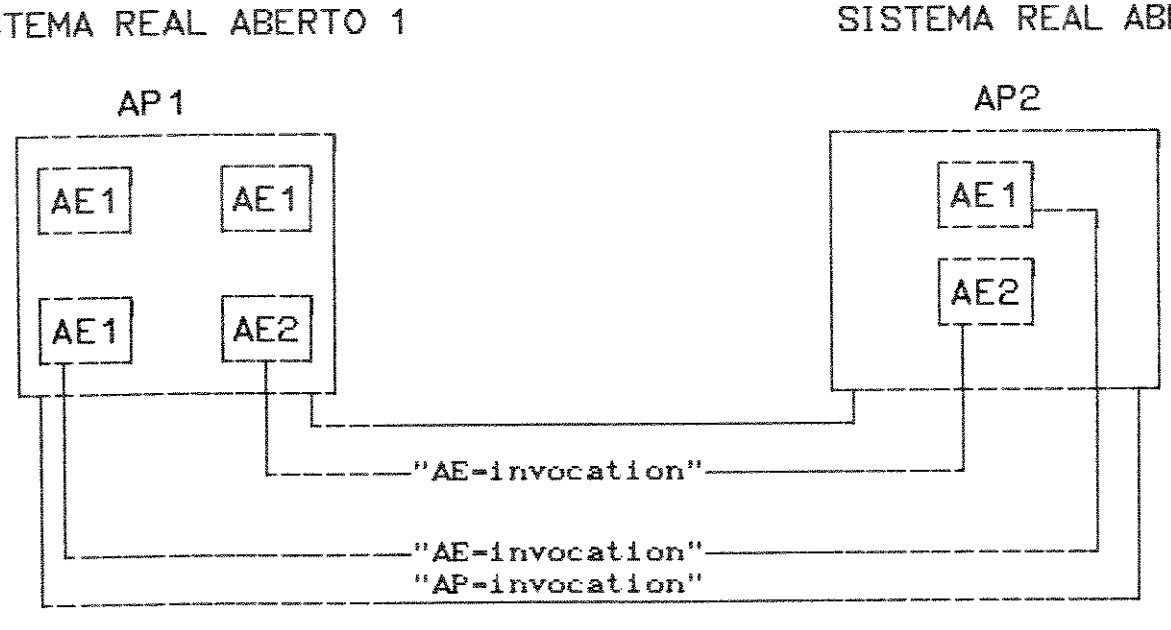


Figura 2.4 - Invocação de Entidade de Aplicação

O tempo de vida de uma "*AE-invocation*" é determinado pela "*AP-invocation*" que ela representa no ambiente *OSI*.

Num dado instante, podem existir zero , uma ou mais "*AE-invocations*" representando uma "*AP-invocation*". Isto significa que existem dois processos localizados em dois sistemas abertos distintos com todos os níveis inferiores setados , isto é , toda a capacidade de comunicação necessária à troca de dados entre os dois sistemas está configurada , e em um determinado instante de tempo, nenhum , um ou mais serviços estão sendo executados entre os sistemas.

Uma "*AE-invocation*" modela as funções de comunicação, juntamente com o estado da informação associada, para uma determinada atividade de comunicação de uma "*AP-invocation*".

As "*AE-invocations*" relacionam-se através de associações de aplicação ("*AAs*") e é através dessas *AAs* que progridem as atividades de comunicação de uma "*AP-invocation*".

Como pode ser visto na figura 2.5, uma "*AE-invocation*" pode , num dado instante ,estar representada por várias *AAs*.Também, neste caso, uma "*AE-invocation*" pode ,em um determinado instante de tempo, estar processando nenhuma , uma , ou várias aplicações , significando que ,apesar de existir uma "*AE-invocation*" entre as entidades AE1 e AE2 , em um instante de tempo, pode não haver nenhum serviço sendo processado entre elas ou , de outro modo , podem existir vários serviços sendo executados ao mesmo tempo.

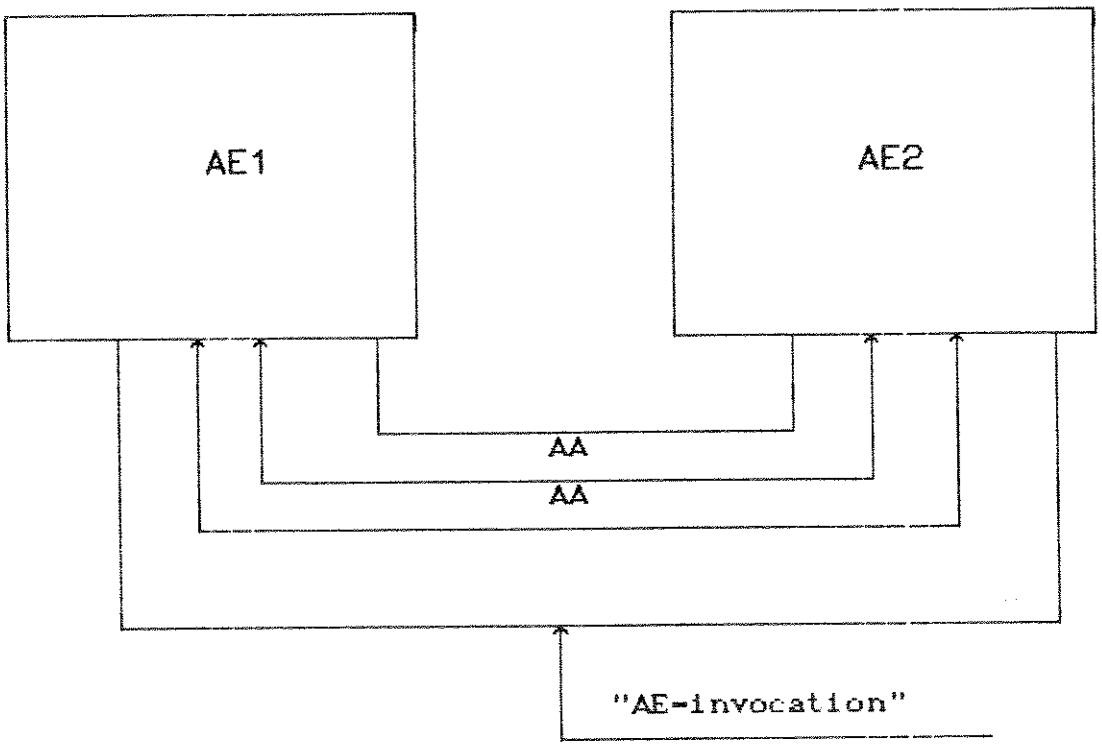


Figura 2.5 - Associação de aplicação

2.1.3 - Elementos de Serviços de Aplicação ("ASEs")

Um *"ASE"* é um conjunto de funções que provê capacidades de comunicação *OSI* para o interfuncionamento de *"AE-invocations"* dentro de um propósito específico.

A capacidade de comunicação de um *ASE* é definida pela especificação de um conjunto de **APDUs** (*"Application-Protocol-Data-Units"*) e o procedimento para governar o seu uso.

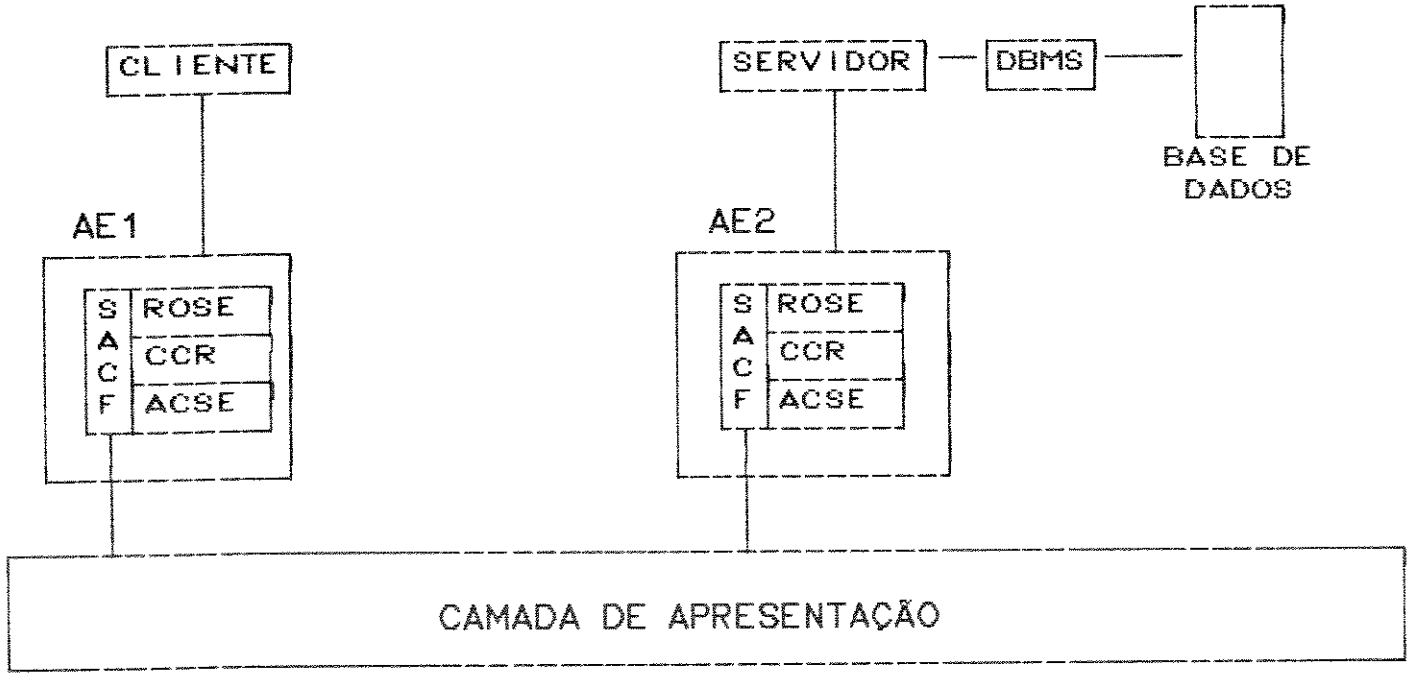
Uma *AE* pode ser composta de um ou mais *ASEs* de diferentes tipos, de maneira a obter uma capacidade específica de

comunicação para um propósito particular. Na figura 2.6, mostra-se um modelo de serviços para o protocolo padrão "*RDA - Remote Database Access*"<13> .Com o padrão proposto para o RDA, quer-se facilitar o acesso de "workstations" inteligentes a bases de dados remotas.Dentre os escopos para a padronização do RDA, destaca-se que a AE que dará suporte à comunicação entre o cliente e o servidor é composta de várias *ASEs (ROSE , CCR , ACSE)*.

No ambiente OSI ,a comunicação entre APs é representada em termos de comunicação entre um par de AEs usando a Camada de Apresentação.A comunicação entre AEs é, na maioria das vezes, interativa.Tipicamente , uma entidade requer que uma determinada operação seja executada pela outra entidade (por exemplo, o pedido do cliente ao servidor para que este lhe retorne um determinado dado de sua Base de Dados) , a outra , por sua vez , tenta executar a operação e retornar o resultado obtido.O objetivo do protocolo *ROSE* <14> é o de propiciar serviços que sirvam como veículo para dar suporte a essas aplicações interativas.

Uma aplicação distribuida é representada pela cooperação de AEs.Onde mais que duas AEs estão envolvidas (como é o caso típico de um RDA), deve existir em geral mais de um Sistema Real Aberto envolvido.Os usuários dos Sistemas Abertos requerem que as aplicações que são ativadas em conjunto com outros Sistemas Abertos sejam completadas com sucesso ,a despeito das falhas que possam ocorrer ; isto requer que haja uma coordenação entre todos os Sistemas envolvidos.O protocolo *CCR (Commitment , Concurrency*

and Recovery) <15> provê os meios para coordenar as atividades nas várias interações entre os Sistemas Abertos.Esse conjunto de ASEs proporciona a capacidade de comunicação necessária às AEs para que possam executar operações em Base de Dados distribuídas.O ACSE , neste caso, é o ASE que provê um meio, consistente e único, para o estabelecimento e término de todas as associações de aplicação envolvidas e será objeto de estudo detalhado nesta tese.



SACF - Funcão de Controle de Associacão Simples

DBMS - Sistema de Gerenciamento de Base de Dados

ROSE - Elemento de Serviço para Operação Remota

CCR - "Commitment , Concurrency and Recovery service element"

Figura 2.6 - Composição de ASEs para o serviço RDA

2.1.4 - Associação de Aplicação ("AA")

Uma *AA* é um relacionamento cooperativo entre duas "*AE-invocations* " para o propósito de comunicação de informação e coordenação de sua operação conjunta.Esse relacionamento é formado pela troca de *APCIs* (" *Application-Protocol-Control-Information* ") usando os serviços da Camada de Apresentação.

Quando a execução de uma determinada aplicação requer a iteração entre duas *AEs* , uma ou mais *AAs* são estabelecidas entre as invocações das duas entidades de aplicação.Uma invocação de AE pode suportar várias *AAs* simultaneamente.Isto significa que ,em um determinado instante de tempo ,em uma interação entre duas AEs distintas podem, estar sendo executadas várias aplicações ; por exemplo , troca de arquivos (através do ASE *FTAM*) , controle de células na manufatura (através do ASE *MMS*) e troca de mensagens interpessoais (através do ASE *MHS*).O término de uma *AA* resulta da ação das invocações de AE relacionadas.Isso pode , por exemplo , ser consequência da resposta a uma falha detectada na comunicação.

2.1.5- Contexto de Aplicação

Para efetivamente haver troca de informação em uma *AA* , o par de invocações de AE deve mutuamente conhecer , e seguir , um

conjunto de regras que governa a troca. Este conjunto de regras é chamado de *Contexto de Aplicação* da *AA*. Este conjunto cobre tudo o que é relevante com respeito à troca de informação , necessária para suportar as atividades das invocações de AE em uma *AA*. Como alguns pontos relevantes destacam-se o endereçamento do chamado e do chamador , oendereçamento da Camada de Apresentação e requerimentos necessários à comunicação definidos para a Camada de Sessão.

Uma *AA* tem um e somente um contexto de aplicação ; entretanto , o conjunto de regras que executam o contexto de aplicação em uma *AA* pode ser alterado durante o tempo de vida de uma *AA*.

Um contexto de aplicação inclui as regras que descrevem o conjunto de características que devem ser conhecidas por ambas as invocações de AE , o relacionamento entre essas características , as ações a serem executadas sobre elas , e os estados permitidos ou proibidos de serem atingidos.

Um exemplo de como esses parâmetros devem ser conhecidos pode ser visto na análise do serviço "*INITIATE*" do *MMS* .

Tem-se na tabela 2.1 os parâmetros do serviço " INITIATE " que são usados para o estabelecimento do contexto de aplicação do MMS e que permite que os usuários do MMS troquem informações entre dois Sistemas Abertos.

Conformance : Con1 , Con2 Parameter Name	req	ind	rsp	cnf	cbb
Argument	M	M			
List of Version Numbers	M	M			
Proposed Max Message Size	U	U			
Proposed Max Serv Outstanding Calling	M	M			
Proposed Max Serv Outstanding Called	M	M			
Proposed Data Structure Nesting Level	U	U			
Proposed Horizontal CCB	M	M			
Client Vertical CBB Calling	M	M			
Server Vertical CBB Calling	M	M			
Result(+)			S	S(=)	
Negociated Version Number			M	M(=)	
Negociated Max Message Size			U	U(=)	
Negociated Max Serv Outstanding Calling			M	M(=)	
Negociated Max Serv Outstanding Called			M	M(=)	
Negociated Data Structure Nesting Level			U	U(=)	
Negociated Horizontal CCB			M	M(=)	
Client Vertical CBB Called			M	M	
Server Vertical CBB Called			M	M	
Result(-)			S	S(=)	
Error Type			M	M(=)	

Tabela 2.1 - Parâmetros do Serviço "INITIATE"

Pode-se notar da tabela 2.1 que algumas "necessidades" para o interfuncionamento das entidades pares do MMS são negociadas na fase de obtenção do contexto de aplicação.O parâmetro " *Proposed Max Message Size* " , por exemplo , indica um número de caracteres máximo , proposto pela entidade de aplicação que deseja estabelecer uma AA com a entidade par , e deve estar contido numa *SSDU ("Session Service Data Unit")*.

Assim como este , outros parâmetros são trocados e negociados

entre as entidades pares desejosas de estabelecerem uma AA . Por exemplo , o parâmetro " *List of Version Numbers* " , indica as versões vigentes do protocolo e o parâmetro *"Proposed Max Serv Outstanding Calling"*, o número máximo de serviços pendentes em que o chamador pode estar esperando por resposta.

O conjunto de regras em um contexto de aplicação pode também incluir:

a) especificações da estrutura lógica da informação a ser trocada ou referenciada;

b) especificação das dependências de invocação entre os *ASEs* , além daquelas subordinações contidas internamente às especificações dos *ASEs* ;

c) regras com respeito à seleção e uso de características opcionais dos *ASEs* ;

d) qualquer regra adicional , além daquelas contidas nas especificações do *ASE* , governando a sequência do uso das primitivas de serviço e, em consequência, a sequência de *APDUs* , para cada *ASE* ;

e) regras de coordenação da operação dos *ASEs* , tais como regras para o interfuncionamento dos serviços de *"request"* e *APDUs* de diferentes *ASEs* ;

f) regras com respeito ao mapeamento de *APCI* de *ASEs* em serviços da Camada de Apresentação , ou de outros *ASEs* ;

g) designações de funções de aplicação , tais como :funções de diretório para a aplicação e as regras governando o

seu uso .

Um contexto de aplicação pode incluir regras que habilitem as invocações de AE a coordenarem suas atividades de comunicação com respeito a várias AAs. Um contexto de aplicação que é aplicado a uma AA, é determinado durante o estabelecimento da associaçãoda seguinte forma :

a) pela identificação de um contexto de aplicação pré-existente ; ou

b) pela transferência da atual descrição do contexto de aplicação.

Em particular, um nome pode ser usado para identificar um contexto pré-definido.

Na lista de parâmetros do serviço *"A-Associate"* do protocolo *ACSE* observa-se a existência do parâmetro *"Application Context Name"*.Esse parâmetro irá identificar univocamente o contexto de aplicação que está sendo proposto pelo requisitante da AA.O respondedor retornará o mesmo nome ou outro que , a partir do momento do estabelecimento da AA e, consequentemente, da definição do contexto a ser utilizado ,estará associado à AA, identificando-o enquanto existir.

Uma vez definido o contexto de aplicação e seu uso, estará definido também o comportamento da comunicação entre as invocações de AE sobre uma associação de aplicação.Os *ASEs* e a definição do seu uso em um contexto de aplicação devem garantir uma perfeita cooperação com a Camada de Apresentação.

Quando uma invocação de AE suporta várias AAs concorrentes , não existe nenhuma obrigação estrutural de que cada AA utilize o mesmo contexto de aplicação.

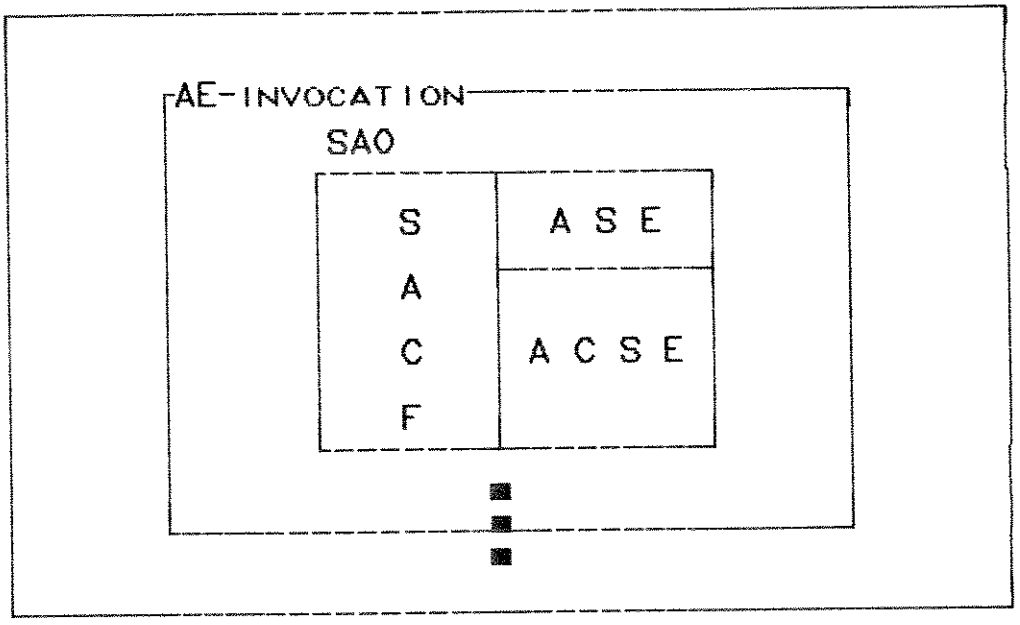
O que se convencionou chamar de contexto de aplicação não diz somente respeito aos parâmetros trocados pelas primitivas de serviço de estabelecimento de uma associação de aplicação deste ou daquele *ASE* : o contexto de aplicação é uma idéia mais ampla.É um conjunto de regras , parâmetros , relacionamento entre esses parâmetros , regras com respeito a seleção e uso de parâmetros opcionais , a especificação da estrutura lógica da informação , etc ; enfim, um conjunto de fatores que constam dos documentos de especificação de Serviços e Protocolos de um determinado *ASE* e que influem diretamente na determinação do contexto de aplicação.

2.1.6 - Objeto de Associação Simples (SAO)

Um "*Single Association Object*" é um componente de uma invocação de AE que modela as funções e o estado da informação relacionada à operação de uma *AA*.

Um único *SAO* contém um ou mais *ASEs* (um dos quais é sempre o ACSE - "Association Control Service Element" <3>) e uma Função de Controle para uma Associação Simples (SACF).A figura 2.7 exemplifica o caso em que uma "*AE-invocation*" está executando uma única associação de aplicação em um dado instante de tempo e, portanto, contém somente um *SAO*.O *SAO* contém dois *ASEs* (ACSE e um

outro ASE qualquer) e o *SACF* , o qual modela o uso das regras definidas no contexto de aplicação que dizem respeito às interações entre os dois ASEs e, também, coordena a interação com com a camada de Apresentação.A *ISO* ainda não possui um protocolo definindo o *SACF* ; entretanto, ao se observar o perfil funcional do *MAP* vê-se que existe um protocolo a nível de aplicação, chamado **API** <7> que coordena a interação entre o ACSE , o MMS e a Aplicação.Isto leva a crer que, apesar de a ISO não ter definido o papel do SACF, a GM na definição do perfil funcional do MAP, já prevê a necessidade de um protocolo com funções semelhantes as definidas para o SACF, que é o API.Muitas das funções de controle de AA que deveriam ficar a cargo do ACSE estão efetuadas pelo API, tais como, controle do número máximo de *"AE-invocations"* e de AAs associadas a essas AE_invocations; serão mais profundamente exemplificadas no capítulo 3 (três) desta tese.



■ - Associação de aplicação

SACF - Função de Controle de Associação única

SAO - Objeto de Associação Única

ASE - Elemento de Serviço de Aplicação

ACSE -Elemento de Serviço de Controle de Associação

Figura 2.7 - Invocação de AE única

Em um dado instante de tempo, uma "*AE-invocation*" pode possuir mais de um SAO e a combinação dos ASEs dentros desses SAOs será um pouco diferente , pois, neste caso, estar-se-á trabalhando com múltiplas associações.A figura 2.8 mostra como ficará a composição de SAOs neste caso.

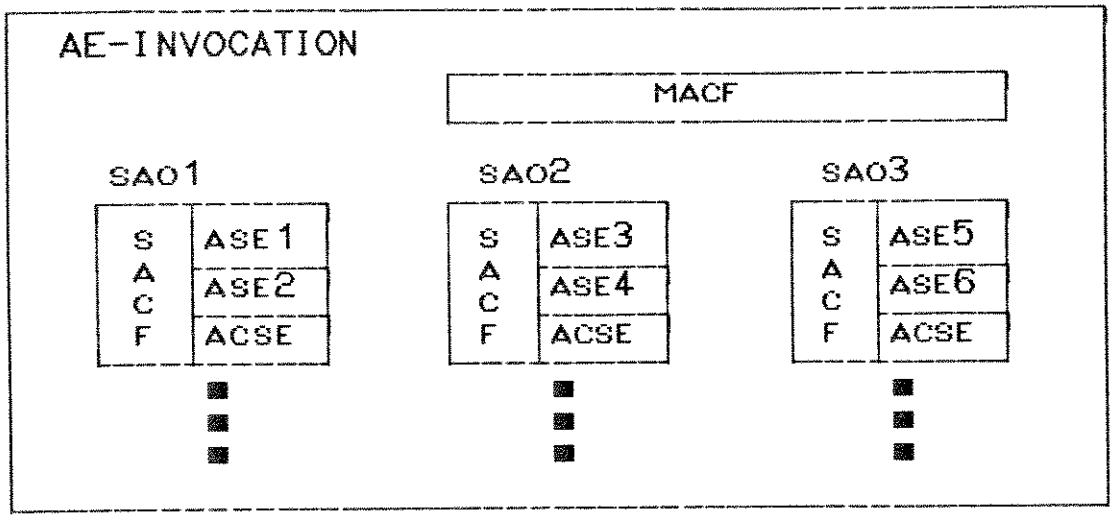


Figura 2.8 - AE-invocation com múltiplos SAOs.

Na figura 2.8 tem-se um caso mais complexo em que uma *AE-invocation* está simultaneamente envolvida em três associações de aplicação e, consequentemente, contém três SAOs.

A primeira das três associações está sendo executada de uma maneira independente das atividades envolvendo as outras duas SAOs.A segunda e terceira associações estão sendo executadas de maneira a exigir uma coordenação entre os dois SAOs envolvidos.Desta forma, existe a figura do *MACF* ("*Multiple Association Control Function*") que controla esses dois SAOs.

Em atividades envolvendo múltiplas AAs torna-se necessário o controle de certas características , por exemplo:

a) sequenciamento das atividades envolvidas

nas diferentes associações;

b) manter a consistência no relacionamento entre as atividades nas diferentes associações.

O conjunto de funções de aplicação que controla essas características está localizado no MACF.A execução dessas funções de coordenação feitas pelo SACF de maneira local , podem necessitar a presença de certas regras na definição do contexto de aplicação.Por exemplo , regras de sincronização podem ser incluídas na definição do contexto de aplicação de modo a permitir a uma AE-invocation coordenar atividades envolvendo várias associações de aplicação.

2.1.7 - Funções de Localização

Já foi vista a importância que o contexto de aplicação tem em uma *AA* com respeito ao interrelacionamento e interfuncionamento entre as *AE* pares.

Como uma *AE* localiza a *AE* destino com a qual ela deseja formar uma associação de aplicação?.

Conforme especificado no documento *"Naming and Addressing"*<16> ,funções de diretório de aplicação processam endereçamento na Camada de Apresentação , *AE-Titles* e informações de endereçamento do protocolo de aplicação , de modo a prover um mapeamento entre essas categorias de informação.

As informações deste mapeamento podem ser guardadas

localmente e estarem disponíveis para acesso através de funções de diretório de aplicação ou podem ser guardadas remotamente.

É de responsabilidade local obter essas informações e torná-las disponíveis a uma função de diretório de aplicação.Se esta informação está armazenada remotamente ,um protocolo *OSI* (serviço de diretório) é usado para acessar essa informação.

Não é necessário que cada *AE* contenha o *ASE* que provê esse serviço de busca remota ; um sistema de gerenciamento local pode obter esse serviço de outra *AE* , ou até mesmo de uma *AE* pertencente a outro *AP*.Alguns nomes são definidos de maneira a habilitar a identificação de certos objetos na Camada de Aplicação (especificamente a dupla **AP-AE chamador** e **AP-AE chamado**), esses nomes podem ser vistos na tabela 2.2.

APPLICATION-PROCESS-TITLE
APLICATION-ENTITY-TITLE
APPLICATION-PROCESS-INVOCATION-IDENTIFIER
APPLICATION-ENTITY-INVOCATION-IDENTIFIER
APPLICATION-ASSOCIATION-IDENTIFIER
APPLICATION-PROCESS-TYPE-TITLE
APPLICATION-ENTITY-TYPE-TITLE
SYSTEM-TITLE

Tabela 2.2 - Nomes Associados á Identificação AP-AE na Camada de Aplicação.

Na lista de parâmetros do serviço *A-associate* (responsável pelo estabelecimento do contexto de aplicação) encontram-se alguns nomes mostrados na tabela 2.2 :**Calling AP-Title , Called AP-Title , Calling AE-Qualifier , Called AE-Qualifier , Calling AP-invocation identifier , Called AP-invocation identifier , Calling AE-invocation identifier, Called AE-invocation identifier** .Esses nomes são trocados como parâmetros, entre os ASEs envolvidos em um processamento de informação, na fase de estabelecimento de contexto, de modo a habilitar aos níveis inferiores a localização dos objetos (ASEs) na Camada de Aplicação, para que possa haver a troca de informações entre ambos.

2.1.8- O uso de uma AA

A capacidade para o estabelecimento e término de uma associação de aplicação , e consequentemente a obtenção do contexto da aplicação , está contida em um *ASE* específico de nome *ACSE* <3>.

Esse *ASE* deve estar presente em qualquer estabelecimento de uma associação de aplicação e será objeto de estudo detalhado nos próximos Capítulos.

2.1.9- O estabelecimento de uma AA

No estabelecimento de uma *AA* , uma invocação de *AE* especifica ao Serviço de Apresentação a localização do par *AE* , com o qual deseja processar informação , pelo seu endereço de apresentação (através dos *PSAPs - "Presentation Service Access Points"*).

Adicionalmente pode usar um ou mais dos seguintes identificadores :

a."AP-invocation-identifier"

b."AE-invocation-identifier"

Um *AE-invocation* pode também usar um identificador de uma associação de aplicação ("Aplication Association Identifier") de modo a identificar a *AA* e os *SAOs* , com os quais deseja processar informação , na entidade par.

Uma invocação de AE pode obter o *"P-Address"* da entidade AE-par usando as funções de diretório de aplicação de modo a prover o mapeamento entre *"AE-Title"* e o *"P-Address"*.

Este modo de endereçamento e estabelecimento de *AA* pode ser visto de uma maneira sistêmica na figura 2.9.

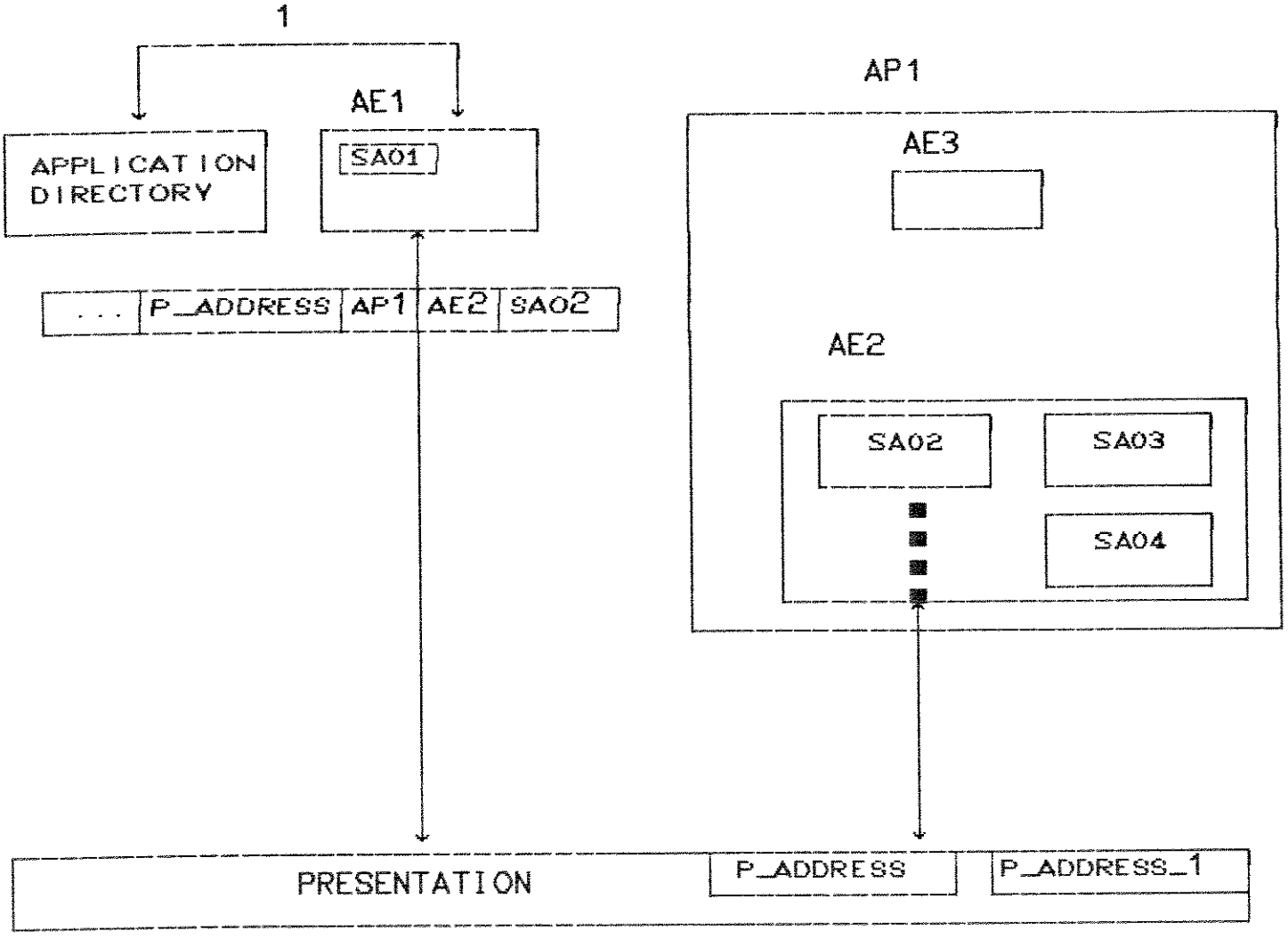


Figura 2.9- Estabelecimento de uma Associação de Aplicação.

Na figura 2.9 a entidade de aplicação AE1 utilizando-se das funções de diretório obtém o "P_Address" da entidade AE2.

Adicionalmente, usando os identificadores *"AP-invocation-idemtifier"* (AP1) e *"AE-invocation-identifier"* (AE2) localiza, univocamente no processo aplicativo 1, a entidade de aplicação AE2 , com a qual deseja processar

informação.Concluindo o estabelecimento da *AE-invocation*, utiliza o identificador SAO1 de modo a identificar a AA e o SAO (SAO2) com o qual processará informação.

Uma outra forma seria as "*AE-invocations*" relacionadas transferirem "*AP-title*" e "*AE-title*" durante o estabelecimento de uma associação de aplicação.Esta informação provê um meio , que é independente do "*Presentation-address*", para designar o *AE*.

Esses parâmetros trocados podem ser vistos na tabela 2.3 , que mostra os campos da "*APDU*" de requisição de associação usada pelo *ACSE* durante a fase de estabelecimento de uma associação de aplicação.

PROTOCOL VERSION
APPLICATION CONTEXT NAME
CALLING AP TITLE
CALLING AP INVOCATION IDENTIFIER
CALLING AE QUALIFIER
CALLING AE INVOCATION IDENTIFIER
CALLED AP TITLE
CALLED AP INVOCATION IDENTIFIER
CALLED AE QUALIFIER
CALLED AE INVOCATION IDENTIFIER
USER INFORMATION

Tabela2.3 - Parâmetros da "A-Associate request APDU".

2.1.10- O uso da Camada de Apresentação <17>

Uma *AE* é conectada a um ou mais *"PSAPs"* de modo a tornar-se endereçável no ambiente *OSI*.As invocações de AE que estão se comunicando usam o Serviço de Apresentação para transferir *"APDUs"* entre elas.O método de uso dos serviços são descritos pelas regras contidas no contexto de aplicação de uma associação de aplicação.

Duas invocações de *AE* podem estabelecer um ou mais contextos

de apresentação de modo a suportar a operação dos protocolos de aplicação entre seus *ASEs* em uma associação de aplicação.

A estrutura de *APDUs* de um *ASE* é especificada por um ou mais nomes de sintaxe abstrata.Para transferir essas *APDUs* entre as invocações de *AE*, usando o serviço de apresentação, é necessário o estabelecimento de um ou mais contextos de apresentação para cada sintaxe abstrata (um parâmetro relacionado é o "Presentation-Context-Definition-List" que é parte integrante da primitiva de serviço "*A-Associate*" do protocolo *ACSE*).

Durante a associação , ocorrências dessas *PDUs* são ligadas aos contextos de apresentação.

2.1.11 - Atividades do Elemento de Serviço de Aplicação (ASE)

Comunicações dentro de uma *AA* fazem uso de um ou mais *ASEs*, e cada uma de suas invocações de *AE* .A operação desses *ASEs* é coordenada por meio de regras contidas no contexto de aplicação em uma associação de aplicação.

A operação de um conjunto de *ASEs* em uma invocação de *AE*, para uma particular associação de aplicação, é modelada pelos *SAOs* , e existem dois aspectos a serem considerados:

1.Coordenação de *ASEs* dentro de um *SAO*;

2.Coordenação de *ASE* com sua entidade par relacionado cada qual com um *SAO*. Isto pode ser visto na figura 2.10.

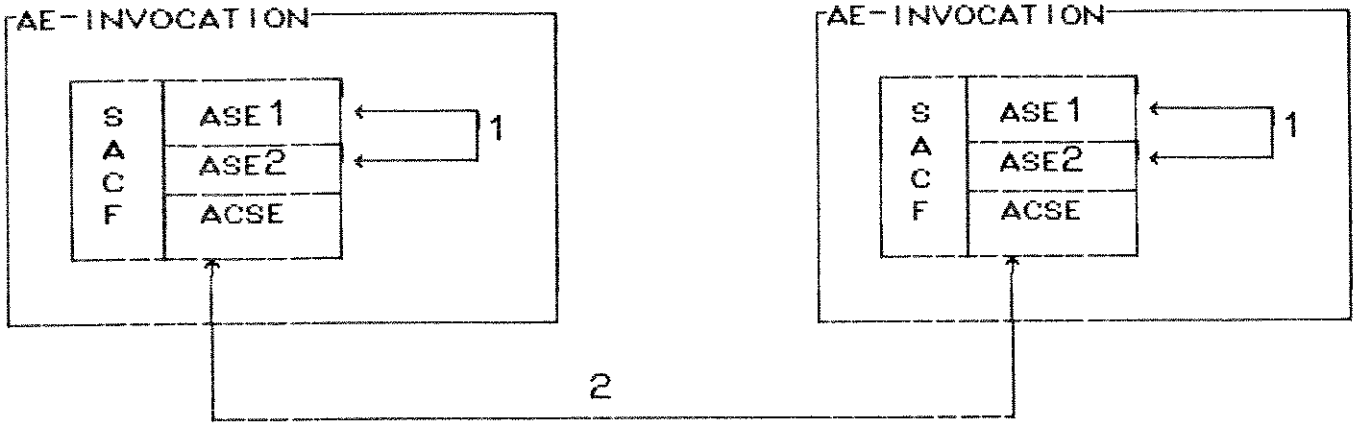


Figura 2.10 - Coordenação de Atividades de ASEs

A coordenação de *ASEs* dentro de um *SAO* é necessária para habilitá-los a executarem suas funções , tanto consecutivamente como concorrentemente, ou ambos.No caso de operação concorrente de *ASEs*, regras de concatenação e separação de *APDUs* dos diferentes *ASEs* são necessárias no contexto de aplicação de uma *AA*.

2.1.12 - Definição da Sintaxe Abstrata

Uma sintaxe abstrata é composta de regras usadas na especificação formal de dados , as quais são independentes de técnicas de codificação para representar os dados.

Para um dado *ASE* , a estrutura da *APDU* é especificada por um conjunto de uma ou mais sintaxes abstratas , a estrutura de

qualquer "*user information*" contida dentro desses *ASEs* também é especificada por outro conjunto de uma ou mais sintaxes abstratas.

A notação utilizada para a codificação dessa sintaxe abstrata é a *ASN.1* ("Abstract Syntax Notation One")<18>.

Um nome de sintaxe abstrata é associado com a definição de um conjunto de *APDUs* ou com a definição de um conjunto de "*user information*".Esse nome é usado para negociar entre chamado e chamador , durante a fase de estabelecimento de contexto , as representações contidas na *APDU* e na informação de usuário que serão trocadas, enquanto existir a associação de aplicação.

2.2 - ACSE - Descrição dos Serviços e Protocolos.

Conforme o documento *ISO/IEC DIS 9545 - "Information Processing Systems - Open Systems Interconnection - Application Layer Structure "* <12> que provê uma base para coordenar o desenvolvimento dos padrões para a Camada de Aplicação , de modo que estes estejam dentro da filosofia e modelo de referência *OSI* , o *ACSE* é um elemento de serviço de aplicação ("*Application Service Element*") que provê um meio consistente e único para o estabelecimento e término de todas as associações de aplicação (*AAs*).Todo e qualquer estabelecimento de uma *AA* exige pelo menos a presença de um *SASE* (*MMS* , *FTAM* , *MHS*, etc) e obrigatóriamente do *ACSE*.

2.2.1 - Os serviços ACSE.

Os serviços definidos na Especificação de Serviços *ACSE* são aqueles providos pelo Protocolo *ACSE* , e podem ser usados por outros elementos de Serviços do Nível de Aplicação.

A especificação de serviços faz uso das diversas definições do modelo *OSI* <2> , da estrutura do Nível de Aplicação <12> e da definição dos Serviços *ACSE* <3>.

2.2.1.1 - Serviços Definidos pelo ACSE.

Os serviços providos pelo *ACSE* para o estabelecimento e término de uma associação de aplicação podem ser visualizados na tabela 2.4.

SERVIÇOS	DESCRIÇÃO DOS SERVIÇOS
A_ASSOCIATE	ESTE SERVIÇO É USADO PARA O ESTABELECIMENTO DE UMA ASSOCIAÇÃO DE APLICAÇÃO , É ATRAVÉS DELE QUE É TROCADO O CONJUNTO DE REGRAS ENTRE DUAS ENTIDADES DE APLICAÇÃO (AE) DE MODO A GOVERNAR A COOPERAÇÃO ENTRE ELAS. NESTA FASE TE-SE O ESTABELECIMENTO DO CONTEXTO DE APLICAÇÃO.
A_RELEASE	ESTE SERVIÇO É USADO PELO REQUISITANTE EM QUALQUER DAS ENTIDADES DE APLICAÇÃO (AE) PARA CAUSAR O TÉRMINO DE UMA AA.SE A UNIDADE FUNCIONAL DA CAMADA DE SESSÃO RELEASE NEGOCIADO FOI SELECIONADA PARA A ASSOCIAÇÃO DE APLICAÇÃO , O RESPONDEDOR PODE RESPONDER NEGATIVAMENTE.ISTO FARÁ COM QUE O RELEASE SEJA MAL SUCEDIDO E A "AA"NÃO SERÁ DESFEITA.
A_ABORT	ESTE SERVIÇO É USADO PELO REQUISITANTE EM QUALQUER DAS AEs PARES DE MODO A CAUSAR UM TÉRMINO ABRUPTO DA ASSOCIAÇÀO DE APLICAÇÃO. NESTE CASO,POSSIVELMENTE,EXISTIRÁ PERDA DAS INFORMAÇÕES EM TRÂNSITO.
A_P_ABORT	ESTE SERVIÇO É USADO PELO PROVEDOR DO SERVIÇO ACSE PARA SINALIZAR UM RELEASE ANORMAL (ABORT) DA ASSOCIAÇÃO DE APLICAÇÃO DEVIDO A PROBLEMAS LOCALIZADOS NOS NÍVEIS INFERIORES À CAMADA DE APLICAÇÃO , SINALIZADO PELO "PRESENTATION".ESTA OCORRÊNCIA INDICA A POSSÍVEL PERDA DE INFORMAÇÃO EM TRÂNSITO. A_P_ABORT É UM SERVIÇO MAPEADO PELO PROVEDOR DO ACSE , DIRETAMENTE DA PRIMITIVA DE P_P_ABORT DA CAMADA DE APRESENTAÇÃO PARA OS NÍVEIS SUPERIORES.

Tabela 2.4 - Serviços Providos pelo ACSE

2.2.1.2 - A Estrutura dos Serviços.

O padrão *ACSE* associa , a cada serviço , uma tabela contendo colunas para qualificação de seus parâmetros nas primitivas "*Request*" , "*Indication*" , *Response*" e "*Confirm*". Cada tabela é definida com os seguintes elementos:

Lista de parâmetros de serviço:Define o conjunto de parâmetros e subparâmetros que estão presentes nas primitivas.A presença de cada parâmetro é descrita por um dos seguintes valores:

1."*Blank*" (branco) - não aplicável;

2."*C*" - condicional à presença anterior;

3."*M*" - mandatório;

4."*P*" - submetidos a condições definidas em <17>;

5."*U*" - opcional do usuário dos serviços do *ACSE*;

6."-" (=) - indica que o parâmetro é

semânticamente igual ao parâmetro na primitiva do serviço imediatamente a sua esquerda na tabela.

Na Tabela 2.5 mostra-se o exemplo da estrutura para o serviço *"A-Associate"*.

PARAMETER NAME	REQ	IND	RSP	CNF
Mode	U	M(=)		
*Application Context Name	M	M(=)	M	M(=)
*Calling AP Title	U	C(=)		
*Calling AE Qualifier	U	C(=)		
*Calling AP Invocation-identifier	U	C(=)		
*Calling AE Invocation-identifier	U	C(=)		
*Called AP Title	U	C(=)		
*Called AE Qualifier	U	C(=)		
*Called AP Invocation-identifier	U	C(=)		
*Called AE Invocation-identifier	U	C(=)		
*Responding AP Title			U	C(=)
*Responding AE Qualifier			U	C(=)
*Responding AP Invocation-identifier			U	C(=)
*Responding AE Invocation-identifier			U	C(=)
User Information	U	C(=)	U	C(=)
Result			M	M(=)
Result Source				M
*Diagnostic			U	C(=)
Calling Presentation Address	P	P		
Called Presentation Address	P	P		
Responding Presentation Address			P	P
*Presentation Context Definition List	P	P		
*Presentation Context Def. Result List		P	P	P
*Default Presentation Context Name	P	P		
*Default Presentation Context Result			P	P
Qualit of Service	P	P	P	P
*Presentation Requirements	P	P	P	P
Session Requirements	P	P	P	P
Initial Synchron. Point Serial Number	P	P	P	P
Initial Assignement of Tokens	P	P	P	P
Session Conection Identifier	P	P	P	P

* Não é usado no modo x.410 - 1984

Tabela 2.5 - Parâmetros do Serviço A-Associate

O parâmetro da tabela 2.5 "*Mode*" especifica o modo no qual os serviços do *ACSE* irão operar para esta associação.Pode-se ter um dos seguintes valores simbólicos :**normal ou x410-1984** <19>.

Cabe aqui abrir um parêntesis (Nota-se pelo documento <19> que o Protocolo x410-1984 [**ou RTS'84**] se comunica diretamente com a Camada de Sessão , isto poderia sugerir que este, quando foi definido pela ISO, não previa a necessidade de se utilizar um protocolo do tipo **ACSE** para o estabelecimento e término de uma **AA**.

Ao se observar as normas que definem o Protocolo de Aplicação MHS'84 vê-se que o mesmo utiliza os serviços do RTS'84 para o estabelecimento e término de uma associação com sua entidade par.O RTS'84 possui serviços que permitem a obtenção de uma pseudo Camada de Apresentação.Embora esses mecanismos não sejam tão claros como os do ACSE , os mesmos se responsabilizam pela AA.Vê-se, portanto, que a figura do ACSE não existia quando foi criado o protocolo RTS'84 e possivelmente, não fazia parte da concepção original da sequência de protocolos da ISO.Talvez pelo fato de que a Camada de Aplicação deveria se dedicar somente a aplicação é que tenha se criado o ACSE , para que este se preocupe com o controle das AAs e os outros protocolos de aplicação se preocupem com a aplicação em si.

Entretanto, tanto o ACSE como o RTS'84 deveriam interagir de alguma forma para que estivessem dentro da filosofia descrita na

Estrutura da Camada de Aplicação <12>.Isto pode ser notado ao verificarmos que a Máquina de Protocolos do ACSE (ACPM) prevê duas máquinas de estados diferenciadas pelo parâmetro "mode" , uma para operação em modo normal e outra exclusiva para o RTS'84.

Nota-se que a ACPM para o RTS'84 prevê trocas de PDU's entre os ACSEs pares , visto que isto é apenas uma adaptação ao RTS'84 e que o mesmo, na realidade, possui os serviços adequados ao controle de uma AA através de uma camada própria de Apresentação.

O objetivo do ACSE, então, é o de prover um meio único e confiável de controle de uma AA para que , prevalecendo a idéia original da ISO de modularidade e interdependência entre camadas , não seja mais necessário que os protocolos de Aplicação tenham mecanismos próprios de controle de uma AA e, com isso, facilite a interconexão dos sistemas que se quer sejam abertos) Fecha-se o parênteses!

Se o valor "mode" não está incluído (por ser opcional ao usuário) na primitiva de "*Request*" , o valor "*Default*" usado pelo provedor dos serviços do *ACSE* é o "normal".Pode-se notar que na primitiva "*Indication*" este parâmetro é do tipo *MC=>* , ou seja , é obrigatória sua presença na primitiva "*Indication*" e é semanticamente igual ao da primitiva "*Request*" , se presente.

O "*Application Context Name*" identifica o contexto da aplicação proposto pelo requisitante para a associação de aplicação ; o respondedor retornará o mesmo ou um nome diferente.O nome retornado especifica o contexto de aplicação a ser utilizado

para a associação que se deseja estabelecer.

Este nome irá identificar univocamente o contexto de aplicação que está sendo proposto pelo requisitante.

"Calling AP-Title" /"Called AP-Title" identificam o *AP* que contém o requisitante e o respondedor pretendido do serviço *"A-Associate"* , respectivamente.

"Calling AE-Qualifier" /Called AE-Qualifier identificam a *AE* particular do *AP* que requisitou o serviço *"A-Associate"* e o particular *AE* do *AP* que contém o respondedor pretendido do serviço *"A-Associate"*.

"Calling AP Invocation-identifier" /"Called AP Invocation-identifier" , estes parâmetros identificam a invocação do *AP* que contém o requisitante do serviço *"A-Associate"* e a invocação do *AP* que contém o respondedor pretendido deste serviço.

"Calling AE Invocation-identifier" /"Called AE Invocation-identifier" , estes parâmetros identificam o *AE* invocador que contém o requisitante do serviço *"A-Associate"*. e o o *AE* invocador que contém o respondedor pretendido deste serviço.

2.2.2 - O Protocolo ACSE.

O protocolo *ACSE* está diretamente relacionado ao estabelecimento e término de associações de aplicação *(AAs)* e tambem utiliza e referencia a Definição do Nível de Aplicação <12>

e a Definição do Nível de Apresentação <17>.A figura 2.11 mostra o interrelacionamento destes padrões.

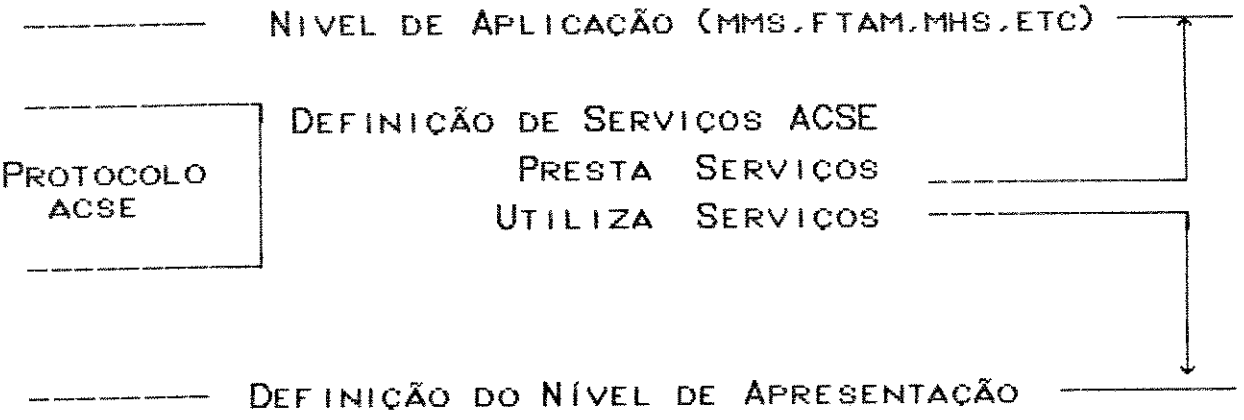


Figura 2.11 - Interrelacionamento entre o Protocolo ACSE e seus Níveis Adjacentes.

O protocolo *ACSE* é estruturado de forma a executar as funções do provedor *ACSE*. Estas funções estão relacionadas com a máquina de estados definida no Protocolo *ACSE* e são as seguintes:

a) Tratar as primitivas de serviço dos aplicativos da Camada de Aplicação local , mapeando-as em *APDUs* a serem enviadas ao usuário remoto via primitivas da Camada de Apresentação;

b) Tratar as *APDUs* corretas recebidas do usuário remoto via primitivas da Camada de Apresentação;

c) Tratar as *APDUs* inválidas recebidas.

A figura 2.12 mostra a tabela de estados da Máquina de Protocolo do ACSE (**ACPM**) para o modo normal ("normal mode").

	STA0 Idle_ Unass_ ociat.	STA1 Await. AARE	STA2 Await. A_ASC_ rsp	STA3 Await. RLRE	STA4 Await. A_RLS_ rsp	STA5 Assoc iated	STA6 Collis. associ- ation Initiat	STA7 Collis. associ- ation Respon.
A_ASCreq	p1 AARQ STA1							
A_ASCrsp+			AARE+ STA5					
A_ASCrsp-			AARE- STA0					
AARQ	p1 A_ASC_ ind STA2 ^p1 AARE- STA0							
AARE+		A_ASC_ cnf+ STA5						
ARE-		A_ASC_ cnf- STA0						
P_CONcnf-		A_ASC_ cnf- STA0						
A_RLSreq						RLRQ STA3		
A_RLSrsp+					RLRE+ STA0		RLRE+ STA3	
A_RLSrsp-					RLRE- STA5			

RLRQ				p2 A_RLS_ ind STA6 ^p2 A_RLS_ ind STA7		A_RLS_ ind STA4		
RLRE+				A_RLS_ cnf+ STA0				A_RLS_ cnf+ STA4
RLRE-				A_RLS_ cnf- STA5				
A_ABRreq		ABRT STA0	ABRT STA0	ABRT STA0	ABRT STA0	ABRT STA0	ABRT STA0	ABRT STA0
ABRT		A_ABR_ ind STA0	A_ABR_ ind STA0	A_ABR_ ind STA0	A_ABR_ ind STA0	A_ABR_ ind STA0	A_ABR_ ind STA0	A_ABR_ ind STA0
P_PABind		A_PAB_ ind STA0	A_PAB_ ind STA0	A_PAB_ ind STA0	A_PAB_ ind STA0	A_PAB_ ind STA0	A_PAB_ ind STA0	A_PAB_ ind STA0

Figura 2.12 - "ACPM state table for normal mode" <3>

Os procedimentos especificados no Protocolo ACSE são definidos em termos de :

- Interação entre AE-pares através da troca de PDU's ACSE;
- Interação entre um Provedor ACSE e o Provedor MMS no mesmo sistema , através da troca de primitivas ACSE;
- Interação entre o Provedor ACSE e a Camada de Apresentação através de primitivas do serviço de Apresentação;

A figura 2.13 mostra estas interações , definindo as interfaces dos elementos do Protocolo ACSE na Camada de Apresentação.

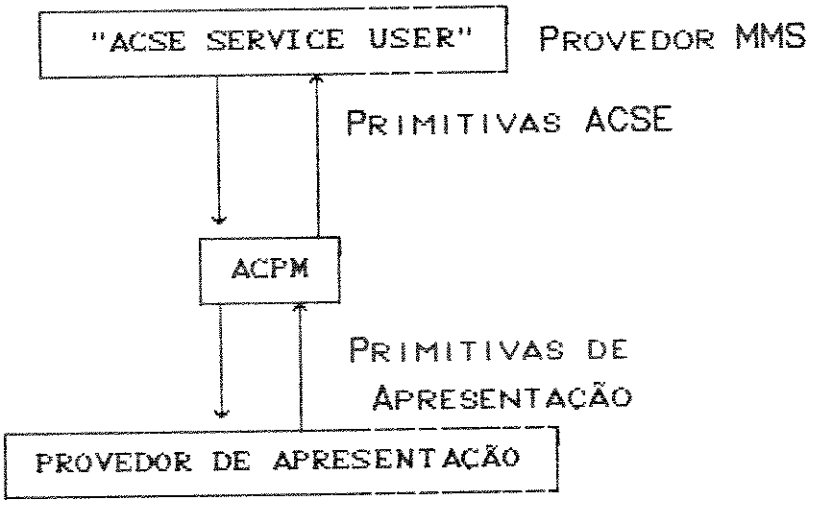


Figura 2.13 - Interações entre os elementos no Protocolo ACSE

2.2.2.1 - O Modelo do Protocolo ACSE.

A máquina de protocolo para o controle da associação *ACPM* (*"Association Control Protocol Machine"*) é modelada como uma máquina de estados finitos.

A ACPM comunica-se com o usuário dos seus serviços por meio de serviços definidos em <3>.

A ACPM comunica-se com a Camada de Apresentação por meio dos serviços definidos em <17>.Vê-se adiante que as primitivas do *ACSE* são diretamente mapeadas para as primitivas da Camada de Apresentação.

A ACPM é estimulada pelo recebimento de eventos de entrada do usuário dos serviços do *ACSE* (primitiva *request* ou *response*) e do provedor dos serviços do Nível de Apresentação (primitiva *indication* ou *confirm*) através da conexão com o Nível de Apresentação que está suportando a associação de aplicação com sua entidade par.

A *ACPM* responde a estes eventos de entrada emitindo eventos de saída ao *"ACSE-service-user"* e ao Nível de Apresentação.Os eventos de saída para o Nível de Apresentação são *"Presentation request ou response primitives"* e ao *"ACSE-service-user"* são *"ACSE indication ou confirm primitives"*.

A recepção de eventos de entrada , a geração de ações dependentes , e o evento de saída resultante são considerados ações concatenadas, isto é , ocorrem simultaneamente.

2.2.2.2 - O Campo de Aplicação do Protocolo ACSE.

O Protocolo *ACSE* especifica:

a) Procedimentos únicos para o estabelecimento , término normal ("*normal release*") ou término abrupto (*abnormal release*") de uma associação de aplicação;

b) A estrutura de *APDUs-ACSE* , utilizada para a troca de informações entre as entidades pares do *ACSE*.

Os procedimentos são definidos em termos de:

a) Interação entre entidades de aplicação pares através da troca de *APDUs-ACSE*;

b) Interação entre um provedor *ACSE* e o usuário *ACSE* no mesmo sistema , através da troca de primitivas *ACSE*;

c) Interação entre um provedor *ACSE* e o Nível de Apresentação através da troca de primitivas do Serviço de Apresentação.

Os procedimentos descritos acima estão todos apresentados a seguir nas tabelas 2.6 e 2.7. que com a figura 2.12, completam a especificação da ACPM.

NOME ABREVIADO	DESTINO	NOME E DESCRIÇÃO
A_ASCIND	AC_USER	A_ASSOCIATE INDICATION PRIMITIVE
A_ASCCNF+	AC_USER	A_ASSOCIATE CONFIRM PRIMITIVE (RESULT = "ACCEPTED")
A_ASCCNF-	AC_USER	A_ASSOCIATE CONFIRM PRIMITIVE (RESULT = "REJECTED [PERMANENT] " OU "REJECTED [TRANSIENT] "
AARQ	AC_PEER	A_ASSOCIATE_REQUEST APDU O AARQ É ENVIADO COMO "USER DATA" NA "P_CONNECT REQUEST PRIMITIVE"
AARE+	AC_PEER	A_ASSOCIATE_RESPONSE APDU (RESULT = ACCEPTED) O AARE+ É ENVIADO COMO "USER DATA" NA "P_CONNECT RESPONSE PRIMITIVE" (RESULT = ACCEPTANCE)
AARE-	AC_PEER	A_ASSOCIATE_RESPONSE APDU (RESULT = REJECTED [PERMANENTE] OU REJECTED [TRANSIENT]) O AARE- É ENVIADO COMO "USER DATA" NA "P_CONNECT RESPONSE PRIMITIVE" (RESULT = USER-REJECTION)
A_RLSIND	AC_USER	A_RELEASE INDICATION PRIMITIVE

A_RLSCNF+	AC_USER	A_RELEASE CONFIRM PRIMITIVE RESULT = AFFIRMATIVE
A_RLSCNF-	AC_USER	A_RELEASE CONFIRM PRIMITIVE RESULT = NEGATIVE
RLRQ	AC_PEER	A_RELEASE_REQUEST APDU A PDU RLRQ É ENVIADA COMO "USER DATA " NA P_RELEASE_REQUEST_PRIMITIVE
RLRE+	AC_PEER	A_RELEASE_RESPONSE_APDU A PDU RLRE+ É ENVIADA COMO "USER DATA" NA P_RELEASE_RESPONSE_PRIMITIVE (RESULT = "AFFIRMATIVE")
RLRE-	AC_PEER	A_RELEASE_RESPONSE_APDU A PDU RLRE- É ENVIADA COMO "USER DATA" NA P_RELEASE_RESPONSE_PRIMITIVE (RESULT = "NEGATIVE")
A_ABR_IND	AC_USER	A_ABORT INDICATION PRIMITIVE (SOURCE = "ACSE SERVICE -USER" OU ACSE SERVICE-PROVIDER")
ABRT	AC_PEER	A_ABORT APDU (SOURCE = "ACSE SERVICE-USER OU ACSE SERVICE-PROVIDER") O ABRT É ENVIADO COMO USER DATA NA P_U_ABORT REQUEST PRIMITIVE
A_PAB_IND	AC_USER	A_P_ABORT INDICATION PRIMITIVE

Tabela 2.6 - Eventos de entrada da ACPM

NOME ABREVIADO	FONTE	NOME E DESCRIÇÃO
A_ASCREQ	AC_USER	A_ASSOCIATE REQUEST PRIMITIVE
A_ASCRSP+	AC_USER	A_ASSOCIATE RESPONSE PRIMITIVE (RESULT = "ACCEPTED")
A_ASCRSP-	AC_USER	A_ASSOCIATE RESPONSE PRIMITIVE (RESULT = "REJECTED [PERMANENT] " OU "REJECTED [TRANSIENT] "
AARQ	AC_PEER	A_ASSOCIATE_REQUEST APDU O AARQ É O CONTEÚDO DO "USER DATA" NA "P_CONNECT INDICATION PRIMITIVE"
AARE+	AC_PEER	A_ASSOCIATE_RESPONSE APDU (RESULT = ACCEPTED) O AARE É O CONTEÚDO DO "USER DATA" NA "P_CONNECT CONFIRM PRIMITIVE" (RESULT = ACCEPTANCE)
AARE-	AC_PEER	A_ASSOCIATE_RESPONSE APDU (RESULT = REJECTED [PERMANENTE] OU REJECTED [TRANSIENT]) O AARE É O CONTEÚDO DO "USER DATA" NA "P_CONNECT CONFIRM PRIMITIVE" (RESULT = USER-REJECTION)
A_RLSREQ	AC_USER	A_RELEASE REQUEST PRIMITIVE

A_RLSRSP+	AC_USER	A_RELEASE RESPONSE PRIMITIVE RESULT= AFFIRMATIVE
A_RLSRSP-	AC_USER	A_RELEASE RESPONSE PRIMITIVE RESULT= NEGATIVE
RLRQ	AC_PEER	A_RELEASE_REQUEST APDU A RLRQ É O CONTEÚDO DO "USER DATA" NA P_RELEASE_INDICATION_PRIMITIVE
RLRE+	AC_PEER	A_RELESE_RESPONSE_APDU A RLRE+ É O CONTEÚDO DO "USER DATA"NA P_RELEASE_CONFIRM_PRIMITIVE VE (RESUL = "AFFIRMATIVE")
RLRE-	AC_PEER	A_RELESE_RESPONSE_APDU A RLRE- É O CONTEÚDO DO "USER DATA"NA P_RELEASE_CONFIRM_PRIMITIVE (RESUL = "NEGATIVE")
A_ABR_REQ	AC_USER	A_ABORT REQUEST PRIMITIVE
ABRT	AC_PEER	A_ABORT APDU O ABRT É O CONTEÚDO DO USER DATA NA P_U_ABORT INDICATION PRIMITIVE
P_PAB_IND	PS_PROVIDER	P_P_ABRT INDICATION PRIMITIVE

Tabela 2.7 - Eventos de saída na ACPM

2.2.2.3 - Os Elementos de Procedimento do Protocolo

Os elementos de procedimento do protocolo são descritos em relação ao envio e recebimento de *APDUs ACSE* e suas relações com os eventos (entrada ou saída) e primitivas de serviço ("*request* , *indication* , *response* , *confirm*") na interface entre o usuário e o provedor *ACSE*.

O Protocolo *ACSE* define cinco (5) tipos de *APDUs* como pode ser visto na tabela 2.8.

PDU ACSE	PRIMITIVA MAPEADA
AARQ_APDU	REQUEST DO SERVIÇO A_ASSOCIATE
AARE_APDU	RESPONSE DO SERVIÇO A_ASSOCIATE
RLRQ_APDU	REQUEST DO SERVIÇO RELEASE
RLRE_APDU	RESPONSE DO SERVIÇO RELEASE
ABRT_APDU	REQUEST DO SERVIÇO ABORT , ESTE SERVIÇO NÃO É CONFIRMADO POR NENHUMA PDU ESPECIAL

Tabela 2.8 - PDU's do ACSE.

As *APDUs* do *ACSE* são repassadas no campo "*user-data*" das primitivas do Nível de Apresentação.

A primitiva do serviço "*A-P-Abort indication*" possui somente o parâmetro "*Provider Reason*" o qual é mapeado diretamente do parâmetro correspondente da primitiva "*P-P-Abort indication*".

2.2.3 - Relacionamento do ACSE com as Camadas de Apresentação e Sessão.

O documento x217 - *Association Control Service Definition for Open System Interconnection* especifica que quando a Camada de Apresentação <17> é suportada pela versão 1 do Protocolo de Sessão <20>, ela é submetida a restrições de tamanho no parâmetro "*user-data*" e, consequentemente, isto reflete-se em uma otimização de codificação para o serviço "*A-Abort*" do *ACSE*. Neste caso a máquina de protocolos do *ACSE* (**ACPM-Association ControlProtocol Machine**) não envia nenhum *APCI (Association Protocol Control Information)*, relacionado ao "*A-Abort*", a entidade par.

Ela simplesmente emite um "*P-U-Abort* request primitive" e, se o parâmetro "*user information*" está incluido na primitiva "*A-Abort request*", este parâmetro é passado como "*user-data*" na primitiva "*P-U-Abort*".

Na implementação do *ACSE* que será vista no capítulo 3, optou-se por uma implementação prevendo o relacionamento com a Camada de Sessão em versões diferentes da 1 (um), ou seja, não há restrições de tamanho do parâmetro "*user-data*" no Protocolo de Apresentação e, portanto, a *APDU* "*A-Abort*" será trocada entre os *ACSEs* localizados nas entidades pares.

3 - Implementação da Máquina de Protocolos do ACSE

Este capítulo define uma máquina de protocolos para o Controle de Associação de Aplicação ("*ACPM - Association Control Protocol Machine*") para o modo normal de operação em termos da figura 2.12 apresentada no Capítulo 2.

Esta figura mostra o interrelacionamento entre os estados possíveis da *ACPM* , os elementos de entrada que ocorrem no protocolo , as ações tomadas , e finalmente , o resultado da *ACPM*.

Esta figura não constitue , por si só , uma definição formal da *ACPM*. Está incluída apenas para prover uma especificação mais precisa e uma melhor visualização dos elementos de procedimento do protocolo definidos no § 7 de <3>.Esta Figura faz uso das seguintes abreviações:

ACSE service-user" ("*AC-user*") - usuário do serviço *ACSE*;

"*peer ACPM*" ("*AC-peer*") - entidade ACPM par ;

"*presentation service provider*" ("*PS-provider*") - provedor da Camada de Apresentação.

Capítulo 3 - IMPLEMENTAÇÃO DA MÁQUINA DE PROTOCOLOS DO ACSE

3.1 - Convenções.

A interseção de uma linha ("*incoming event*") com uma coluna ("*state*") forma uma célula.

Na figura 2.12, uma célula em branco representa a combinação de uma linha com uma coluna (representando um estado) que não é definida.Para as células preenchidas , a *ACPM* prevê uma ou mais ações a serem tomadas.

Estas ações podem ser imediatas ou condicionais.Ações imediatas implicam em :

a) Um evento de saída ;e

b) Um estado resultante.

Ações condicionais implicam em :

Ações imediatas seguidas da condição "verdadeira ou falsa" (^ representa um Booleano "Not") ;

No caso das células em branco que implicam interseções inválidas para a ACPM , as ações a serem tomadas são :

1 - Se o evento de entrada está relacionado com o recebimento de uma *APDU* ou é um evento oriundo da Camada de Apresentação ("*PS-provider*") , a ACPM emite a ambos , isto é , ao usuário do serviço *ACSE* e ao provedor da Camada de Apresentação , a primitiva "*A-ABR ind*" e a *APDU ABRT* do *ACSE* como eventos de saída, respectivamente.

2 - Se o evento de entrada vem do usuário do serviço

ACSE ("ACSE service-user") , qualquer ação tomada pela ACPM é de caráter local.

Neste ponto o documento <3> deixa muito vago o que se imagina por uma ação de caráter local.Nesta implementação foi feita uma análise caso a caso de cada interseção inválida e o resultado , será visto mais adiante na especificação de implementação do *ACSE*.

3.2 - Programação da ACPM

Este ítem tem direto relacionamento com a figura 2.12 e as tabelas 2.6 e 2.7 .Esta programação foi feita baseada no Sistema Didático **"SISDI-MAP"** que foi discutido no capítulo primeiro desta tese.

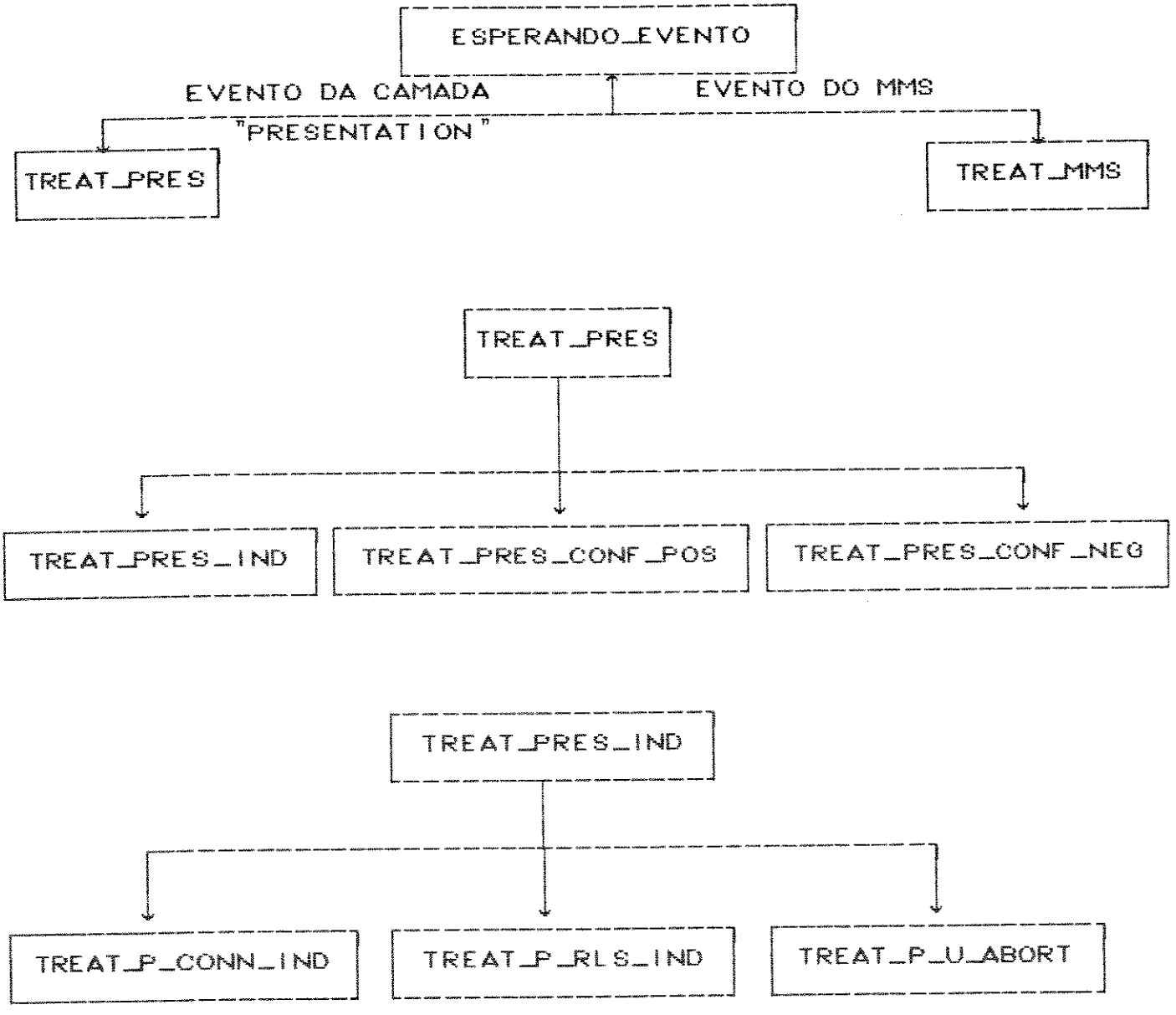
O *"SISDI-MAP"* tem seu modelo conceitual apresentado na figura 1.1.

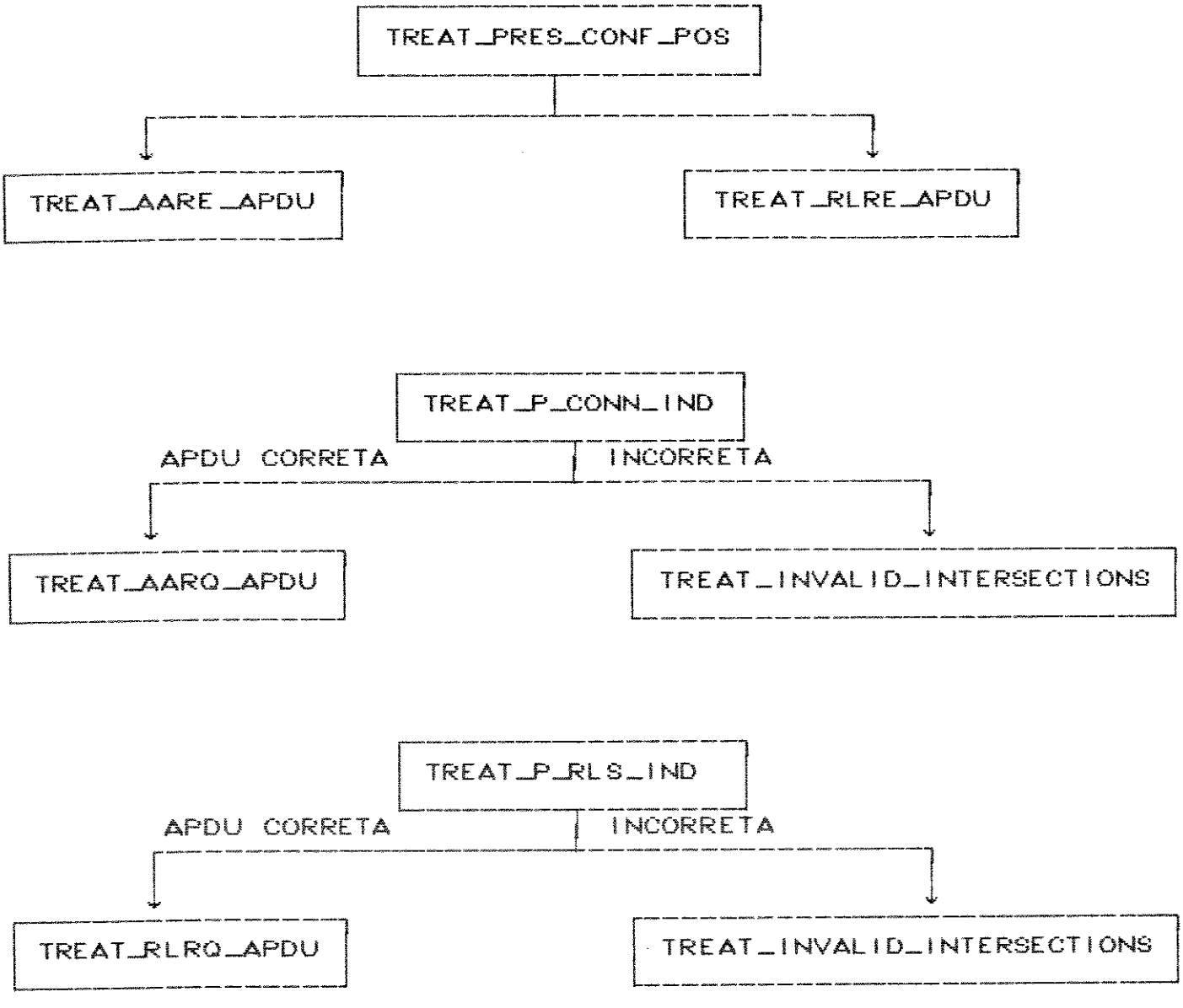
Pode-se notar pela figura 1.1 que o processo que faz o papel de usuário do *ACSE* é o *MMS ("Manufacturing Message System")* <1> e o papel do provedor da Camada de Apresentação é o simulador Nível 6 , doravante denominados *MMS* e *PRES*.

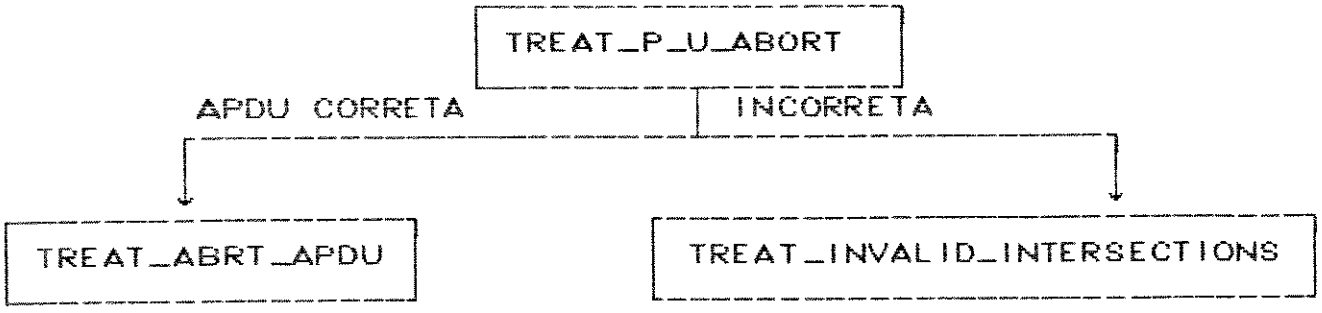
3.2.1 - Especificações de implementação.

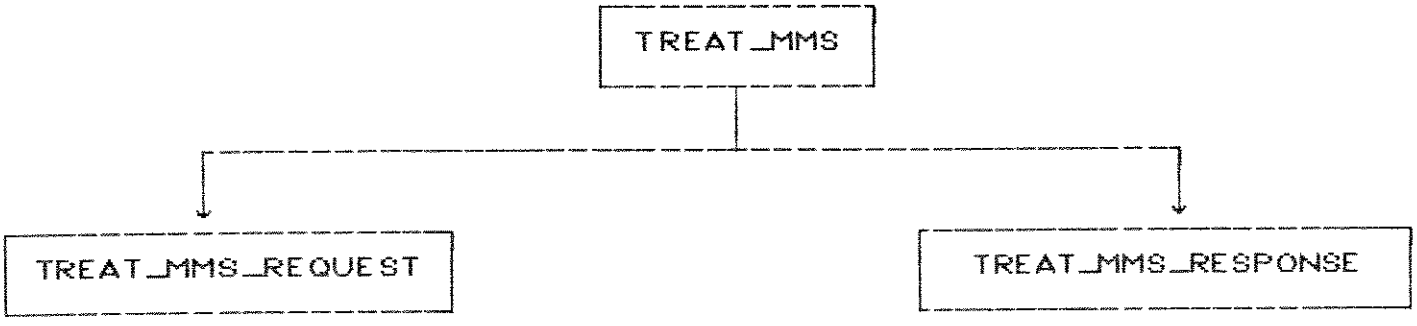
A implementação do PROC_ACSE é baseada na ACPM apresentada

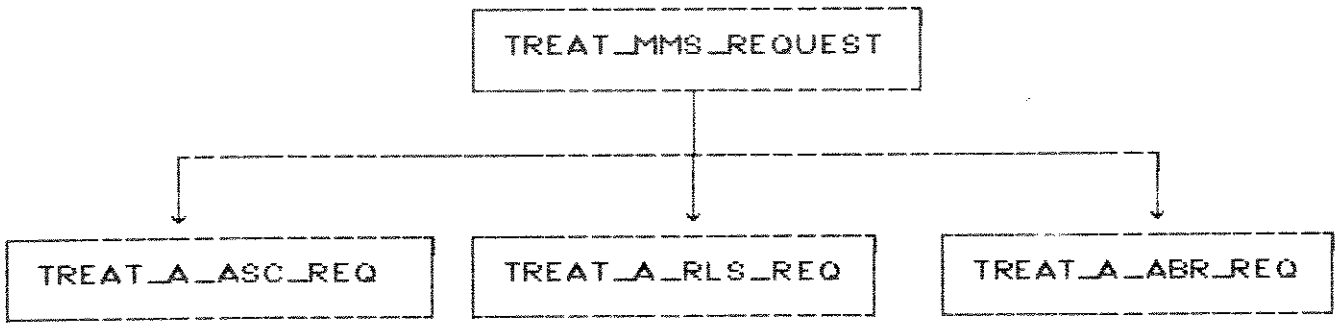
nos documentos do ACSE e tem como estrutura apresentada na figura 3.1.




TREAT_PRES_CONF_POS
TREAT_AARE_APDU
TREAT_RLRE_APDU
TREAT_P_CONN_IND
APDU CORRETA
INCORRETA
TREAT_AARQ_APDU
TREAT_INVALID_INTERSECTIONS
TREAT_P_RLS_IND
APDU CORRETA
INCORRETA
TREAT_RLRQ_APDU
TREAT_INVALID_INTERSECTIONS


TREAT_P_U_ABORT
APDU CORRETA
INCORRETA
TREAT_ABRT_APDU
TREAT_INVALID_INTERSECTIONS


TREAT_MMS
TREAT_MMS_REQUEST
TREAT_MMS_RESPONSE


TREAT_MMS_REQUEST
TREAT_A_ASC_REQ
TREAT_A_RLS_REQ
TREAT_A_ABR_REQ

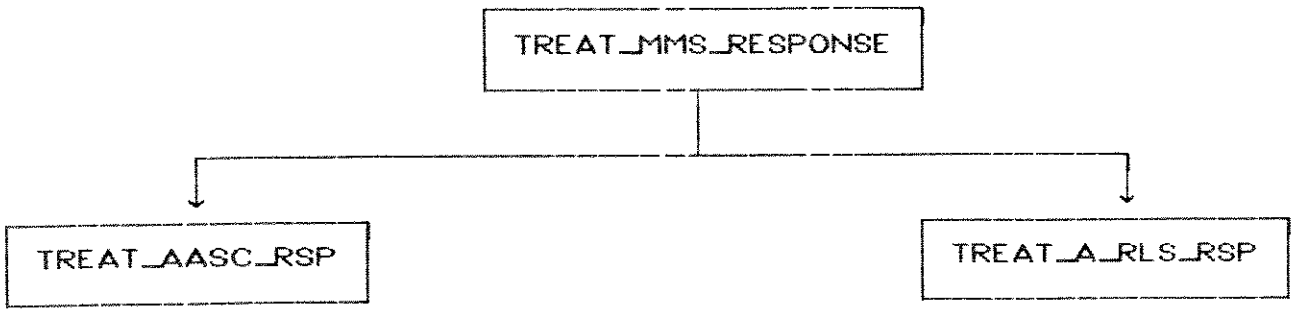


Figura 3.1 - Estrutura do tratamento de mensagens recebidas e/ou enviadas pelo MMS e Camada de Apresentação.

```
Programa Principal :
     BEGIN
          waiting_event;
          CASE origem do evento OF
            pres : treat_pres
            mms  : treat_mms
          END
     END;
```

Neste caso, o programa principal espera por eventos de entrada, que como visto anteriormente podem se providos pelo usuário do serviço ACSE (aqui representado pelo MMS) ou pelo provedor da Camada de Apresentação (aqui representado pelo PRES).

3.2.1.1 - Estrutura do procedimento de tratamento das primitivas de serviço da Camada de Apresentação.

Os procedimentos de tratamento das primitivas de serviço da Camada de Apresentação , estão representando as linhas da figura 2.12 ("*incomming events*").

```
PROCEDURE treat_pres;
    BEGIN
        CASE tipo da primitiva OF
            indication  : treat_pres_indication
            confirm_pos : treat_pres_conf_pos
            confirm_neg : treat_pres_conf_neg
        END
    END;
```

```
PROCEDURE treat_pres_indication;
     BEGIN
          CASE tipo do serviço OF
               p_connection : treat_p_conn_ind
               p_release    : treat_p_ris_ind
               p_u_abort    : treat_p_u_abort

          END
     END;

PROCEDURE treat_p_conn_ind;
     BEGIN
          IF APDU correta
               THEN treat_aarq_apdu
               ELSE treat_invalid_intersections
          END
     END;
```

O texto do documento <3> deixa bem claro em seu § 7.4.3 ; o recebimento de uma *APDU* ou campo dentro da *APDU* que não está definido dentro do padrão *ASN.1* descrito na negociação, deve ser tratado como um erro de protocolo.

O texto também diz em seu § 7.3.3.4 *"Protocol errors"* :

- Dois (2) tipos de erros de protocolo no *ACSE* são possíveis :

1 - para um particular estado da ACPM , uma *APDU* inesperada é recebida ; ou

2 - um campo inválido é encontrado durante o processamento de uma *APDU* de entrada.

- Se uma *APDU* inesperada é recebida , o procedimento de "release" abrupto (através da *APDU ABRT*) é invocado.Se um campo inválido é encontrado , o procedimento de tratamento desta *APDU* é interrompido e o procedimento de "release" abrupto é invocado.

- Como parte do procedimento de "release" abrupto , a ACPM emite uma primitiva do tipo *"A-Abort indication"* aos usuários dos serviços *ACSE* e um *"ABRT APDU"* ao seu ACPM par , a menos que o erro ocorra durante o procedimento de estabelecimento da associação, como resultado da recepção de uma *"AARQ APDU "* inválida.O texto <3> § 7.3.3.4 segue adiante e diz que no caso de uma implementação utilizando uma versão para a Camada de Sessão , que não a um (1) , devem ser tomadas providências que ver-se-à no procedimento "treat_invalid_intersection".

Como pode-se notar na figura 1.1, existe um processo chamado Processo de Interface de Operação de Usuário.Esta interface foi projetada com o intuito de ser ao mesmo tempo modular , portátil e de fácil manutenção.

Existe ainda outro módulo , encarregado do tratamento de sinalização com os módulos de protocolo.Com a utilizaçào desses

módulos o usuário pode configurar a interface de acordo com suas necessidades , indicando o tipo de monitoração de mensagens que deseja e o tipo de acesso ao sistema de protocolo , como por exemplo , para introduzir erros nas mensagens dentro do sistema.Um desses erros diz respeito à *APDU* recebida pelo *ACSE*.

Como na implementação do "*SISDI_MAP*" não existia um processo que codificava e decodificava as *APDUs*, segundo a *ASN.1*, imaginou-se um meio prático de testar erros provenientes desta codificação nas *APDUs* trocadas entre as entidades pares, de modo que o protocolo pudesse ser totalmente exercitado e testado.

É através do processo que realiza a Interface de Operação do Usuário que simula-se um erro de codificação onde o usuário preenche um campo específico com o código erro.Este campo é o que está sendo exemplificado no teste "*IF APDU* correta" .

```
PROCEDURE treat_p_ris_ind;
     BEGIN
          IF APDU correta
               THEN treat_rirq_apdu
               ELSE treat_invalid_intersections
          END
     END;

PROCEDURE treat_p_u_abort;
```

```
BEGIN
      IF APDU correta
            THEN treat_abrt_apdu
            ELSE envia_a_abr_ind
                  muda estado para STAO
      END
END;
```

```
PROCEDURE treat_p_p_abr_ind;
          BEGIN
               CASE estado ACPM OF
*OBS                STAD   : call_error
                    OTHERS : envia a_p_abr_ind
                             muda estado máquina ACPM para STAD
               END
          END;
```

*OBS : O item § 3.1.2 do documento <3> diz :

Se o evento de entrada está relacionado com o recebimento de uma *APDU* ou é um evento oriundo da Camada de Apresentação , a ACPM emite a ambos , isto é , ao usuário do serviço *ACSE* e ao provedor da Camada de Apresentação (que é este caso analisado) *A_Abort_indication* e *ABRT_APDU*, respectivamente.

Entretanto o documento, apesar de generalizar para qualquer caso, falha quando do tratamento deste em especial.Neste caso, não cabe a interpretação acima, pois a ACPM está no estado STAD ,ou seja , ainda não existe conexão estabelecida e foram detectados problemas nos níveis inferiores, representados pelo recebimento da primitiva "*p_p_abr_ind*".

Isto deve ser indicado ao operador do sistema como alarme para que ele tome as providências cabíveis , por exemplo , reinicializar todo o sistema.

```
PROCEDURE treat_aarq_apdu;
        BEGIN
            IF estado ACPM STA0
                THEN
                    IF p1 (ACPM pode suportar a requisição de
conexão??)
                        THEN envia_a_assoc_ind
                            muda estado para STA2
                        ELSE envia_aare_neg_apdu
                    END
*               ELSE treat_invalid_intersections
            END
        END;
```

*OBS : Como dito anteriormente no caso da recepção de uma *AARQ APDU* inválida, a análise do documento <3> levou à conclusão de que o recebimento desta *APDU* inválida deveria ser tratada como um erro de protocolo e, portanto, seguir os procedimentos normais adotados.

Entretanto, com a utilização da *"API" (Application Protocol Interface")* <7> o recebimento desta *APDU* inválida não ocorre, visto que a mesma funciona como um filtro entre o processo

aplicativo e o *ACSE* e, portanto, este tipo de erro é filtrado

Na fase de implementação do ACPM na linguagem "*C*", algumas alterações foram feitas, pois não mais se fez necessário o teste do estado atual da ACPM.

```
PROCEDURE treat_rlrq_apdu;
          BEGIN
               CASE estado ACPM OF
                    STA3 : IF p2 (ACPM originou esta associação)
                              THEN envia_a_rls_ind
                                   muda estado para STA6
                              ELSE envia_a_rls_ind
                                   muda estado para STA7
                              END
                    STA5 : envia_a_rls_ind
                           muda estado para STA4
                    OTHERS : treat_invalid_intersections
               END
          END;
PROCEDURE treat_abrt_apdu;
     BEGIN
          CASE estado ACPM OF
*OBS           STA0 : call_error
               OTHERS : envia_a_abr_ind
```

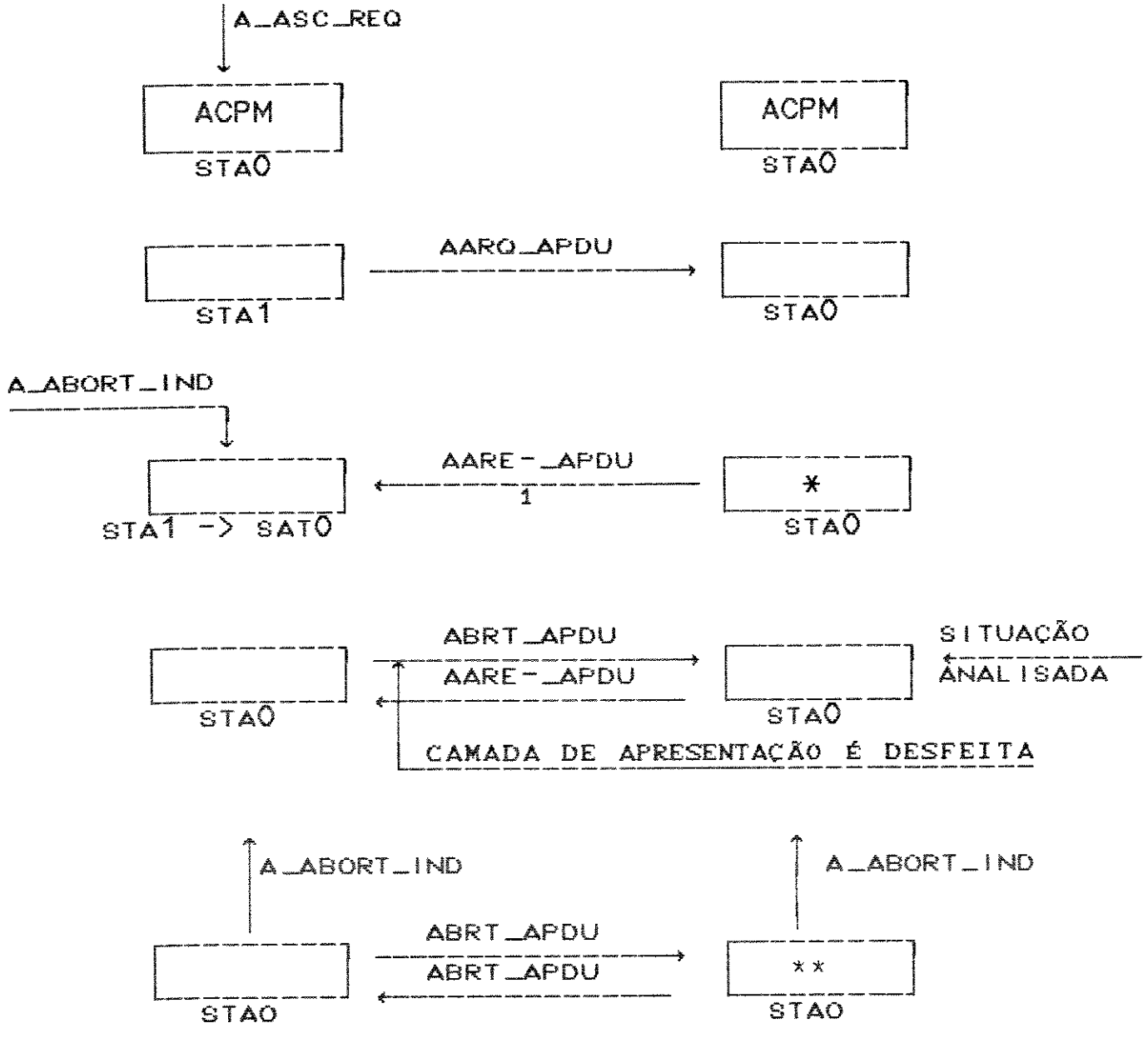
```
                    muda estado para STA0

        END

    END;
```

*OBS : Neste caso, a análise do documento <3> levaria a crer que, estando recebendo uma *APDU* em um estado inválido, dever-se-ia emitir uma primitiva "*A_Abort_indication*" ao usuário do serviço *ACSE* e uma *ABRT_APDU* ao *ACSE* par.

Entretanto, analisando o exemplo da figura 3.2 vê-se que o mesmo não procede, uma vez que, quando da emissão da primeira "*ABRT_APDU*", a Camada de Apresentação já está desfeita.Neste caso , o estado inválido apenas será tratado de uma maneira local , com isso indicar-se-á ao operador do sistema um erro para que ele possa tomar as providências cabíveis.



* - Hipótese : esta ACPM não suporta a conexão , isto é , as versões do protocolo não coincidem.

** - Neste caso não é possivel que a ACPM esteja enviando ABRT_APDU e A_ABORT_IND pois não mais existe a Camada de Apresentação , que foi desfeita anteriormente.

1 - Recebimento de uma APDU em estado inválido.

Figura 3.2 - Exemplo de tratamento do recebimento de uma ABRT_APDU inválida

```
PROCEDURE treat_aare_apdu;
     BEGIN
          IF APDU correta
               THEN
                    IF aare_apdu_pos
                         THEN
                              CASE estado ACPM OF
                                   STA1 : envia_a_asc_cnf(+)
                                          muda estado para STA5
                                   OTHERS : treat_invalid_intersections
                              END

                         ELSE
                              CASE estado ACPM OF
                              STA1 : envia_a_asc_cnf(-)
                                     muda estado para STA0
                              OTHERS : treat_invalid_intersections
                              END
                    END
               ELSE treat_invalid_intersections
          END
     END;
```

```
PROCEDURE treat_rlre_apdu;
    BEGIN
        IF APDU correta
            THEN
                IF rlre_apdu_pos
                    THEN
                        CASE estado ACPM OF
                            STA3 : envia_a_rls_cnf(+)
                                   muda estado para STA0

                            STA7 : envia_a_rls_cnf(+)
                                   muda estado para STA4
                            OTHERS : treat_invalid_intersections
                        END

                    ELSE
                        CASE estado ACPM OF
                            STA3 : envia_a_rls_cnf(-)
                                   muda estado para STA5
                            OTHERS : treat_invalid_intersections
                        END
                END
```

```
            ELSE treat_invalid_intersections
        END
    END;
```

3.2.1.2 -Estrutura do procedimento de tratamento das primitivas de serviço da Camada de Aplicação.

Os Procedimentos de tratamento das primitivas de serviço da Camada de Aplicação (neste caso o MMS) , estão representando as linhas da tabela da figura 2.12 ("*incoming events*").

```
PROCEDURE treat_mms;
    BEGIN
        CASE tipo da primitiva OF
            request      : treat_mms_request
            response_pos : treat_mms_response
            response neg : treat_mms_response
        END
    END;

PROCEDURE treat_mms_request
    BEGIN
        CASE tipo da primitiva OF
            a_associate : treat_a_asc_req
```

```
                a_release    : treat_a_ris_req
                a_abort      : treat_a_abr_req
            END
        END;

PROCEDURE treat_a_asc_req;
        BEGIN
*OBS        IF p1 (ACPM pode suportar a requisição de conexão?)
                THEN envia_aarq_apdu
                     muda para estado STA1
                ELSE  call_error
            END
        END;
```

*OBS : O documento de Especificação do Protocolo ACSE faz alusão ao teste "*p1*" apenas uma única vez em todo o documento , não deixando uma idéia clara do que seria este teste.

A interpretação do texto permitea concluir que o mesmo estaria relacionado à capacidade do *ACSE* de suportar um número genérico "n" de conexões devido à implementação , ou seja , teria que se controlar o número de conexões abertas até que o valor máximo permitido fosse encontrado e, a partir daí, o ACPM não mais suportaria a requisição de conexão.

Entretanto, ainda restaria uma dúvida no caso do tratamento da recepção da primitiva "*a_asc_req*", quando recebida no estado STA0 ("idle"), pois não há uma ação a ser tomada se "p1" não for verdadeira (^p1) , como no do recebimento de uma "*AARQ_APDU*".

Como na implementação do *ACSE* existe a presença da *API* que efetivamente controlará o número máximo de conexões que poderão ser suportadas , optou-se por desconsiderar a existência do teste "p1" no primeiro caso.

```
PROCEDURE treat_a_ris_req;
     BEGIN
          IF estado da ACPM STA5
               THEN envia_rirq_apdu
                    muda estado para STA3
*OBS           ELSE call_error
          END
     END;

PROCEDURE treat_a_abr_req;
     BEGIN
          IF estado da ACPM STA0
*OBS           THEN call_error
               ELSE envia_abrt_apdu
                    muda estado para STA0
```

```
            END
      END;

PROCEDURE treat_mms_response;
      BEGIN
            CASE tipo da primitiva OF
                  a_associate : treat_a_asc_rsp
                  a_release   : treat_a_ris_rsp
            END
      END;

PROCEDURE treat_a_asc_rsp;
      BEGIN
            CASE estado ACPM OF
                  STA2 :    IF a_asc_rsp(+)
                                    THEN envia_aare_apdu(+)
                                          muda para estado STA5
                                    ELSE envia_aare_apdu(-)
                                          muda estado para STA0
                              END
*OBS              OTHERS call_errors
            END
      END;
```

```
PROCEDURE treat_a_ris_rsp;
     BEGIN
          CASE estado ACPM OF
               STA4 : IF a_ris_rsp(+)
                         THEN envia_rlre_apdu(+)
                              muda estado para STA0
                         ELSE envia_rlre_apdu_(-)
                              muda estado para STA5
                      END
               STA6 : IF a_ris_apdu(+)
                         THEN envia_rlre_apdu(+)
                              muda estado para STA3
                         ELSE call_errors
                      END
*OBS           OTHERS call_errors
          END
     END;
```

*OBS : Nos procedimentos a_asc_req , a_ris_req , a_asc_rsp e a_ris_rsp seguiu-se o que o documento <3> previa em seu item A.3.1, alínea "a", que trata das intersecções inválidas - "Se um evento de entrada advem do usuário do *ACSE* , qualquer ação tomada pela ACPM deve ser local" - com isso o operador , através dos erros recebidos , terá condição de tomar as providências

necessárias ao restabelecimento da comunicação.

3.3 - Especificação detalhada de realização do processo ACSE (DD - P_ACSE)

```
/* Processo ACSE */
#include "master.arq"
#include "globtes.h"
#define "TEST_ACSE 5
                /* Estrutura de Dados do P_ACSE */
/* Tipos do ACSE */
typedef enum   /* Tipos de Estados o ACSE */
        { sta0 , sta1 , sta2 , sta3 , sta4 , sta5 , sta6 , sta7
        }Acse_state_type;
typedef enum   /* Tipos de Usuários do ACSE */
        {called , calling
        {Acse_user_type;

      /* Tabela de Controle da Máquina de Estados do ACSE */
typedef struct
       {Connection_id    ident_connect;    /*  identificador  da
conexão*/
        Acse_state_type  acse_state;    /* estado da transação */
        Acse_user_type   user_type;     /* tipo de usuário do  acse
```

```
*/
        }Acse_tab;

                        /* Variáveis do ACSE */
Acse_tab          acse_tab [Max_context];  /* tabela de estados */
Block_struct    *block;    /* bloco de mensagem do sistema  didático
*/.
Prot_version    prot_vers;      /* versão do protocolo ACSE */
uint8           index_tab;
                /* Portas de entrada e saída do ACSE */
T_QUEUE     *mbi_mms;
T_QUEUE     *mbi_pres;
T_QUEUE     mbo_mms;
T_QUEUE     mbo_pres;
T_QUEUE     mbo_op;

/* Definição das variáveis "TRUE" e "FALSE" */
Boolean     false = 0;
Boolean     true= 1;

                        /* Programa Principal */

#include "progtes.c"
      p_acse (T_QUEUE *ingate_mms,  T_QUEUE  *ingate_pres,  T_QUEUE
```

```
*outgate_mms, T_QUEUE *outgate_pres, T_QUEUE *outgate_op)
        {
         int     wait_time = 0;
/*       int (prot_vers)  = 0; */
         mbi_mms    = ingate_mms;
         mbi_pres   = ingate_pres;
         mbo_mms    = outgate_mms;
         mbo_pres   = outgate_pres'
         mbo_op     = outgate_op;
         init();
         for (;;)  /* loop eterno */
           {
            if (espera_mens (mbi_mms, wait_time, CONSOME,  &block)
                == OPERAÇÃO_OK)
             {
              treat_mms ();
             }
            else
             {
              if (espera_mens (mbi_pres, wait_time,  CONSOME,  &block)
                  == OPERAÇÃO_OK)
              {
               treat_pres ();
              }
```

```
            }
          }
        }
                    /* Fim do Programa Principal */

                              /* Funções */
/* Função obtenção do index para controle da Máquina de Estados do
                              ACSE */
int  find_index (Connection_id      a)
       {
        int i = 1;
        while ((a  !=  acse_tab  [i]  .ident_connect)  &&  (  i  <=
Max_context)) ++i;
       return (i);
       }
                    /* Inicialização de variáveis */

      init ()
        {
         int     index_tab = 1;
         do
          {
           acse_tab [index_tab] . acse_state = sta0;
           acse_tab [index_tab++] . ident_connect = 0;
```

```
        }
        while (index_tab <= Max_context);
       }

/* Procedimento de tratamento das mensagens recebidas do MMS */
    treat_mms ()
      {
       switch (block -> primitive )
          {
           case request : treat_mms_request (); break ;
           case response_pos : treat_mms_response (); break;
           case response_neg : treat_mms_response (); break;
           default :call_error (1);
          }
       }

/* Procedimento de tratamento de mensagens "Request" recebida do
                                MMS */
     treat_mms_request ()
       {
        switch (block -> data . acse_serv . service_name )
          {
           case a_associate : treat_a_asc_req (); break;
           case a_release   : treat_a_rls_req (); break;
```

```
            case a_abort      : treat_a_abr_req (); break;
            default           : call_error (2);
           }
        }

/* Procedimento de tratamento da mensagem "A_associate_request"
                          recebida do MMS */
     treat_a_asc_req ()
       {
        index_tab = find_index (block -> connection_id);
        if (index_tab <= Max_context) /* já existe a associação */
         {
          call_error (3);
         }
        else
         {
          index_tab = find_index (0); /* encontra posição vaga na
tabela de contextos */
          prot_vers . version1 = 1;   /* versão aceita por esta
implementação */
          send_aarq_apdu (true);
          acse_tab [index_tab] . ident_connect = block ->
connection-id; /* inclue nova associação na tabela de contextos do
ACSE */
```

```
            acse_tab  [index_tab] . user_type = calling;
            acse_tab  [index_tab] .acse_state = sta1;
          }
        }

/* Procedimento de  tratamento  de  mensagem  "A_release_request"
                          recebida do MMS */
      treat_a_rls_req ()
        {
         index_tab = find_index (block -> connection_id);
         if (acse_tab [index_tab] . acse_state == sta5)
          {
           send_rlrq_apdu ();
           acse_tab  [index_tab] .acse_state = sta3;
          }
         else   call_error (4);
        }

/*  Procedimento  de  tratamento  de  mensagem  "A_abort_request"
                          recebida do MMS */
       treat_a_abr_req ();
        {
         index_tab = find_index (block -> connection_id);
         if (acse_tab [index_tab] . acse_state == sta0)
```

```
        {
         call_error (5);
        }
       else
        {
         send_abrt_apdu (acse_service_user);
         acse_tab  [index_tab] .acse_state = sta0;
        {
      {

/*  Procedimento de tratamento das mensagens "MMS_response
                        recebidas do MMS */
     treat_mms_response ()
       {
        switch (block -> data . acse_serv . service_name )
          {
           case a_associate : treat_a_associate_rsp (): break;
           case a_release   : treat_a_rls_rsp ();      : break;
           default          : call-error (6);
          }
       }

/*      Procedimento      de      tratamento      da      mensagem
            "A_associate_response" recebida do MMS */
```

```
treat_a_assoc_rsp ()
  {
   index_tab = find_index (block -> connection_id);
   if (acse_tab [index_tab] . acse_state == sta2)
    {
     if (block -> data . acse_serv . prim_acse .
a_associate_serv . a_associate_res . associate_pdu . result ==
accepted )
      {
       send_aare_pos_apdu (); /* estabelecimento da conexão
bem sucedida no respondedor (result=accepted , diag=null) */
       acse_tab [index_tab] .acse_state = sta5;
      }
     else
      {
        send_aare_neg_apdu (acse_service_user); /*
estabelecimento da conexão mal sucedida no respondedor */
       acse_tab [index_tab] .acse_state = sta0;
      }
    }
   else call_error (7);
  }

/* Procedimento de tratamento da mensagem
```

```
                "A_release_response" recebida do MMS */
      treat_a_ris_rsp ()
        {
         index_tab = find_index (block -> connection_id);
         if (acse_tab [index_tab] . acse_state == sta4)
          {
           if (block -> data . acse_serv . prim_acse .
a_release_serv . a_release_res . result == accepted )
            {
             send_rire_apdu (); /* release aceito pelo respondedor
(result=affirmative) */
             acse_tab [index_tab] .acse_state = sta0;
            }
           else
            {
             send_rire_apdu (); /* release negado pelo respondedor
(result=negative) */
             acse_tab [index_tab] .acse_state = sta5;
            }
          }
         else if (acse_tab [index_tab] . acse_state == sta6)
                {
                 if (block -> data . acse_serv . prim_acse .
a_release_serv . a_release_res . result == accepted )
```

```
                    {
                      send_rlre_apdu  ();  /*  release  aceito  pelo
respondedor (result=affirmative) */
                     acse_tab [index_tab] .acse_state = sta3;
                    }
                   else call_error (8);
                    }
        }
/* Procedimento de tratamento de mensagens recebidas da Camada  de
                Apresentação (indication , confirm) */
      treat_pres ()
        {
         switch (block -> primitive)
            {
             case indication  : treat_pres_indication (); break;
             case confirm_pos : treat_pres_conf_pos () ; break;
             case confirm_neg : treat_pres_conf_neg () ; break;
             default          : call_error (6); break;
            }
         }

 /* Procedimento de tratamento de mensagens "Indication"  recebidas
                      da Camada de Apresentação */
       treat_pres_indication ()
```

```
        {
         switch (block -> data . pres_serv . service_name )
            {
             case p_connection : treat-p-con-ind ()  ; break;
             case p_release    : treat_p-rel-ind ()  ; break;
             case p_u_abort    : treat_p_u_abr ()    ; break;
             case p_p_abort    : treat_p_p_abr_ind (); break;
             default           : call_error (6);
            {
        {
/* Procedimento de tratamento de mensagem "P_connection_indication"
                recebida da Camada de Apresentação */
      treat_p_conind ()
        {
          if (block -> data . pres_serv . prim_pres .
p_connection_serv . p_connection_ind . valid_pdu == true ) /*
verifica a validade da pdu recebida como "user_data" na primitiva
"p_conn_ind" */
          {
           treat_aarq_apdu ();
          {
         else treat_invalid_intersections ();
        }
```

```
/* Procedimento de tratamento de  mensagem  "P_release_indication"
                recebida da Camada de Apresentação */
     treat_p_rls_ind ()
       {
        if (block -> data . pres_serv . prim_pres . p_release_serv
. p_release_ind . valid_pdu == true )
          {
           treat_rlrq_apdu ();
          }
         else treat_invalid_intersections ();
        }

/* Procedimento de tratamento de  mensagem  "P_u_abort_indication"
                recebida da Camada de Apresentação */
     treat_p_u_abr ()
       {
        if (block -> data . pres_serv . prim_pres . p_u_abort_serv
. p_u_abort_ind . valid_pdu == true )
          {
           treat_abrt_apdu ();
          }
         else treat_invalid_intersections ();
        }
```

```
/* Procedimento de tratamento de mensagem "P_P_abort_indication" */

     treat_p_p_abr_ind ();
       {
         index_tab = find_index (block->connection_id);
         if (acse_tab [index_tab] . acse_state == sta0)
          {
           treat_invalid_intersections ();
          }
         else
          {
           send_a_p_abr_ind ();
           (acse_tab [index_tab] . acse_state = sta0);
          }
       }

/* Procedimento de tratamento de  mensagem  Confirmation  positiva
                recebida da Camada de Apresentação */

     treat_pres_conf_pos ()
       {
        switch (block->data .prse_serv . service_name)
          {
           case p_connection       :   treat_aare_apdu ();break;
           case p_release          :   treat_rlre_apdu (),break;
```

```
            default                    :    call_error (6);
           }
        }

/* Procedimento de tratamento de mensagem Confirmation negativa
                recebida da Camada de Apresentação */

      treat_pres_conf_neg ()
        {
         index_tab = find_index (block->connection_id);
         if (acse_tab [index_tab] . acse_state == sta1)
           {
           send_a_asc_cnf (confirm_neg,rejected_permanent,presen
tation_service_provider,false);
           acse_tab [index_tab] . acse_state = sta0;
           }
         else treat_invalid_intersections ();
        }

/* Procedimento de tratamento da AARQ_APDU */

       treat_aarq_apdu ()
         {
         index_tab = find_index(0);
```

```
        if(bolck->data .pres_serv  .prim_pres.p_connection_serv  .
p_connection_ind . user_data . protocol_version_valid == true)
/* foi enviado o parâmetro versão do protocolo */
           {
              if (block->data . pres_serv . prim_pres . p_connection_
serv . p_connection_ind . user_data . protocol_version . version1
== prot_vers.version1)
              {
                send_a_assoc_ind ();
                acse_tab[index_tab].ident_connect= block->
connection_id;
                acse_tab[index_tab].acse_state = sta2;
                acse_tab[index_tab].user_type = called;
                else send aare_neg_apdu (acse_service_provider);
                 }
          else
                {
                 send_a_assoc_ind ();
                 acse_tab[index_tab].ident_connect = block->
connection_id;
                 acse_tab[index_tab].acse_state = sta2;
                 acse_tab[index_tab].user_type = called;
                }
               }
           }
```

```
/* Procedimento de tratamento da RLRQ_APDU */
     treat_rlrq_apdu ()
       {
        index_tab = find_index (block->connection_id);
        if (acse_tab [index_tab] . acse_state == sta3;
         {
          if (acse_tab [index_tab] . user_type == calling)
           {
            send_a_rls_ind();
            acse_tab [index_tab] . acse_state = sta6;
           }
          else
            {
             send_a_rls_ind();
             acse_tab [index_tab] . acse_state = sta7;
            }
         }
        else if (acse_tab [index_tab] . acse_state == sta5)
              {
               send_a_rls_ind();
               acse_tab [index_tab] . acse_state = sta4;
              }
            else treat_invalid_intersections ();
```

```
        }

/* Procedimento de tratamento da ABRT_APDU */

      treat_abrt_apdu ()
        {
         index_tab = find_index (block-> connection_id);
         if (acse_tab [index_tab] . acse_state == sta0)
          {
            call_error (8);
          }
         else
           {
            send_a_abr_ind ();
            acse_tab [index_tab] . acse_state = sta0;
           }
        }

/* Procedimentos de tratamento AARE_APDU */

      treat_aare_apdu ()
        {
          index_tab = find_index (block->connection_id);
          if (block->data . pres_serv . prim_pres . p_connection_serv
```

```
. p_connection_cnf . valid_apdu == true)
          {
           if (block->data . pres_serv . prim_pres . p_connection_serv
. p_connection_cnf . result == acceptance)
/* isto significa que a APDU contida na primitiva
P_connect_response_pos é do tipo AARE+*/
            {
             if (acse_tab [index_tab] .  acse_state == sta1)
               {
                send_a_asc_cnf (confirm_pos, accepted,
 acse_service_user, false);
                (acse_tab [index_tab] .  acse_state == sta5;
               }
             else treat_invalid_intersections ();
            else if (acse_tab [index_tab] .  acse_state == sta1
                  {
                   send_a_asc_cnf_neg ();
                   acse_tab [index_tab] .  acse_state = sta0;
                  }
                 else call_error (9);
               }
              else treat_invalid_intersections ();
          }
        }
```

```
/* Procedimento de tartamento da RLRE_APDU */

     treat_rlre_apdu ()
       {
        index_tab = find_index (block->connection_id);
        if (block->data . pres_serv . prim_pres . p_release_serv
. P_release_cnf . valid_pdu == true)
         {
          if (block->data . pres_serv . prim_pres . p_release_serv
. P_release_cnf . result == affirmative)
/*   isto   significa   que   a   PDU   contida   na   primitiva
P_release_response_pos é do tipo RLRE+ */
           {
           if (acse_tab [index_tab] .  acse_state == sta3)
            {
             send_a_rls_cnf (confirm_pos);
            acse_tab [index_tab] .  acse_state = sta0;
            }
           else
             {
              if (acse_tab [index_tab] .  acse_state == sta7)
               {
                send_a_rls_cnf (confirm_pos);
                acse_tab [index_tab] .  acse_state = sta4;
```

```
                }
                else treat-invalid-intersections ();
              }
            }
          else
           {
            if (acse_tab [index_tab] .  acse_state == sta3)
             {
              send_a_ris_cnf (confirm_neg);
              acse_tab [index_tab] .  acse_state == sta5;
             }
            else treat-invalid-intersections ();
           }
          }
        else treat-invalid-intersections ();
        }

/* Procedimento de tratamento de invalid-intersections

      treat_invalid_intersections ()
        {
         send_a_abr_ind ();
         pede_mem (sizeof (block_struct) , block);
         send_abrt_apdu (acse_service_provider);
```

```
        acse_tab [index_tab] .  acse_state = sta0;
        acse_tab [index_tab] .  ident_connect = sta0;
       }

/* Envia primitiva a_abort_indication ao MMS */

     send_a_abr_ind ();
       {
        block->primitive = indication;
        block->service = acse;
        block->data . acse_serv . service_name = a_abort;
        block->data . acse_serv . prim_acse . a_abort_serv .
a_abort_ind . source = acse_service_provider;
       envia_mens (mbo_mms , block);
        }

/* Envia primitiva a_associate_confirmation ao MMS */

     send_a_asc_cnf (Primitive_type  result1, Assoc_result  resut2,
Source  source, Boolean  result4, Assoc_diagnostic  result5)
       {
        block->primitive = result1;
        block->service = acse;
        block->data . acse_serv . service_name = a_associate;
```

```
        block->data . acse_serv . prim_acse . a_associate_serv .
a_associate_res . associate_pdu . protocol_version_valid =
false;
        block->data . acse_serv . prim_acse . a_associate_serv .
a_associate_res . associate_pdu . resutl = result2;
        block->data . acse_serv . prim_acse . a_associate_serv .
a_associate_res . associate_pdu . resut_source = source;
        block->data . acse_serv . prim_acse . a_associate_serv .
a_associate_res . associate_pdu . diagnostic_valid = result4;
        block->data . acse_serv . prim_acse . a_associate_serv .
a_associate_res . associate_pdu . diagnostic = result5;
/*   os   demais   parâmetros   são   os   mesmos   da   primitiva
p_connection_confirmation */
        envia_mens (mbo_mms , block);
        }

/*  Envia  aarq_apdu  mapeada  como  user  data  na  primitiva
     p_connect_request */

     send_aarq_apdu (Boolean  result)
       {
        block->primitive = request;
        block->service = acse;
        block->data . pres_serv . service_name = p_connection;
```

```
        block->data . pres_serv . prim_pres . p_connection_serv .
p_connection_req . user_data . protocol_version_valid =result;
       block->data . pres_serv . prim_pres . p_connection_serv .
p_connection_req . user_data . protocol_version = prot_vers;
/*   os   demais   parâmetros   são   os   mesmos   da   primitiva
a_associate_request */
       envia_mens (mbo_pres , block);
       }

/* Envia rlrq_apdu mapeada como user data na primitiva
p_release_request */

     send_rlrq_apdu ()
       {
        block->primitive = request;
        block->service = acse;
        block->data . pres_serv . service_name = p_release;
/*   os   demais   parâmetros   são   os   mesmos   da   primitiva
a_release_request */
       envia_mens (mbo_pres , block);
       }

/* Envia abrt_apdu mapeada como user data na primitiva
p_u_abort_request */
```

```
     send_abrt_apdu (Source  result)
       {
        block->primitive = request;
        block->service = acse;
        block->data . pres_serv . service_name = p_u_abort;
        block->data . pres_serv . prim_pres . p_u_abort_serv .
p_u_abort_req . user_data_valid = true;
       block->data . pres_serv . prim_pres . p_u_abort_serv .
p_u_abort_req . user_data . source = result;
/*   os   demais   parâmetros   são   os   mesmos   da   primitiva
a_abort_request */
       envia_mens (mbo_pres , block);
       }

/* Envia  aare_apdu  mapeada  como  user  data  na  primitiva
p_connect_response */

     send_aare_apdu ()
       {
        block->primitive = response_pos;
        block->service = acse;
        block->data . pres_serv . service_name = p_connection;
        block->data . pres_serv . prim_pres . p_connection_serv .
p_connection_res . result = acceptance;
       block->data . pres_serv . prim_pres . p_connection_serv .
```

```
p_connection_res . user_data_valid = true;
        block->data . pres_serv . prim_pres . p_connection_serv .
p_connection_res . user_data . protocol_version_valid = true;
       block->data . pres_serv . prim_pres . p_connection_serv .
p_connection_res . user_data . protocol_version = prot_vers;
/*   os   demais   parâmetros   são   os   mesmos   da   primitiva
a_associate_response */
       envia_mens (mbo_pres , block);
       }

/*  Envia  rlre_apdu  mapeada  como  user  data  na  primitiva
p_release_response_pos */

     send_rlre_apdu ()
       {
        block->service = acse;
        block->data . pres_serv . service_name = p_release;
/*   os   demais   parâmetros   são   os   mesmos   da   primitiva
a_release_response_pos */
       envia_mens (mbo_pres , block);
       }

/*  Envia  a_p_abort_indication */

      send_a_p_abr_ind ()
```

```
        {
         block->primitive = indication;
         block->service = acse;
         block->data . acse_serv . service_name = a_p_abort;
/*   os   demais   parâmetros   são   os   mesmos   da   primitiva
a_abort_indication */
        envia_mens (mbo_mms , block);
        }

/* Envia a_release_confirme_positive ao  MMS  */

      send_a_ris_cnf (primitive_type  result)
        {
         block->primitive = result;
         block->service = acse;
         block->data . acse_serv . service_name = a_release;
/*   os   demais   parâmetros   são   os   mesmos   da   primitiva
p_release_confirme_pos */
         envia_mens (mbo_pres , block);
         }

/*  Envia  aare_negative_apdu na  primitiva p_connect_response */

      send_aare__neg_apdu (Source  result)
```

```
        {
         block->primitive = response_neg;
         block->service = acse;
         block->data . pres_serv . service_name = p_connection;
         if (result == acse_service_provider)
          {
           block->data . pres_serv . prim_pres . p_connection_serv .
p_connection_res . result = user_rejection;
           block->data . pres_serv . prim_pres . p_connection_serv .
p_connection_res . user_data_valid = true;
           block->data . pres_serv . prim_pres . p_connection_serv .
p_connection_res . user_data . result = rejected_permanent;
           block->data . pres_serv . prim_pres . p_connection_serv .
p_connection_res . user_data . diagnostic_valid = true;
           block->data . pres_serv . prim_pres . p_connection_serv .
p_connection_res.          user_data.          diagnostic          =
no_common_acse_version;
          }
         envia_mens (mbo_pres , block);
        }

/* Envia a_associate_confirm_negative ao MMS */

      send_a_asc_cnf_neg ()
```

```
        {
         block->service = acse;
         block->data . acse_serv . service_name = a_associate;
         if (block->data . pres_serv . prim_pres . p_connection_serv .
p_connection_cnf . user_data . diagnostic_valid != true;
/* significa que quem gerou a AARE- foi o ACSE */
          {
           block->data . acse_serv . prim_pres . a_associate_serv .
a_associate_res . associate_pdu . result_source =
acse_service_provider;
           else block->data . acse_serv . prim_pres . a_associate_serv
. a_associate_res . associate_pdu . result_source
= acse_service_user;
          . envia_mens (mbo_mms , block);
          }
        }

/* Envia a_release_indication ao MMS */

      send_a_rls_ind ()
        {
         block->primitive = indication;
         block->service = acse;
         block->data . acse_serv . service_name = a_release;
```

```
/*   os   demais   parâmetros   são   os   mesmos   da   primitiva
p_release_indication */
        envia_mens (mbo_mms , block);
        }

/* Envia primitiva a_associate_indication ao MMS */

     send_a_assoc_ind ();
       {
        block->primitive = indication;
        block->service = acse;
        block->data . pres_serv . service_name = a_associate;
        block->data . acse_serv . prim_acse . a_associate_serv .
a_associate_req . associate_pdu . protocol_version_valid  =
false;
/*   os   demais   parâmetros   são   os   mesmos   da   primitiva
p_connection_indication */
         envia_mens (mbo_mms , block);
        }

call_error (uint8 code_error)
        {
         #if defined (debug)
             printf ("\n Erro de execução  do  programa  número  %d
     \n" ,code_error );
```

```
        endif
    }
```

4.1 - Exemplo de Execução do ACSE no SISIDI_MAP.

Esta seção apresenta um exemplo de execução dos serviços disponíveis no *ACSE* , segundo uma visão da troca de mensagens de comunicação entre os processos do *SISDI_MAP* , conforme ilustrado na figura 1.1.

Este exemplo complementa o "*Exemplo de Execução dos Serviços MMS no SISDI_MAP* <11>" , sendo que aqui será dada uma visão mais abrangente dos serviços envolvidos com o processo *ACSE*.

4.2 - Estabelecimento do Contexto entre os APs.

O *PROC_AP* do controlador interage com o *PROC_API* , que inicia o envio de primitivas que seguem a sequência abaixo para a solicitação do estabelecimento de contexto :

PROC_API ----> PROC_MMS ----> PROC_ACSE ----> PROC_PRESENTATION ----> PROC_OP ----> PROC_PRESENTATION ----> PROC_ACSE ----> PROC_MMS ----> PROC_API .

Serão mostrados as primitivas trocadas na sequência acima cujas respostas possuem estruturas semelhantes.

4.2.1. - Primitiva "INITIATE_REQUEST".

O PROC_API , via primitiva "*pede_mem*" do núcleo , preenche os

parâmetros desta mensagem e coloca-a na porta do *"Proc_MMS"* (via primitiva *"envia_men"* do núcleo).

- connection_id : 0
- primitive : request
- service : mms_cont
- data_area_lenght : 296
- data :mms_cont_serv :
    - service_name : initiate
    - dummy [5] :
    - prim_cont_manag : initiate_serv :
        - dummy [256] :
        - initiate_req_pdu :
            - list_of_version_numbers : 1 0 0 0 0 0 0 0
            - prop_max_msg_size_valid : 1 (true)
            - max_msg_size : 256
            - prop_max_serv_outst_calling : 5
            - prop_max_serv_outst_called : 7
            - prop_data_str_nest_level_valid : 0 (false)
            - prop_data_str_nest_level :
            - prop_max_nesting_level :3
            - prop_horizontal_cbb : str1 , socs , rcv1
            - client_vert_cbb_calling : con1 , con2 , con3...
            - server_vert_cbb_calling : con1 , con2 ,

con3...

4.2.1.1 - Primitiva "A_Associate_Request"

O Proc_MMS recebe a mensagem do ítem 4.2.1. através da porta do Proc_MMS alocada para tal e, utilizando-se do mesmo bloco de memória , coloca-a na porta do Proc_ACSE.Os parâmetros da primitiva "*A_associate_request*" são vistos pelo Proc_ACSE com a seguinte máscara:

- connection_id : 0
- primitive : request
- service : acse
- data_area_lenght : 296
- data :acse_serv :
    - service_name : associate :
        - associate_req :
            - dummy :
            - calling_pres_address : 1 0 0
            - called_pres_address : 2 0 0
            - pres_context_definition_list_valid :0 (false)
            - pres_context_definition_list :
            - pres_context_def_list_result_valid : 0 (false)
            - pres_context_def_list_result :
            - default_pres_context_name_valid : 0 (false)
            - default_pres_context_name :

```
- qualit_of_service : 1
- pres_requirements_valid : 0 (false)
- pres_requirements :
- mode : 1 (normal)
- session_requirements : 132
- initial_synchr_point_serial_number : 12
- initial_assigment_of_tokens : 1
- associate_pdu :
    - prot_version_valid : 1 (true)
    - prot_version : 1
    - applic_context_name_valid : 0 (false)
    - applic_context_name :
    - calling_AP_title_valid : 0 (false)
    - calling_AP_title :
    - calling_AE_qualifier_valid : 0 (false)
    - calling_AE_qualifier :
    - calling_AP_inv_ident_valid : 0 (false)
    - calling_AP_inv_ident :
    - calling_AE_inv_ident_valid : 0 (false)
    - calling_AE_inv_ident :
    - called_AP_title_valid : 0 (false)
    - called_AP_title :
    - called_AE_qualifier_valid : 0 (false)
    - called_AE_qualifier :
    - called_AP_inv_ident_valid : 0 (false)
    - called_AP_inv_ident :
```

- called_AE_inv_ident_valid : 0 (false)
- called_AE_inv_ident :
- user_information_valid : 1 (true)
- user_information : "Initiate_request"

Pode-se notar que a *"APDU"* do ACSE correspondente a *"Associate_pdu"* (**AARE_APDU**) já é montada na estrutura da mensagem para o ACSE , que deverá preencher somente o campo *"protocol_version_valid"* e *"protocol-version"*, pois todos os outros já vem prenchidos pela *"API"* e o *"MMS"*.

O ACSE, neste caso, está fazendo apenas o papel de repassador dos parâmetros que já vêm preenchidos pelas camadas superiores , gerando para isso a primitiva para o *"Proc_presentation"* e a APDU correspondente para o ACSE par.

Dentro do parâmetro *"user_information"* encontra-se toda a estrutura da primitiva *"Initiate_request"*.O *"Proc_acse"* ao receber a primitiva *"A_associate_request"* guarda a informação da *"AA"* estabelecida na tabela indexada pelo identificador da conexão que controla a máquina de estados finitos, conforme mostrado na figura 4.1.

Caso a resposta a essa requisição - *"A_associate_response"*- contenha a *APDU "AARE_APDU"* com uma resposta negativa ao estabelecimento da *"AA"* , a tabela mostrada na figura 4.1 será reinicializada.

INDEXADO PELO "CONNECTION_ID"

IDENTIFICADOR DA CONEXÃO	CONNECTION_ID		
ESTADO DA TRANSAÇÃO	STA1		
TIPO DO USUÁRIO ACSE	CALLING		

Figura 4.1 - Tabela de estabelecimento de conexão

4.2.1.2 - Primitiva "P_connection_request".

O Proc_acse trata a mensagem do ítem 4.2.1.1 , e utilizando-se do mesmo bloco de memória , coloca-a na porta do *Proc_Presentation*.

Os parâmetros da primitiva "*P_connection_request*" são vistos pelo "*Proc_Presentation*" com a seguinte máscara :

- connection_id : 0
- primitive : request
- service : presentation
- data_area_lenght : 296
- data : pres_serv :
    - service_name : p_connection
    - prim_pres : p_connection_serv :

- p_connection_req :
    - dummy1 :
    - calling_presentation_address : 1 0 0
    - called_presentation_address : 2 0 0
    - .
    - .
    - .
    - user_data_valid : 1 (true)
    - user_valid : "initiate_request"

O "*Proc_Presentation*", como apenas simula a Camada de Apresentação , não se utiliza dos demais parâmetros da primitiva "*P_connection*".

O "*Proc_Presentation*" armazena o bloco recebido e envia a mensagem de solicitação de transferência de "*APDU*" ao "*Proc_OP*" que responde a esta solicitação, indicando se deseja ou não inserção de erros <11> , o "*Proc_Presentation*" comuta a primitiva (trocando o "connection_id") enviando ao "*Proc_acse*" a mensagem do Bloco de Protocolo armazenada , alterando apenas o tipo de primitiva (parâmetro "primitive" no bloco) de "*request*" para "*indication*" , e assim é feito pelos outros processos até a mensagem chegar ao "*Proc_API*".

Neste instante, o ciclo é novamente exercitado , desta vez para as primitivas "*response*" e "*confirm*" , num esquema semelhante ao descrito nos ítens anteriores.

5 - CONCLUSÕES.

O modelo de referência **RM-OSI/ISO** foi um importante marco na solução de problemas complexos como a interconexão de sistemas computacionais heterogêneos.

Gasta-se muito esforço e dinheiro , ainda hoje , para se interligar e interrelacionar sistemas computacionais de diferentes fabricantes e tecnologias, de modo a se ter, do ponto de vista do usuário localizado na ponta da rede , uma visão transparente da mesma.

Neste sentido, a implementação de protocolos diversos dentro da filosofia **OSI-ISO** passou a ser uma tarefa importante , dentro do ambiente industrial que é um grande consumidor de equipamentos diversos , no atual momento em que se busca atingir a conectividade em todos os níveis do universo industrial : *MAP* , *TOP* , *CAE* , *CAD* , *CAM*; enfim, a busca do *CIM (Computer Integrated Manufacturing)* .

Então , o estudo de protocolos em um ambiente universitário é uma ferramenta importante para que se formar profissonais com algum conhecimento básico para um mercado ávido por esses conhecimentos.

É neste ponto que se encaixa a proposta de se fazer um sistema didático - "**SISDI_MAP**" - em que se pretende que outras implementações sejam feitas ao longo do tempo e , com isso ,

possa-se vir futuramente a atingir o objetivo maior deste projeto que é um sistema real de comunicação entre equipamentos, totalmente em conformidade às especificações do MAP.

Outro aspecto importante foi mostrar , especificamente neste trabalho , a importância da definição do contexto em que vai se desenrolar a aplicação e que tipos de parâmetros são relevantes para isto.

O *ACSE* mostrou-se um protocolo de aspecto simples porém importantíssimo, uma vez que com sua utilização nenhum protocolo aplicativo tem a necessidade de se preocupar com o estabelecimento da conexão.

Fica, assim, a Camada de Aplicação responsável por seu aspecto mais importante que é o de se ocupar com a aplicação propriamente dita.

Imagina-se que o *ACSE* não foi previsto inicialmente na concepção do modelo OSI e que surgiu mais tarde como uma necessidade de se ter um protocolo único, e de aspecto modular, para o controle de uma associação de aplicação.

Esta conclusão foi baseada na análise do protocolo MHS-1984 < 19> em que se nota alguns serviços de controle de aplicação próprios do protocolo.

O ACSE provê uma máquina de controle do protocolo específico para o MHS-1984, porém o mesmo não utiliza os aspectos mais relevantes de controle de associação do ACSE, pois em sua implementação não há troca de APDUS_ACSE : o ACSE funciona apenas

como um repassador de dados do MHS-1984.

Verificou-se no capítulo 1 que o *SACF* modela o uso das regras definidas no contexto de aplicação que dizem respeito às iterações entre dois ASEs e também coordena a interação com a camada de Apresentação.A *ISO* ainda não possui um protocolo definindo o *SACF* , entretanto ao se observar o perfil funcional do *MAP* vê-se que existe um protocolo a nível de aplicação chamado **API** <7> que coordena a interação entre o ACSE , o MMS e a Aplicação.Isto leva a crer que apesar de a ISO não ter definido o papel do SACF, a GM na definição do perfil funcional do MAP ,já prevendo a necessidade de um protocolo com funções semelhantes as definidas para o SACF, definiu o API.

O modelo de implementação está centrado numa estrutura de comunicação que utiliza o conceito de comunicação através de "mailboxes"<11>.

O total de código do PROC_ACSE foi de aproximadamente 24 kbytes.

6 - REFERÊNCIAS BIBLIOGRÁFICAS.

[1] MENDES ,M. J. - Comunicação Fabril e o projeto MAP/TOP , IV Escola Brasil-Argentina de Informática (EBAI) , janeiro/1989.

[2] ISO/DIS 7498 - "Information Processing Systems - Open System Interconnection - Basic Reference Model", setembro/1989.

[3] CCITT X.217 - Association Control Service Definition for Open Systems Interconnection for CCITT Applications - DRAFT Recommendation. / X.227 - Association Control Service Definition for Open Systems Interconnection for CCITT Applications - DRAFT Recommendation, dezembro/1987

[4] GM - MAP 3.0 - "MMS Application Interface Specification and Connection Management Interface Specification" , março/1988

[5] PAGLIONE , A. J. e outros - Aspectos de Implementação do Protocolo RS - 511 do MAP , Anais do 3º CONAI , Setembro/1988.

[6] ISO/DIS 9506 - "Manufacturing Message Specification , Part 1 : Service Specification , Part 5 : Protocol Specification" , DRAFT 6 , Agosto/1988

CAPÍTULO 6 : REFERÊNCIAS BIBLIOGRÁFICAS

[7] MADEIRA , E. R. M. e MENDES , M. J. - Interfaces de Programas de Aplicação para o Protocolo MMS (RS551) , Anais do 3º CONAI , Setembro/1988.

[8] PAGLIONE , A. J. e outros - SISDI_MAP - Sistema Didático do Protocolo e da Interface de Aplicação MMS do MAP , Anais do Seminário Franco - Brasileiro em Sistemas Informáticos Distribuídos , setembro/1989.

[9] ZABEU , M. C. - Um Modelo de Núcleo , Tese de Mestrado apresentada na FEE-UNICAMP , Novembro/1989.

[10] LIMA , J. M. S. - Uma Interface de Comunicação Homem-Máquina para execução de Protocolos. Trabalho apresentado como tópicos especiais na UNICAMP, outubro/1989.

[11] CARVALHO , D. A. J. - Aspectos de Especificação e Implementação da Estrutura de Mensagem de um Sistema Didático do Protocolo MMS , Tese de Mestrado , FEE-UNICAMP , Novembro/1989.

[12] ISO/IEC DIS 9545 - "Information Processing Systems - Open Systems Interconnection - Application Layer Structure", January/1989.

[13] ISO DP 9579 - Remote Database Access (RDA) - 3rd Working Draft Stage , January/1988.

CAPÍTULO 6 : REFERÊNCIAS BIBLIOGRÁFICAS

[14] CCITT X.219/X.229 - "ROSE - Remote Operations : Services and Protocol Specification".

[15] ISO/DIS 8649/3 - "Information Processing - Open Systems Interconection - Definition of Common Aplication Service Elements and" ISO/DIS 8650/3 - "Specification of Protocols for Common Aplication Service Elements - Part 3 : Commitment , Concurrency and Recovery", April/1986.

[16] ISO/DIS 7498/3 - "Naming and Addressing - Part 3 to the OSI Reference Model"

[17] CCITT X.216 - Presentation Service Definition for Open Systems Interconnection for CCITT Applications - DRAFT Recommendation.

[18] ISO/DIS 8824 - "Information Processing Systems - Open System Interconnection - Specification of Abstract Syntax Notation One (ASN.1)" , Setembro/1986.

[19] Recommendation x410-1984 - CCITT Recommendation x410 - Message Handling Systems : Remote Operation and Reliable Transfer Server / x411-1984 - Massage Transfer Layer.

[20] ISO/DIS 8326 AND 8327 - "Session service definition and Session protocol specification for Open System Interconnection.