UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA DA COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

"ASPECTOS DE ESPECIFICAÇÃO E IMPLEMENTAÇÃO DA CAMADA DE APRESENTAÇÃO DO PADRÃO MAP UTILIZANDO O AMBIENTE EPOS"

por : Paulo César Minoru Inazumi

orientador : Prof. Dr. Manuel de Jesus Mendes

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

Novembro/1990

Este trabalho contou com o apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq

AGRADECIMENTOS E DEDICATÓRIA

Ao CNPq, pelo incentivo financeiro.

À Universidade Estadual de Campinas, pelas condições propícias de trabalho.

Ao Prof. Dr. Manuel de Jesus Mendes, pela orientação ao longo do desenvolvimento deste trabalho.

Ao Prof. Dr. Maurício Ferreira Magalhães pela co-orientação e auxílio na busca de suporte de equipamentos.

Ao grupo SISDI-MAP, em especial a amiga Verônica, pelas inúmeras discussões, e ao companheiro de trabalho Carlos Mendez, com o qual caminhamos juntos neste trabalho.

Aos colegas Paulo Maurício, Ruy, Humberto, Sakahi e em especial aos amigos desde os remotos tempos de graduação Hsieh e Rodrigo.

Ao grande amigo Samuel E. de Lucena pelo companheirismo e os bons momentos de convivência.

Aos amigos Hélio, Isaias e Maurício pelo apoio na conclusão deste trabalho.

Aos amigos do LCA e a todos aqueles que, direta ou indiretamente, contribuiram para a realização deste trabalho.

Finalmente, porém o mais importante, aos meus pais, Seize e Sakae, que sem o seu amor e apoio nada se concretizaria.

Para meus pais, Seize e Sakae.

RESUMO

Este trabalho apresenta a implementação do protocolo de apresentação do modelo OSI/ISO para o projeto SISDI-MAP (Sistema Didático MAP). Utilizou-se o ambiente EPOS ("Engineering Project Oriented System") como suporte de desenvolvimento principalmente para a especificação formal da implementação e geração automática de código. Apresentam-se os conceitos referentes à camada de apresentação, o modelo de implementação, alguns aspectos da implementação e da geração automática de código e, finalmente, um exemplo de execução deste protocolo.

ABSTRACT

This work presents the implementation of the presentation protocol of OSI/ISO models for the SISDI-MAP (Didactic System MAP) project. The EPOS (Engineering Project Oriented System) environment has been used as a development support, mainly in the formal specification of the implementation and in the automatic code generation. The presentation layer concepts, the implementation model, and some aspects of the implementation and automatic code generation are presented, as well as an example of the execution of the protocol.

ÍNDICE

Índice i
Lista de Figuras iv
Lista de Tabelas vi
Terminologias e Abreviaturas vii

CAPÍTULO I - Introdução
1.0. - Introdução 1.1
1.1. - Objetivos do Trabalho 1.2
1.2. - O Projeto SISDI-MAP 1.3
1.3. - O Ambiente EPOS 1.6
1.4. - Desenvolvimento do Trabalho 1.10

CAPÍTULO II - A Camada de Apresentação
2.0. - Introdução 2.1
2.1. - A Camada de Apresentação 2.2
2.1.1. - Propósito da Camada de Apresentação 2.3
2.1.2. - Unidades Funcionais 2.6
2.2. - Especificação de Serviços 2.8
2.2.1. - Estabelecimento de Conexão 2.9
2.2.2. - Término de Conexão 2.11
2.2.3. - Gerenciamento de Contexto 2.12
2.2.4. - Transferência de Informação 2.13
2.2.5. - Controle de Diálogo 2.14
2.3. - Especificação de Protocolos 2.15
2.3.1. - Estabelecimento de Conexão 2.16
2.3.2. - Término de Conexão 2.18
2.3.3. - Gerenciamento de Contexto 2.20
2.3.4. - Transferência de Informação 2.20
2.3.5. - Controle de Diálogo 2.21
2.4. - Tabelas de Estado 2.22
2.5. - Escopo de Implementação 2.26
2.6. - Especificação dos Requisitos 2.28
2.6.1. - Descrição do Problema 2.28
2.6.2. - Projeto Conceitual 2.31

CAPÍTULO III - Implementação da Camada de Apresentação

3.0. - Introdução 3.1
3.1. - O Ambiente Multitarefa 3.2
3.2. - O Modelo de Implementação 3.3
3.2.1. - Portas de Comunicação 3.4
3.2.2. - O Mecanismo de Envio de Mensagens 3.5
3.2.3. - O Mecanismo de Recepção de Mensagens 3.5
3.2.4. - Temporizadores 3.6
3.2.5. - Bloco de Mensagens de Protocolo 3.6
3.2.6. - Estruturas de Dados 3.9
3.2.7. - Parâmetros de Inicialização 3.9
3.3. - Especificação da Implementação 3.10
3.3.1. - O Módulo Principal 3.10
3.3.2. - O Processo de Apresentação 3.13
3.3.3. - Especificação das Portas de Comunicação 3.18
3.3.4. - Tratamento das Mensagens do ACSE 3.19
3.3.5. - Tratamento das Mensagens do MMS 3.23
3.3.6. - Tratamento das Mensagens da Sessão 3.25
3.3.7. - Tratamento de Erros 3.26
3.3.8. - Ações Terminais 3.27
3.4. - A Geração de Código 3.27
3.4.1. - O Processo Proc_Presentation 3.29
3.4.2. - Procedimentos 3.31
3.3. - Documentação Gráfica 3.33

CAPÍTULO IV - Exemplo de Execução

4.0. - Introdução 4.1
4.1. - Estabelecimento de Conexão 4.1
4.2. - Primitiva "P_CONNECT request" 4.2
4.3. - PPDU CP ("Connect Presentation") 4.4
4.4. - Primitiva "S_CONNECT indication" 4.4
4.5. - Primitiva "P_CONNECT response" 4.5
4.6. - PPDU CPA ("Connect Presentation Accept") 4.6
4.7. - PPDU CPR ("Connect Presentation Reject") 4.6
4.8. - Primitiva "S_CONNECT confirm" 4.7

CAPÍTULO V - Conclusões

5.0. - A implementação 5.1

5.1. - O Ambiente EPOS 5.2

5.1.1. - A Linguagem EPOS-R 5.2

5.1.2. - A Linguagem EPOS-S 5.3

5.1.3. - O Gerador de Código C-Composer 5.4

5.2. - O Ambiente Multitarefa 5.5

5.3. - O Futuro 5.5

CAPÍTULO VI - Referências Bibliográficas

6.0 - Referências Bibliográficas..... 6.1

ANEXO A - A Linguagem de Especificação EPOS-R

ANEXO B - A Linguagem de Especificação EPOS-S

ANEXO C - O Gerador de Código C-Composer

LISTA DE FIGURAS

Figura 1.0. - Arquitetura Funcional do SISDI-MAP 1.5
Figura 1.1. - Ambiente EPOS e seus Componentes 1.7
Figura 2.0. - Modelo Usuário-Provedor de Serviços de Apresentação 2.2
Figura 2.1. - Escopo das Sintaxes Abstrata e de Transferência 2.4
Figura 2.2. - Diagrama Temporal dos Seviços Confirmados ... 2.8
Figura 2.3. - Diagrama Temporal do Seviço P_P_ABORT 2.9
Figura 2.4. - Modelo da Camada de Apresentação 2.15
Figura 2.5. - Estabelecimento de Conexão 2.17
Figura 2.6. - Liberação de Conexão pelo Provedor 2.19
Figura 2.7. - A Camada de Apresentação no SISDI-MAP 2.27
Figura 2.8. - Processo de Decisão do Serviço P_CONNECT 2.34
Figura 3.0. - Modelo de Implementação da Camada de Apresentação 3.3
Figura 3.1. - Bloco de Mensagens de Protocolo 3.7
Figura 3.2. - Estrutura de Dados da Apresentação 3.8
Figura 3.3. - Estrutura de Dados PRES_TAB 3.9
Figura 3.4. - Módulo Principal do Processo de Apresentação 3.11
Figura 3.5. - Especificação da Inclusão do Arquivo "MASTER_P.INC" 3.12
Figura 3.6. - Diagrama de Blocos do Processo de Apresentação 3.13
Figura 3.7. - Portas de Entrada do Processo de Apresentação 3.14
Figura 3.8. - Especificação do Processo de Apresentação ... 3.15
Figura 3.9. - Condição de Existência de Mensagens do ACSE . 3.17
Figura 3.10. - Porta de Comunicação de Entrada do ACSE 3.18
Figura 3.11. - Especificação do Procedimento TREAT_ACSE 3.19
Figura 3.12. - Especificação do Procedimento TREAT_ACSE_REQUEST 3.21

Figura 3.13. - Especificação do Procedimento TREAT_P_CONNECTION_REQUEST 3.21
Figura 3.14. - Especificação do Procedimento TREAT_P_DATA_REQUEST 3.23
Figura 3.15. - Especificação do Procedimento TREAT_SES 3.26
Figura 3.16. - Especificação do Procedimento TREAT_INVALID_INTERSECTIONS_PROVIDER 3.28
Figura 3.17. - Especificação do Procedimento DECLARE_VARIABLES_LOCAIS 3.29
Figura 3.18. - O Processo de Apresentação 3.30
Figura 3.19. - Código do Procedimento TREAT_ACSE 3.31
Figura 3.20. - Código do Procedimento TREAT_INVALID_INTERSECTIONS_PROVIDER 3.32
Figura 3.21. - Diagrama Hierárquico Principal 3.34
Figura 3.22. - Rede de Petri do Módulo de Apresentação 3.35
Figura 3.23. - Diagrama Hierárquico do Processo de Apresentação 3.36
Figura 3.24. - Rede de Petri do Processo de Apresentação ... 3.37
Figura 3.25. - Fluxograma do Processo de Apresentação 3.38
Figura 3.26. - Diagrama Hierárquico do Procedimento TREAT_ACSE 3.39
Figura 3.27. - Rede de Petri do Procedimento TREAT_ACSE 3.40
Figura 3.28. - Fluxograma do Procedimento TREAT_ACSE 3.41
Figura 4.0. - Estabelecimento de uma Conexão de Apresentação 4.2
Figura 4.1. - Bloco de Mensagens de Protocolo para a Primitiva P_CONNECT request 4.3

LISTA DE TABELAS

Tabela 2.0. - Grupo de Serviço para o Estabelecimento de Conexão 2.9
Tabela 2.1. - Parâmetros do Serviço P_CONNECT 2.10
Tabela 2.2. - Grupo de Serviços para o Término de Conexão . 2.11
Tabela 2.3. - Grupo para o Gerenciamento de Contexto 2.12
Tabela 2.4. - Parâmetros do Serviço P_ALTER_CONTEXT 2.13
Tabela 2.5. - Grupo de Serviços para Transferência de Informação 2.14
Tabela 2.6. - Grupo de Serviços para o Controle de Diálogo 2.15
Tabela 2.7. - PPDUs da Camada de Apresentação 2.16
Tabela 2.8. - Mapemento dos Parâmetros da PPDU CP sobre os Parâmetros do Serviço S_CONNECT 2.18
Tabela 2.9. - Serviços de Apresentação Mapeados na Sessão . 2.22
Tabela 2.10. - Estados da PPM 2.24
Tabela 2.11. - Tabela de Estado do Estabelecimento de uma Conexão 2.25
Tabela 2.11.a. - Tabela de Decisão (1/2) do Serviço P_CONNECT 2.36
Tabela 2.11.b. - Tabela de Decisão (2/2) do Serviço P_CONNECT 2.37

TERMINOLOGIAS E ABREVIATURAS

ACSE - Association Control Service Element
ANS.1 - Abstract Syntax Notation One
AP - Application Process
API - Application Process Interface
BER - Basic Encoding Rules
CEP - Connection End Point
EPOS - Enginnering Project Oriented System
EPOS-A - EPOS Analysis
EPOS-C - EPOS Communication
EPOS-D - EPOS Documentation
EPOS-M - EPOS Management
EPOS-P - EPOS Project
EPOS-R - EPOS Requirements
EPOS-S - EPOS Specification
ISO - International Standards Organization
MAP - Manufacturing Automation Protocol
MMS - Manufacturing Message Specification
OSI - Open System Interconnection
OSIE - OSI Environment
PCI - Protocol Control Information
PDU - Protocol Data Unit
PPDU - Presentation PDU
PPM - Presentation Protocol Machine
PSAP - Presentation SAP
SAP - Servive Access Point
SDU - Service Data Unit
TOP - Technical Office Protocol

CAPÍTULO I - INTRODUÇÃO

1.0. - INTRODUÇÃO

A necessidade de interconexão de computadores heterogêneos com o objetivo de agilizar o fluxo de informações e utilizar recursos disponíveis remotamente levou à necessidade de definição de padrões de comunicação como único meio de viabilizar esta interconexão.

A base para a definição destes padrões, aceitos mundialmente, consiste no Modelo de Referência OSI da ISO de 7 camadas /ISO 7498, 1982/. O trabalho sobre o Modelo de Referência foi iniciado com o objetivo de proporcionar um ambiente padrão para a comunicação entre porções distribuídas de uma aplicação, isto é, entre processos de aplicação distribuídos que cooperam entre si para desempenhar o processamento das informações requeridas por uma aplicação.

Com o objetivo de simplificar o processo de padronização, algumas aplicações específicas definiram conjuntos de padrões mais adequados às suas particularidades, mas sempre mantendo a filosofia do modelo OSI/ISO como referência. Uma das áreas de aplicação que definiu um destes conjuntos é a automação industrial através do padrão MAP / TOP ("*Manufacturing Automation Protocol / Technical Office Protocol*").

Atualmente encontra-se disponível a versão MAP/TOP 3.0 /GM-MAP, 1987/, congelada para os próximos anos, e que é seguida neste trabalho. A versão MAP 3.0 é um documento de especificação dos serviços e protocolos do padrão MAP e que utiliza, basicamente, linguagem natural e diagramas de estado na especificação dos serviços e protocolos.

1.1. - OBJETIVOS DO TRABALHO

Este trabalho concentra-se na implementação da camada de apresentação, que corresponde à camada de número 6 do modelo de referência OSI da ISO. Esta implementação visa os serviços de apresentação disponíveis na versão MAP 3.0.

A camada de apresentação é responsável pela representação dos dados trocados entre entidades de aplicação, preservando seu significado para as mesmas, uma vez que, para que processos de aplicação sejam capazes de se comunicarem entre si deve existir um acordo sobre a semântica de todos os aspectos relacionados com a troca de informações.

A fase inicial consistiu na elaboração de um projeto lógico funcional e detalhado, a partir das especificações de serviços e protocolos, obtendo-se como produto final uma descrição do software em pseudo-código, e que, se possível, proporcionasse a geração automática de código na fase de implementação.

A segunda etapa consistiu na implementação dos serviços em uma máquina compatível com a linha IBM/PC e sua correspondente integração a um Sistema Didático, denominado SISDI-MAP. Procurou-se realizar uma implementação modular e expansível aos serviços não implementados.

Para proporcionar o suporte adequado a todas as fases do desenvolvimento deste trabalho utilizou-se um ambiente denominado EPOS ("*Engineering Project Oriented System*"), voltado para o desenvolvimento de software/hardware para Projetos de Tempo Real.

A realização deste trabalho faz parte de um conjunto de duas outras propostas semelhantes correspondentes à camada de sessão (camada 5) /Mendez, 1990/ e ao gerenciamento de rede (camada de aplicação) /Barros, 1990/. O trabalho também permite uma análise da utilidade de um sistema do tipo EPOS na implementação deste

protocolo, e que pode ser variável de camada para camada, fazendo com que o desenvolvimento dos três trabalhos em paralelo, permita uma análise adequada de experiências. Além disso alcançar-se-á uma documentação homogênea que poderá servir, futuramente, como uma base de implementação de protocolos MAP nos mais variados tipos de máquinas.

1.2. - O PROJETO SISDI-MAP

O projeto SISDI-MAP (SIStema DIdático MAP) /Paglioli et al., 1989; Jacinto, 1990/ foi concebido para fins didáticos de modo a permitir ao usuário uma visualização da execução de protocolos que compõem aplicações típicas de um ambiente de manufatura.

O SISDI-MAP é composto por software básico e software aplicativo. O software básico /Zabeu, 1989/ é dedicado e customizado, provendo um núcleo de tempo real de modo a se obter um ambiente multitarefa em equipamentos compatíveis com a linha IBM-PC (MS-DOS). Foi desenvolvido, em sua grande parte, em linguagem C e é utilizado pelo software aplicativo na forma de uma biblioteca de funções.

Na versão atual do SISDI-MAP, o software aplicativo possui a seguinte arquitetura funcional, como ilustra a Figura 1.0 :

- Interface de Operação de Usuário;
- Programas Aplicativos (APs);
- Interface de Aplicação (API);
- Protocolo MMS;
- Protocolo ACSE;
- Protocolo de Apresentação;
- Protocolo de Sessão;
- Simulador da Camada de Transporte e Inferiores.

A Interface de Operação de Usuário destina-se a fornecer uma interface de operação "amigável" (janelas, "*menus*", gráficos) para o operador do sistema, simplificando seu acesso aos serviços

oferecidos pelo sistema, além de fornecer informações sobre a execução de cada serviço solicitado pelo próprio usuário ou pelo AP ("*Application Process*"). O usuário pode então executar uma sequência de comandos MMS ("*Manufacturing Message Specification*") de modo interativo, interfaceando diretamente com a API ("*Application Process Interface*"), ou de modo automático, transferindo o controle da execução para o AP, como ilustra a Figura 1.0. Opcionalmente, permite a introdução de erros de comunicação entre protocolos fim-a-fim (nível do simulador da camada de transporte) objetivando testes de situações de erro.

Os Programas Aplicativos (AP) representam programas reais com a função de solicitar os serviços oferecidos pela rede através da API, como por exemplo uma célula flexível de manufatura. A Interface de Aplicação (API) provê uma maior portabilidade ao sistema visto que define uma biblioteca de funções padronizadas que é utilizada pelos APs. Esta biblioteca facilita a programação dos APs, pois realiza tarefas que a princípio seriam de sua responsabilidade, além de proporcionar uma interface "amigável" para o MMS.

O MMS, o ACSE, a Apresentação e a Sessão são responsáveis pela execução das funções dos provedores MMS /ISO-DP 9506, 1987/, ACSE ("*Association Control Service Element*") /ISO-IS 8649, 1988/, Apresentação /ISO-DIS 8823, 1986/ e Sessão /ISO-DP 8326, 1983/, respectivamente. Estas funções estão relacionadas com as máquinas de estado definidas em cada protocolo.

O 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. O protocolo ACSE fornece as facilidades básicas para o controle de uma associação (estabelecimento e liberação) entre duas entidades de aplicação que se comunicam por meio de uma conexão de apresentação.

O protocolo de apresentação fornece serviços que podem ser

selecionados pela camada de aplicação para a interpretação da sintaxe dos dados trocados, resolvendo os eventuais problemas de diferença de sintaxe entre os sistemas abertos comunicantes. O protocolo de sessão organiza e sincroniza o diálogo, bem como gerencia a troca de dados entre entidades da camada de apresentação comunicantes.

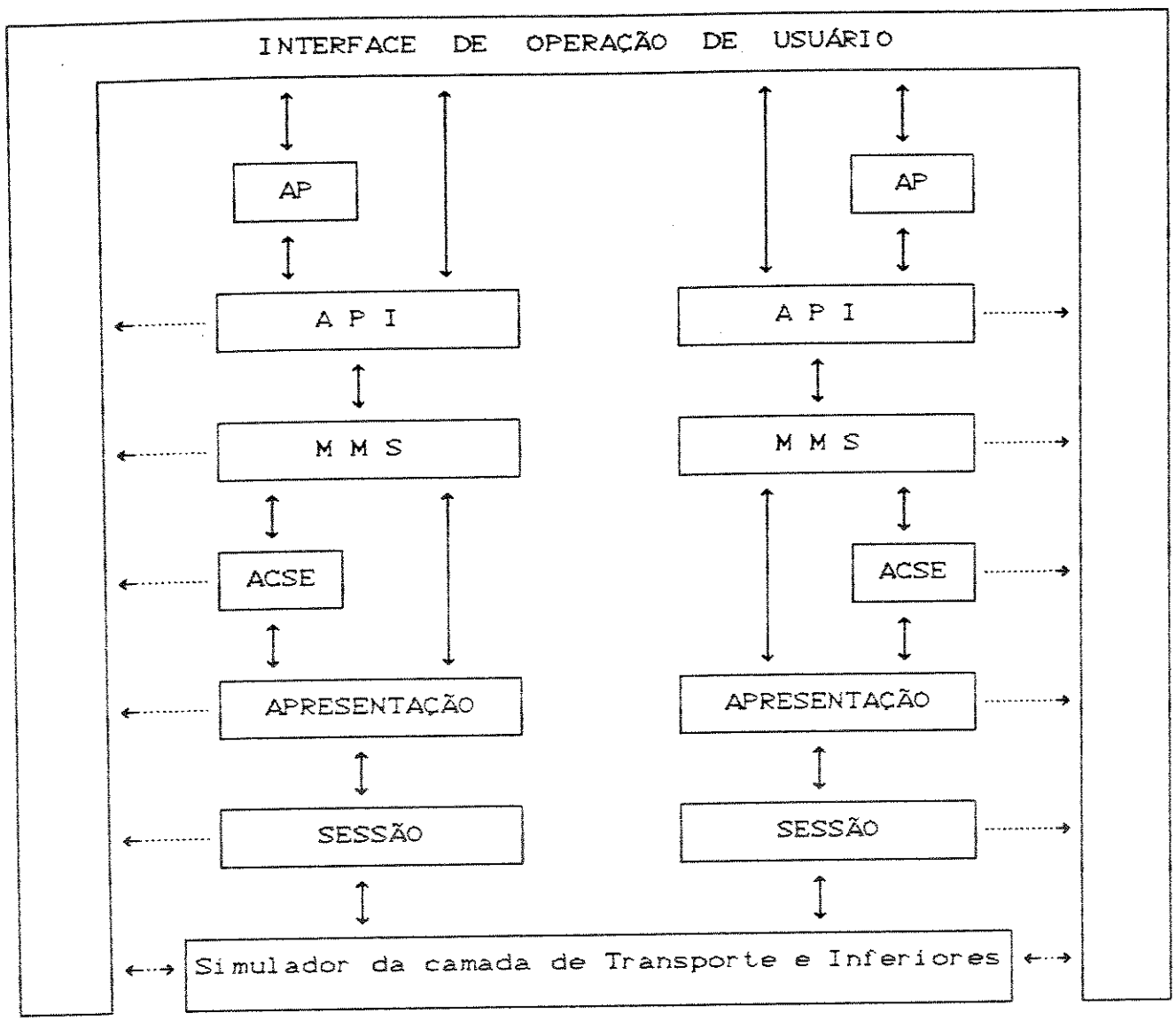


Figura 1.0. - Arquitetura Funcional do SISDI-MAP

O simulador da camada de transporte é responsável pela comunicação

fim-a-fim entre os protocolos deste nível, através da simulação das primitivas de serviços da camada de transporte e demais camadas inferiores do modelo OSI da ISO. Para tornar a comunicação do Sistema Didático mais próxima de um ambiente real é permitida a introdução de erros, via a interface de operação de usuário, no conteúdo ou no formato das PDUs ("*Protocol Data Unit*") transportadas pelas primitivas de serviços da camada de transporte. Este recurso permite que os protocolos MMS, ACSE, Sessão e Apresentação possam ser testados com relação às suas funções de controle de erros definidas em suas especificações.

1.3. - O AMBIENTE EPOS

O EPOS /Lauber et al., 1986; Biewald et al., 1980/ é um ambiente integrado de engenharia de software para o suporte no desenvolvimento e gerenciamento de projetos de software e hardware. Foi desenvolvido no "*Institute for Control Engineering and Process Automation of the University of Stuttgart, West Germany*".

O sistema EPOS (Figura 1.1) tem como principais objetivos : a formulação de uma descrição consistente, completa e sem ambiguidades do problema e de seus requisitos; o desenvolvimento de um software bem estruturado, livre de erros e completamente documentado, particularmente para sistemas de tempo real; tradução (automática) do projeto de software para uma linguagem de programação arbitrária; permitir a especificação do "*layout*" do hardware do sistema; auxiliar no planejamento, gerenciamento e controle do projeto e, por fim, proporcionar uma maior qualidade do software, uma vez que impõe procedimentos de desenvolvimento estruturados e sistemáticos.

Para atingir estes objetivos os seguintes princípios são adotados pelo sistema EPOS : suporte computacional integrado para todas as áreas de atividades de um projeto (desenvolvimento, gerenciamento do projeto e da configuração, garantia de qualidade); suporte

computacional consistente para todas as fases de um projeto (desde a especificação até a implementação / manutenção, abrangendo todo o ciclo de vida de um projeto); suporte computacional não somente para software, mas para o sistema software / hardware; suporte computacional para vários métodos de projeto; suporte gráfico extensivo; e suporte para o desenvolvimento do projeto em equipes.

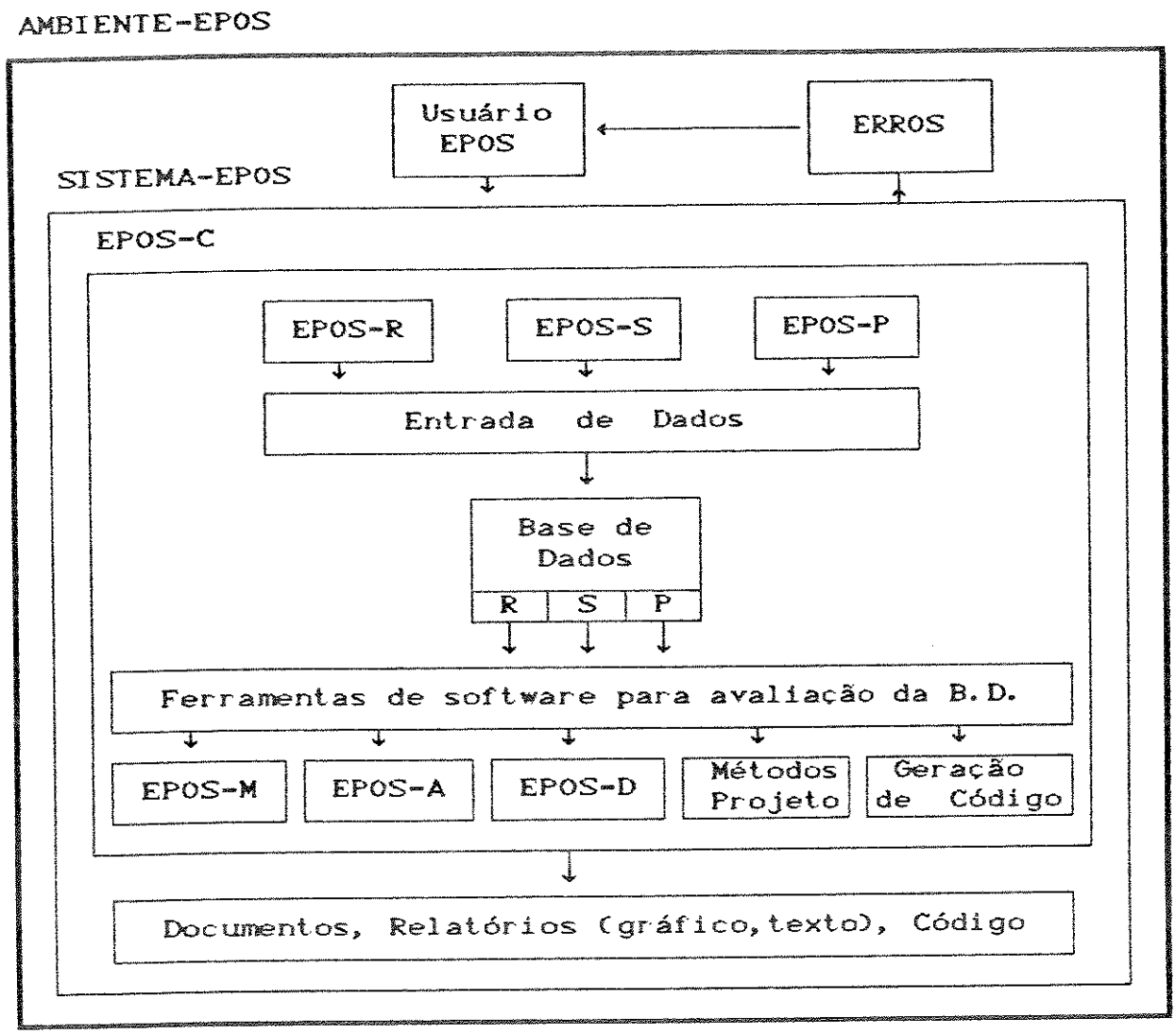


Figura 1.1. - Ambiente EPOS e seus Componentes

O sistema EPOS é constituído pelos seguintes componentes :

- Três linguagens de especificação :
  - Linguagem para descrição dos requisitos (EPOS-R);
  - Linguagem para descrição do software e hardware (EPOS-S);

- Linguagem para gerenciamento do projeto, administração do produto e garantia de qualidade (EPOS-P);

- Uma base de dados do projeto;

- Cinco ferramentas de software para avaliação da base de dados :
  - Ferramenta para suporte nas atividades relacionadas ao gerenciamento da configuração e do projeto (EPOS-M);
  - Ferramenta para análise da informação contida na base de dados (EPOS-A);
  - Ferramenta para a geração de documentos e relatórios (textos, gráficos, listas) (EPOS-D);
  - Ferramenta para suporte nos vários métodos de projeto ("*Design Method*");
  - Ferramenta para tradução do projeto de software para uma determinada linguagem de programação ("*Code Generation*").

- Um programa de controle do usuário, que proporciona a comunicação entre o usuário EPOS e o sistema EPOS (EPOS-C).

A seguir apresentamos uma breve descrição das principais ferramentas de software disponíveis neste ambiente EPOS.

EPOS-R : É uma linguagem de especificação semi-formal utilizada nas fases iniciais do projeto, no caso, durante a fase de descrição do problema (especificação das necessidades do usuário e dos requisitos do sistema) e do modelamento da solução conceitual do problema. Uma descrição mais detalhada sobre seus elementos encontra-se no anexo A.

EPOS-S : É uma linguagem de especificação com sintaxe formal e semântica definida para descrever os diferentes níveis do projeto do sistema. Os elementos básicos desta linguagem são denominados "objetos de projeto". Uma descrição mais detalhada sobre seus elementos encontra-se no anexo B.

EPOS-P : É uma linguagem de especificação formal utilizada para a descrição de informação relacionada ao gerenciamento do projeto, ao gerenciamento da configuração e à garantia da qualidade do produto. De maneira semelhante ao EPOS-S, é estruturada em "objetos de gerenciamento".

EPOS-M : É uma ferramenta para o suporte nas atividades relacionadas ao projeto e ao gerenciamento de configuração, assim como à administração do produto (Ex. : administração de versões/variações, gerenciamento de configuração).

EPOS-A : É uma ferramenta utilizada para analisar e testar as informações descritas pelas linguagens de especificação EPOS-R, EPOS-S e EPOS-P. Esta análise permite a detecção prematura de erros de especificação e de projeto.

EPOS-D : É uma ferramenta para a geração de documentação e possibilita os seguintes tipos de documentação : textos, tabelas de decisão, gráficos, diagramas, e listas. Tais documentações são baseadas nas descrições elaboradas nas linguagens de especificação EPOS-R, EPOS-S e EPOS-P.

Suporte nos Métodos de Projeto : O EPOS proporciona ao seu usuário vários métodos de projeto, ficando a cargo do projetista selecionar o método mais apropriado para o problema a ser solucionado. Os métodos de projeto disponíveis são :

- Orientado a dispositivos,
- Orientado a fluxo de dados,
- Orientado a módulos,
- Orientado a eventos e
- Orientado a função.

Para suportar estes diferentes métodos de projeto o ambiente EPOS fornece ferramentas para : implementação do método de projeto desenvolvido, com verificação consistente dependente do método e respectiva documentação; análise métrica dependente do método de

projeto; e transição entre diferentes métodos de projeto.

Geração de Código : O sistema EPOS possui ferramentas para a geração de código que suportam a tradução do projeto de software para a implementação em uma linguagem de programação selecionada; isto é, a partir da especificação do projeto realizada em EPOS-S, o sistema EPOS gera automaticamente o arquivo com o programa fonte na linguagem de programação escolhida.

Para algumas linguagens de programação orientadas para aplicações de alto nível (Ex. : C/Atlas), o código do programa pode ser gerado automaticamente a partir das especificações em EPOS-S. Para certas linguagens de programação (PASCAL, FORTRAN, ADA, PEARL) uma grande parte do código fonte pode ser gerada automaticamente com o auxílio do gerador de programa apropriado. Para as demais linguagens de programação, a geração de código fica disponível por meios da "seleção de código", onde nos próprios objetos de projeto do EPOS-S, partes de códigos são inseridas manualmente pelo projetista.

EPOS-C : Programa para suportar a comunicação entre o usuário do EPOS e o sistema EPOS.

1.4. - DESENVOLVIMENTO DO TRABALHO

A apresentação deste trabalho é realizada em cinco capítulos, sendo o primeiro esta introdução. Numa primeira etapa apresentam-se os conceitos e as definições associados ao trabalho proposto, e numa segunda etapa, a descrição da implementação realizada.

No capítulo 2, denominado "A Camada de Apresentação", introduzem-se os conceitos sobre a camada de apresentação. Descrevem-se as funções da camada, seu relacionamento dentro do modelo de referência OSI/ISO e todos os serviços e protocolos oferecidos pela camada. Além disso, faz-se uma breve descrição das

tabelas de estados, que também são utilizadas para uma descrição auxiliar e mais formal do protocolo, e da especificação do problema e da solução conceitual realizada na linguagem EPOS-R.

No capítulo 3, denominado "A Implementação da Camada de Apresentação", descreve-se o modelo de implementação e a sua respectiva especificação, que foi efetuada na linguagem EPOS-S. Adicionalmente apresentam-se alguns aspectos da geração automática de código, que foi efetuada pelo gerador C-Composer a partir da especificação EPOS-S, e a documentação gráfica gerada pelo sistema EPOS.

No capítulo 4, denominado "Exemplo de Execução", apresenta-se um exemplo de execução do protocolo da camada de apresentação, no caso, o protocolo para o estabelecimento de uma conexão de apresentação.

Finalmente, no capítulo 5, denominado "Conclusões", discutem-se aspectos da implementação e sobre a utilização do ambiente EPOS como suporte de desenvolvimento, e apresentam-se as conclusões obtidas sobre o trabalho realizado.

CAPÍTULO II - A CAMADA DE APRESENTAÇÃO

2.0. - INTRODUÇÃO

Na primeira parte deste capítulo apresentam-se uma exposição geral do propósito da camada de apresentação, alguns conceitos básicos referentes às sintaxes envolvidas na camada, e as suas unidades funcionais. Posteriormente apresenta-se uma especificação dos serviços e protocolos de apresentação, descrevendo os serviços de comunicação fornecidos pela camada através da definição das primitivas de serviços e das interações com as primitivas de sessão. Adicionalmente, descrevem-se brevemente os procedimentos do protocolo em tabelas de estado nos documentos da ISO.

Na segunda parte, apresenta-se um resumo de como foi realizada a especificação dos requisitos da implementação desta camada utilizando a linguagem de especificação EPOS-R. Esta especificação é composta por uma descrição do problema, que foi realizada em linguagem natural, e pela apresentação de uma solução conceitual para o mesmo. Esta solução conceitual foi feita tomando-se como base as tabelas de estado dos serviços de apresentação e foram mapeadas nos elementos formais conhecidos como processos de decisão do EPOS.

2.1. - A Camada de Apresentação

A camada de apresentação /Giozza et al., 1986; Halsall, 1986; Mendes, 1989; Rose, 1988; Tanenbaum, 1987/ corresponde a camada de número 6 do modelo de referência OSI da ISO /ISO 7498, 1986/. Esta camada presta serviços à camada adjacente superior (camada de Aplicação) e, para tal, faz uso dos serviços da camada adjacente inferior, no caso a camada de Sessão. O modelo usuário-provedor de apresentação consiste de dois usuários de serviços de apresentação (usuário local e remoto) e do próprio provedor de serviços de apresentação. Os usuários se comunicam com o provedor de serviços de apresentação através de dois PSAPs ("*Presentation Service Access Point*"), como ilustra a Figura 2.0.

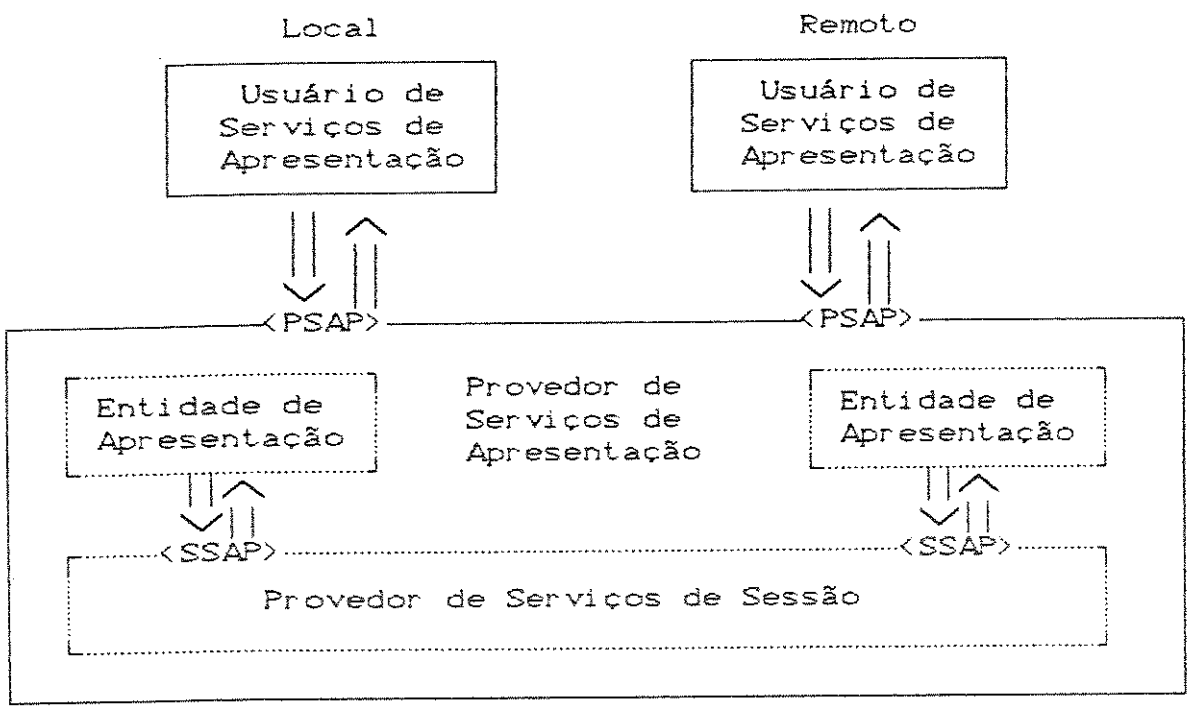


Fig. 2.0. - Modelo Usuário-Provedor de Serviços de Apresentação

Cabe aqui ressaltar a principal diferença, em termos funcionais, entre a camada de Aplicação e a camada de Apresentação : A camada de Aplicação trata principalmente da semântica dos dados, isto é, do significado dos dados, ao passo que a camada de Apresentação manipula principalmente com a sintaxe dos dados, isto é, da

representação dos mesmos referente à sua parte sintática.

2.1.1. - Propósito da Camada de Apresentação

Uma Aplicação é o conjunto de requisitos de processamento das informações do usuário /Bartioli, 1983/. Um Processo de Aplicação (AP) é um elemento lógico dentro de um sistema que desempenha o processamento das informações requeridas para uma aplicação específica. Quando uma aplicação é distribuída, cada porção distribuída da aplicação é um processo de aplicação.

Assim, o projeto de um ambiente de interconexão de sistemas abertos implica que as entidades de aplicação envolvidas, visando proporcionar serviços de informações distribuída, podem estar residentes em diferentes sistemas de computadores. Os APs, apesar de estarem num ambiente distribuído, devem trabalhar de forma ordenada, interagindo na execução de tarefas e, para tal, trocando mensagens entre si.

Apesar do valor de um dado armazenado em cada sistema ser do mesmo tipo (abstrato), por exemplo, inteiro, caracter, etc., tais dados são definidos independentemente do meio onde serão executados; consequentemente seus valores poderão ter representações locais distintas (por exemplo : em 8 bits, 16 bits, etc.). Assim, programas e dados existirão localmente em um computador representados segundo suas convenções próprias, de acordo com a sua sintaxe local, e no caso de sistemas heterogêneos é frequente que estas sintaxes locais sejam diferentes, de acordo com suas regras locais de codificação.

Neste contexto, quando APs de sistemas heterogêneos desejarem se comunicar com o objetivo de trocarem dados, é essencial que as mensagens trocadas entre si tenham um significado comum (semântica compartilhada). Para que isto seja possível, todas as mensagens trocadas entre duas entidades de aplicação estão em uma sintaxe abstrata de comum acordo, e como a forma de representação dos

tipos de dados pode ser diferente em dois sistemas, principalmente quando se fala em sistemas heterogêneos, é negociada uma sintaxe de transferência que é utilizada para transferir os valores dos dados entre os dois sistemas, de modo a proporcionar uma comunicação harmoniosa e independente das sintaxes locais.

Note-se que não se faz necessário que os APs se acertem na mesma sintaxe local. É importante, contudo, que estejam de acordo com a sintaxe concreta comum de transferência de dados, chamada de sintaxe de transferência.

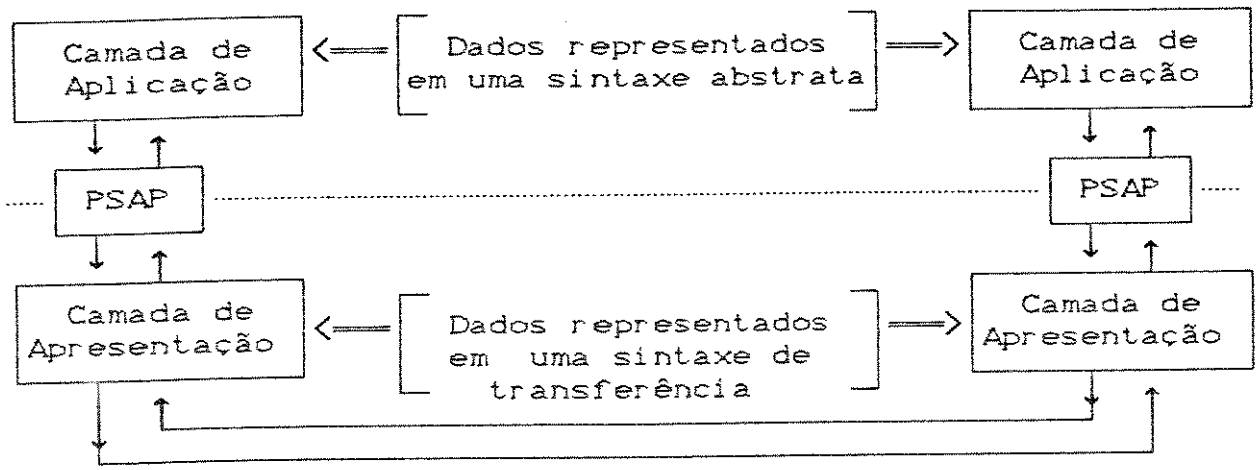


Figura 2.1. - Escopo das Sintaxes Abstrata e de Transferência

Neste ponto podemos observar uma das funções típicas da camada de apresentação : providenciar mecanismos que permitam aos processos de aplicação definir e selecionar sintaxes de transferência através da negociação entre um conjunto de sintaxes de transferência que cada sistema suporte, escolhendo aquela mais adequada (e eficiente) para a aplicação envolvida (Por exemplo, negociar sintaxes mais adequadas para a transferência de arquivos gráficos, arquivos textos, arquivos numéricos, etc.).

Durante a fase de negociação de sintaxes de transferência, quando ocorre uma associação de uma sintaxe abstrata com uma sintaxe de transferência diz-se que se constituiu um **Contexto de Apresentação**. Esta associação nada mais é do que a determinação de

uma sintaxe de transferência que suporte a sintaxe abstrata do usuário e a atribuição de um identificador a esta associação.

Como já foi dito, a diversidade de tipos de dados trocados entre os usuários de serviços de apresentação é muito grande, consequentemente é possível a existência de diversos contextos de apresentação, e que formam o **Conjunto de Contextos de Apresentação**.

Entre estes contextos de apresentação, existe um contexto especial, denominado *Contexto Default*. Trata-se de um contexto de apresentação que sempre é conhecido pelo provedor de apresentação e onde é adotada uma sintaxe de transferência pré-estabelecida. Este contexto *default* é utilizado quando o conjunto de contextos de apresentação é vazio, isto é, quando não houve sucesso na negociação de sintaxes de transferência e em determinadas primitivas de serviços de apresentação que sempre transmitem dados neste contexto, independentemente da existência ou não de outros contextos de apresentação.

Dentro do OSIE ("*OSI Environment*") a sintaxe local origem e a sintaxe local remota somente são reconhecidas dentro de seus próprios sistemas. Do ponto de vista da fronteira entre a Camada de Apresentação e a Camada de Sessão e das demais camadas inferiores, a única sintaxe vista é a sintaxe de transferência, como ilustra a Figura 2.1.

Nota-se então que a informação a ser transferida entre entidades de aplicação possui três versões sintáticas : a sintaxe local origem, a sintaxe local destino e a sintaxe de transferência. Esta sintaxe de transferência pode ser uma das sintaxes locais, uma sintaxe totalmente diferente, sem nenhuma relação com ambas as sintaxes locais, bem como serem as três idênticas.

Assim, outra função atribuída à camada de apresentação é a transformação (codificação) dos dados do usuário local, que estão expressos em sintaxe abstrata, para a sintaxe de transferência

escolhida. Uma vez feita esta transformação da sintaxe abstrata para a sintaxe de transferência, no usuário remoto, faz-se necessário executar o processo inverso, isto é, transformação (decodificação) dos dados, agora expressos na sintaxe de transferência, para a sintaxe abstrata que o usuário remoto entenda.

Além destas três funções principais, uma quarta função corresponde ao mapeamento de diversas requisições de serviços feitas pelas entidades de aplicação nas correspondentes primitivas de serviços da camada de sessão.

2.1.2. - UNIDADES FUNCIONAIS

Existem na camada de apresentação uma série de Unidades Funcionais. Estas unidades funcionais correspondem a agrupamentos lógicos de serviços e podem ser classificados em duas grandes categorias :

- Unidade Funcional de Sessão e
- Unidade Funcional de Apresentação.

A Unidade Funcional de Sessão engloba os serviços que são mapeados diretamente na camada de sessão. Consequentemente, a seleção desta unidade funcional está sujeita às restrições impostas pelos serviços de sessão. A Unidade Funcional de Apresentação agrega os serviços proporcionados pela camada de apresentação propriamente ditos, e pode ser subdividida em três novas categorias :

- Unidade Funcional "*kernel*",
- Unidade Funcional de Gerenciamento de Contexto, e
- Unidade Funcional de Restauração de Contexto.

A Unidade Funcional "*kernel*" compreende os serviços básicos da camada de apresentação. Os serviços que fazem parte desta unidade funcional estão sempre disponíveis (não negociáveis) e

proporcionam facilidades para os serviços orientados à conexão. Nos serviços com conexão a comunicação ocorre em três etapas : estabelecimento de uma conexão, transferência de dados e liberação de uma conexão.

A Unidade Funcional de Gerenciamento de Contexto é opcional e a sua utilização é negociável. Se estiver selecionada, o conjunto de contextos de apresentação pode ser alterado durante a conexão. Esta alteração pode ser uma deleção de um contexto já definido e disponível para uso, bem como a definição de um novo contexto e a sua respectiva inserção no conjunto de contextos de apresentação corrente.

A Unidade Funcional de Restauração de Contexto também é opcional e a sua utilização é negociável. Se estiver selecionada, o serviço de apresentação pode "lembrar" e restaurar o conjunto de contexto definido em determinados pontos durante uma conexão. Assim, se o usuário de serviços requisitar o retorno para um destes pontos, o conjunto de contexto definido nestes pontos ativos pode ser restaurado. Tais pontos ativos são : ponto de sicronização maior, ponto de sincronização menor, ponto de interrupção de atividade e ponto de início de sincronização.

A Unidade Funcional de Restauração de Contexto não deve ser selecionada se a unidade funcional de gerenciamento de contexto também não estiver selecionada na mesma conexão de apresentação. Isto se deve ao fato de que a operação de restaurar um contexto consiste em deletar contextos e/ou definir novos contextos, de modo que o conjunto de contexto de apresentação definido antes do pedido de restauração seja modificado para o conjunto de contexto de apresentação definido no ponto ativo, e estas caracteristicas de deletar/definir contextos de apresentação são funções dos serviços pertencentes à unidade funcional de gerenciamento de contexto.

2.2. - Especificação de Serviços

O modelo da camada de apresentação proposto pela ISO é um modelo orientado a serviços /Hollis, 1983/. Neste modelo, a camada de apresentação recebe solicitações de seus serviços provenientes da camada de aplicação e solicita serviços da camada de sessão.

Todas as primitivas de serviços definidas para a OSI são utilizadas pela camada de apresentação. São elas : *request*, *indication*, *response* e *confirm*. De maneira semelhante a descrição dos serviços de outras camadas do RF-OSI/ISO, os elementos de serviços da camada de apresentação são ditos **confirmados** (Figura 2.2) se todas as quatro primitivas de serviços são utilizadas e **não-confirmados** se somente as primitivas *request* e *indication* são utilizadas.

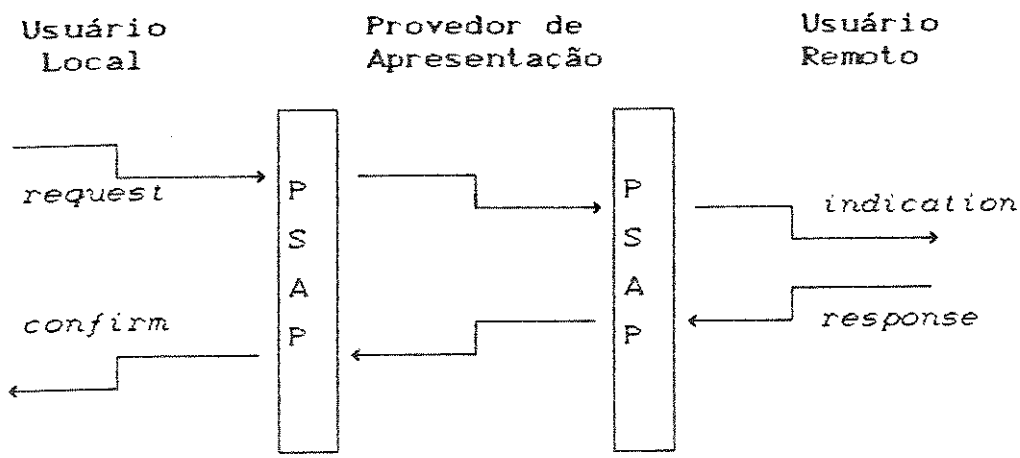


Figura 2.2. - Diagrama Temporal dos Serviços Confirmados

Um caso particular se refere ao serviço de aborto pelo provedor, no caso da apresentação denominado P_P_ABORT, onde sua operação é descrita como "iniciada pelo provedor" e somente a primitiva *indication* é utilizada (Figura 2.3).

Os serviços da camada de apresentação são logicamente agrupados de acordo com as funções que executam. Estes grupos são subdivididos em Elementos de Serviços, que por sua vez são constituídos por

primitivas de serviços de apresentação. São definidos cinco grupos de serviços : Estabelecimento de conexão, Término de conexão, Gerenciamento de conexão, Transferência de informação e Controle de diálogo. A seguir apresenta-se uma breve descrição de cada um destes grupos.

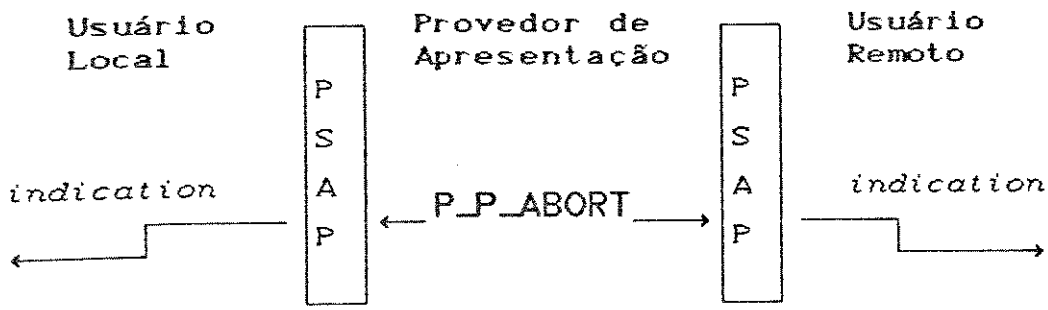


Figura 2.3. - Diagrama Temporal do Serviço P_P_ABORT

2.2.1. - Estabelecimento de Conexão

O grupo P_CONNECT de estabelecimento de conexão (Tabela 2.0) proporciona ao usuário de serviços de apresentação estabelecer uma conexão de apresentação com outro usuário-par. Os usuários de serviços trocam parâmetros entre si, através dos quais estabelecem as características da conexão, em particular : seleção de unidades funcionais dos serviços de apresentação; definição de um conjunto inicial de contextos de apresentação; especificação de uma sintaxe abstrata a ser utilizada no contexto "*default*"; e determinação das características da conexão de sessão.

< Estabelecimento de Conexão >		
Nome do serviço	Tipo de serviço	U.F.
P_CONNECT	Confirmado	*Kernel*

Tabela 2.0. - Grupo de Serviço para o Estabelecimento de Conexão

Na Tabela 2.1 mostram-se os parâmetros definidos para cada

primitiva de serviço do P_CONNECT. Nesta Tabela, para cada parâmetro estão associados qualificadores para cada primitiva de serviço, onde :

- M ("*Mandatory*") - Parâmetro Obrigatório
- U ("*User Option*") - Parâmetro Opcional
- C ("*Conditional*") - Parâmetro Condicional
- S ("*Session*") - Parâmetro requerido pela Sessão
- (branco) - Parâmetro Ausente

O simbolo "(=)" anexado a alguns destes qualificadores indica que o parâmetro é semanticamente igual ao parâmetro da primitiva de serviço da coluna imediatamenta à sua esquerda.

Parâmetro	REQ	IND	RSP	CNF
Endereço de Apresentação Origem	M	M(=)		
Endereço de Apresentação Destino	M	M(=)		
Endereço de Apresentação Respondedor .			U	C(=)
Multiplos Contextos Definido	U	C	U	C(=)
Lista de Definição de Contextos de Apresentação	U	C(=)		
Lista Resultado da Definição de Contextos de Apresentação		C	C	C(=)
Nome do Contexto *Default*	U	C(=)		
Resultado do Contexto *Default*		C	C	C(=)
Qualidade do Serviço	S	S	S	S
Requisitos de Apresentação	U	C	U	C(=)
Requisitos de Sessão	S	S	S	S
Número Serial do Ponto de Sincronização Inicial	S	S	S	S
Atribuição Inicial de *Tokens*	S	S	S	S
Identificador de Conexão de Sessão ...	S	S	S	S
Dados do Usuário	U	C(=)	U	C(=)
Resultado			M	M(=)

Tabela 2.1. - Parâmetros do Serviço P_CONNECT

Por exemplo, o parâmetro "Lista de Definição de Contextos de Apresentação" é opcional (U) e somente está presente quando a camada de aplicação requer colocar um ou mais contextos de apresentação no conjunto de contexto a ser definido no momento do estabelecimento de uma conexão. Se presente na primitiva *request*, possuirá o mesmo valor na primitiva *indication* (C(=)) e provocará o retorno do parâmetro "Lista Resultado da Definição de Contextos

de Apresentação", que está condicionado (C) à presença do primeiro parâmetro.

Na primitiva *indication* (C) este parâmetro indica ao usuário par somente aquelas definições de contexto propostas que não são suportadas pelo provedor de apresentação. Nas primitvas *response* (C) e *confirm* (C(=)) o usuário par indica, dentre os contextos aceitos pelo provedor, as definições aceitas e rejeitadas. Note-se então que o conjunto inicial de contextos de apresentação é negociado entre ambos os usuários e as entidades de apresentação.

2.2.2. - Término de Conexão

O grupo de término de conexão (Tabela 2.2) provê serviços que permitem : a liberação ordenada de uma conexão de apresentação pelo usuário de serviços de uma forma não destrutiva, e o término abrupto de uma conexão de apresentação de uma forma que pode ser destrutiva. Este término abrupto pode ser iniciado por parte de qualquer um dos usuários de serviços ou pelo próprio provedor de apresentação.

O término normal de uma conexão é fornecido pelo serviço P_RELEASE, e proporciona ao usuário de serviços liberar uma conexão de apresentação sem perdas de dados que estejam em trânsito. Esta liberação normal de conexão é obtida através do mapeamento no serviço S_RELEASE da camada de sessão.

< Término de Conexão >		
Nome do serviço	Tipo de serviço	U.F.
P_RELEASE	Confirmado	*Kernel*
P_U_ABORT	Não Confirmado	*Kernel*
P_P_ABORT	Inic. pelo Prov.	*Kernel*

Tabela 2.2. - Grupo de Serviços para o Término de Conexão

O serviço P_U_ABORT é utilizado por qualquer um dos usuários de serviços para forçar a liberação (término abrupto) de uma conexão de apresentação em qualquer instante da comunicação e também para informar o correspondente usuário de apresentação desta liberação. É um serviço destrutivo e que não requer acordo entre ambos os usuários para solicitar a liberação.

De maneira semelhante ao serviço P_U_ABORT, porém agora voltado para o provedor, o serviço P_P_ABORT proporciona meios pelo qual o provedor de apresentação pode indicar o término abrupto de uma conexão de apresentação por razões internas. Também é um serviço destrutivo, pois a sua execução pode interromper qualquer outro serviço ativo que esteja sendo executado concorrentemente.

2.2.3. - GERENCIAMENTO DE CONTEXTO

O grupo P_ALTER_CONTEXT de gerenciamento de contexto (Tabela 2.3) permite : a definição de contextos de apresentação através da negociação entre os dois usuários de serviços e o provedor de apresentação, deixando-os disponíveis para uso durante uma nova instância da comunicação; e a deleção de contextos de apresentação a partir do conjunto de contextos disponíveis para uso.

< Gerenciamento de Contexto >		
Nome do serviço	Tipo de serviço	U.F.
P_ALTER_CONTEXT	Confirmado	G/R

Onde :
G/R - Unidade Funcional de Gerenciamento/Restauração de Contexto

Tabela 2.3. - Grupo para o Gerenciamento de Contexto

A Tabela 2.4 ilustra os parâmetros de todas as primitivas do serviço P_ALTER_CONTEXT.

Parâmetro	REQ	IND	RSP	CNF
Lista de Definição de Contextos de Apresentação	U	C(=)		
Lista de Deleção de Contextos de Apresentação	U	C(=)		
Lista Resultado da Definição de Contextos de Apresentação		C	U	C(=)
Lista Resultado da Deleção de Contextos de Apresentação		C	U	C(=)
Dados do Usuário	U	C(=)	U	C(=)

Tabela 2.4. - Parâmetros do Serviço P_ALTER_CONTEXT

Os parâmetros "Lista de Definição de Contextos de Apresentação" e "Lista de Deleção de Contextos de Apresentação" são opcionais (U), dependendo do tipo de alteração desejada. Ambos tem a forma de uma lista, onde cada elemento representa uma especificação para um contexto de apresentação a ser definido ou deletado do conjunto de contextos de apresentação corrente.

Se estes parâmetros iniciais estão presentes, uma respectiva resposta será enviada através dos parâmetros "Lista Resultado da Definição de Contextos de Apresentação" e "Lista Resultado da Deleção de Contextos de Apresentação". Assim, estes parâmetros indicam a aceitação ou rejeição de cada uma das definições e/ou deleções de contexto proposta nos parâmetros iniciais. Tem a forma de uma lista de valores resultados, onde existe uma correspondência um para um entre os elementos desta lista de valores resultados e os elementos da lista de definição/deleção proposta.

2.2.4. - TRANSFERÊNCIA DE INFORMAÇÃO

O grupo de transferência de informação (Tabela 2.5) fornece serviços que permitem aos usuários trocarem informações durante uma conexão de apresentação. Esta troca de informação pode estar sujeita ao controle do *token* com os serviços P_DATA e

P_CAPABILITY_DATA, ou não estar sujeita a esse controle através do P_EXPEDITED_DATA e P_TYPED_DATA. Estes serviços são mapeados diretamente nos respectivos serviços de sessão.

< Transferência de Informação >		
Nome do serviço	Tipo de serviço	U.F.
P_TYPED_DATA	Não Confirmado	SS
P_DATA	Não Confirmado	*Kernel*
P_EXPEDITED_DATA	Não Confirmado	SS
P_CAPABILITY_DATA	Confirmado	SS

Onde : SS - Unidade Funcional de Sessão

Tabela 2.5. - Grupo de Serviços para Transferência de Informação

2.2.5. - CONTROLE DE DIÁLOGO

O grupo de controle de diálogo (Tabela 2.6) fornece serviços que permitem :

- gerenciamento do *token* : P_TOKEN_GIVE, P_TOKEN_PLEASE e P_CONTROL_GIVE;
- sincronização : P_SYNC_MINOR e P_SYNC_MAJOR;
- ressincronização : P_RESYNCHRONIZE;
- tratamento de excessões : P_U_EXCEPTION_REPORT para os usuários e P_P_EXCEPTION_REPORT para o provedor, e
- gerenciamento de atividades : P_ACTIVITY_START, P_ACTIVITY_END, P_ACTIVITY_INTERRUPT, P_ACTIVITY_RESUME e P_ACTIVITY_DISCARD.

Todos estes serviços são mapeados diretamente nos serviços correspondentes da camada de sessão.

< Controle de Diálogo >		
Nome do serviço	Tipo de serviço	U.F.
P_U_EXCEPTION_REPORT	Não Confirmado	SS
P_P_EXCEPTION_REPORT	Inic. Provedor	SS
P_TOKEN_GIVE	Não Confirmado	SS
P_TOKEN_PLEASE	Não Confirmado	SS
P_CONTROL_GIVE	Não Confirmado	SS
P_SYNC_MINOR	Confir. Opcional	SS
P_SYNC_MAJOR	Confirmado	SS
P_RESYNCRHRONIZE	Confirmado	SS
P_ACTIVITY_START	Não Confirmado	SS
P_ACTIVITY_RESUME	Não Confirmado	SS
P_ACTIVITY_END	Confirmado	SS
P_ACTIVITY_INTERRUPT	Confirmado	SS
P_ACTIVITY_DISCARD	Confirmado	SS

Onde : SS - Unidade Funcional de Sessão

Tabela 2.6. - Grupo de Serviços para o Controle de Diálogo

2.3. - Especificação de Protocolo

A máquina de protocolo de apresentação é composta por entidades de apresentação que se comunicam com os usuários através de PSAPs e por meios das primitivas de serviços de apresentação, como ilustra a Figura 2.4.

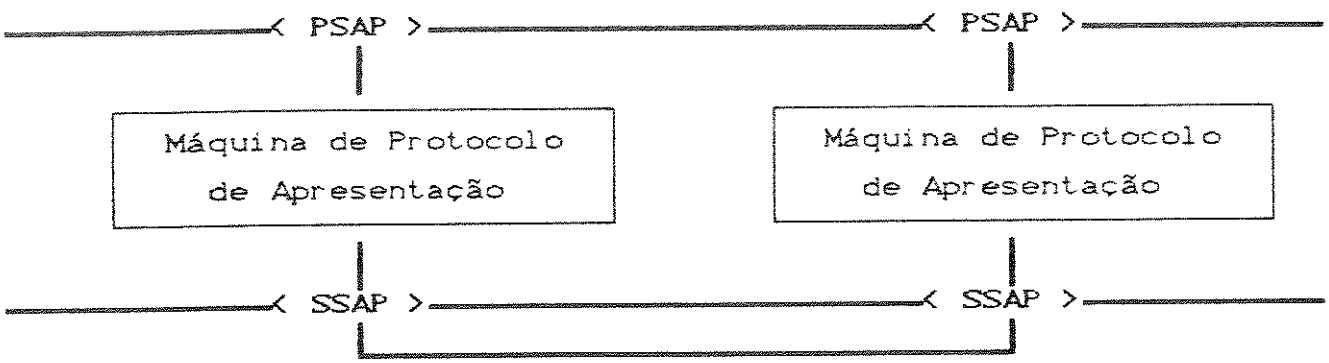


Figura 2.4. - Modelo da Camada de Apresentação

Os elementos de procedimento do protocolo são descritos em função do envio e recebimentos de PPDUs ("*Presentation PDU*"). O protocolo

de apresentação define 14 PPDUs, como mostra a Tabela 2.7.

Código – Nome da PPDU	serviço que gera a PPDU
CP – Connect Presentation	P_CONNECT *request*
CPA – Connect Presentation Accept	P_CONNECT *response(+)*
CPR – Connect Presentation Reject	P_CONNECT *response(-)*
ARU – Abnormal Release User	P_U_ABORT *request*
ARP – Abnormal Release Provider	P_P_ABORT *indication*
TD – Transfer Data	P_DATA *request*
TE – Transfer Expedited Data	P_EXPEDITED_DATA *request*
TTD – Transfer Typed Data	P_TYPED_DATA *request*
TC – Transfer Capability Data	P_CAPABILITY_DATA *request*
TCC – Transfer Capability Data Ack.	P_CAPABILITY_DATA *response*
AC – Alter Context	P_ALTER_CONTEXT *request*
ACA – Alter Context Ack.	P_ALTER_CONTEXT *response*
RS – Resynchronize	P_RESYNCHRONIZE *request*
RSA – Resynchronize Ack.	P_RESYNCHRONIZE *response*

Tabela 2.7. - PPDUs da Camada de Apresentação

2.3.1. - Estabelecimento de Conexão

Uma entidade de apresentação, denominada iniciadora, ao receber uma P_CONNECT *request* envia uma PPDU CP ("*Connect Presentation*") contendo as informações e os parâmetros necessários para a operação de estabelecimento de uma conexão de apresentação. A entidade de apresentação remota, denominada respondedora, pode recusar a conexão (se por exemplo, o valor do parâmetro da PPDU CP ou da S_CONNECT *indication* são inaceitáveis) e, neste caso, envia uma PPDU CPR ("*Connect Presentation Reject*") indicando a razão do provedor. Alternativamente, se o respondedor não recusar a conexão, este envia uma P_CONNECT *indication* para o seu usuário. Se o respondedor então receber uma P_CONNECT *response negative*, este envia uma PPDU CPR indicando a "rejeição pelo usuário"; mas se receber uma P_CONNECT *response positive*, envia uma PPDU CPA

("*Connect Presentation Accept*") aceitando a conexão.

Se o iniciador é incapaz de estabelecer uma conexão devido à impossibilidade de estabelecer uma conexão de sessão, este envia uma P_CONNECT *confirm negative* indicando "rejeição pelo provedor" e a conexão não é estabelecida. Se o iniciador recebe uma PPDU CPR recusando a conexão de apresentação, então envia uma P_CONNECT *confirm negative* indicando "rejeição pelo usuário", se o parâmetro "razão do provedor" não está disponível, ou "rejeição pelo provedor", caso contrário, e a conexão não é estabelecida. Se o iniciador recebe uma PPDU CPA indicando a aceitação da conexão de apresentação, envia uma P_CONNECT *confirm positive* indicando o resultado da solicitação como aceito e a conexão é estabelecida. Neste último caso, o conjunto de contexto de apresentação inicial para ambas as entidades de apresentação é o conjunto negociado nos parâmetros da PPDU CPA.

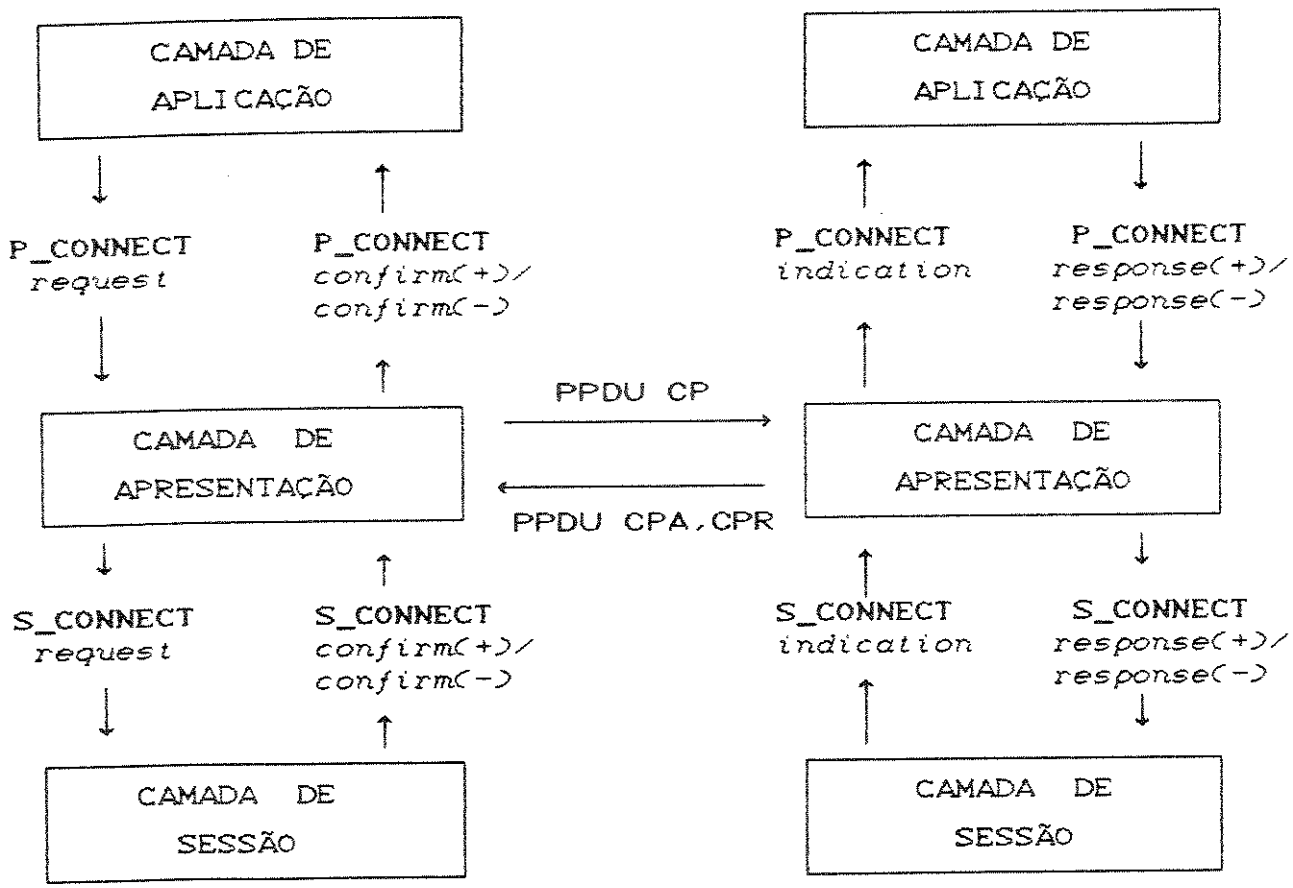


Figura 2.5. - Estabelecimento de Conexão

A Figura 2.5 ilustra o procedimento de estabelecimento de uma conexão de apresentação através da interação entre as primitivas da camada de apresentação e da camada de sessão.

A PPDU CP é transportada da entidade de apresentação iniciadora para a entidade de apresentação respondedora através das primitivas *request* e *indication* do serviço S_CONNECT. A Tabela 2.8 define o mapeamento dos parâmetros desta PPDU sobre os parâmetros do serviço S_CONNECT. Note-se que os parâmetros que são específicos da entidade de apresentação são mapeados no parâmetro "*User-Data*" do serviço de sessão, e os demais são utilizados para preencher os parâmetros da primitiva de serviço S_CONNECT.

Parâmetros da PPDU CP	Parâmetros da S_CONNECT req.
Seletor de Apresentação Origem ..	*session service User-Data*
Endereço de Sessão Origem	Endereço do SSAP Origem
Seletor de Apres. Destino	*session service User-Data*
Endereço de Sessão Destino	Endereço do SSAP Destino
Multiplos Contextos Definido	*session service User-Data*
Lista de Contextos Propostos	*session service User-Data*
Contexto *Default*	*session service User-Data*
Versão do Protocolo	*session service User-Data*
Qualidade do Serviço	Qualidade do Serviço
Requisitos de Apresentação	*session service User-Data*
Requisitos do Usuário de Sessão .	*session service User-Data*
Requisitos de Sessão Revisados ..	Requisitos de Sessão
Número Serial do Ponto de Sincronização Inicial	Número Serial do Ponto de Sincronização Inicial
Atribuição Inicial de *Tokens*	Atrib. Inicial de *Tokens*
Ident. de Conexão de Sessão	Ident. de Conexão de Sessão
Dados do Usuário	*session service User-Data*

Tabela 2.8. - Mapemento dos Parâmetros da PPDU CP sobre os Parâmetros do Serviço S_CONNECT

2.3.2. - Término de Conexão

O procedimento para a liberação normal de uma conexão de apresentação é utilizado por uma entidade iniciadora que tenha recebido uma P_RELEASE *request*. A liberação normal da conexão de

apresentação ocorre em paralelo com a liberação normal da conexão de sessão. Não existem PPDUs definidas explicitamente, mas implicitamente dadas pelo mapeamento sobre os respectivos serviços de sessão.

O procedimento de liberação abrupta de conexão pode ocorrer :

● Quando uma entidade de apresentação recebe uma P_U_ABORT *request* e, ou uma conexão foi estabelecida ou uma PPDU CP foi enviada, e nenhuma PPDU (CPA ou CPR) foi recebida indicando o resultado desta tentativa de estabelecimento de conexão, então é enviada uma PPDU ARU ("*Abnormal Release User*") e a conexão é liberada. A entidade de apresentação remota ao receber uma PPDU ARU envia uma P_U_ABORT *indication* e libera a conexão.

● Quando uma entidade de apresentação recebe uma PPDU não esperada ou ocorre um erro de protocolo, esta envia um P_P_ABORT *indication* para seu usuário e uma PPDU ARP ("*Abnormal Release Provider*") para a entidade de apresentação par e a conexão é liberada. Uma entidade de apresentação ao receber uma PPDU ARP envia um P_P_ABORT *indication* e libera a conexão (Figura 2.6). Uma entidade de apresentação também pode receber um S_P_ABORT *indication* proveniente da camada de sessão e, neste caso, envia um P_P_ABORT *indication* e a conexão também é liberada.

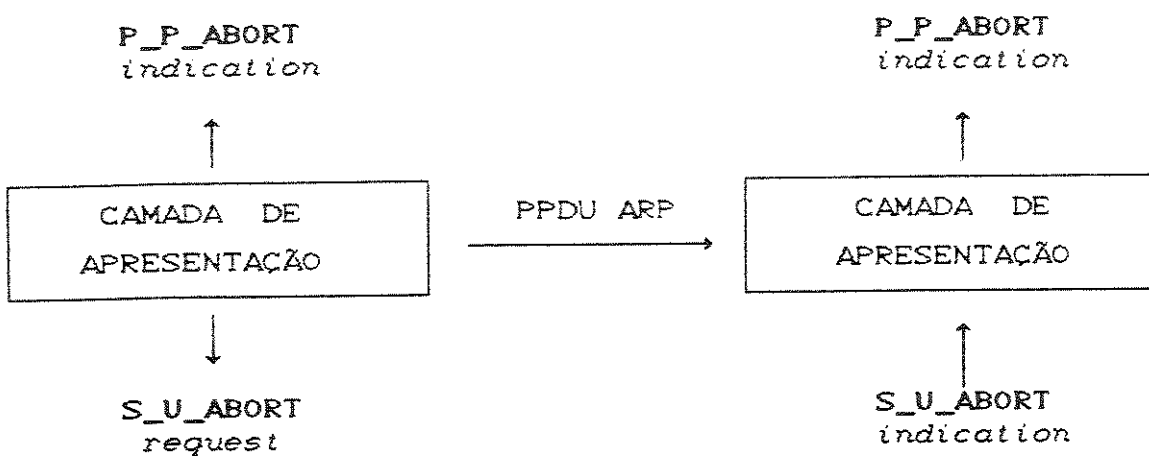


Figura 2.6. - Liberação de Conexão pelo Provedor

2.3.3. - Gerenciamento de Contexto

O procedimento de alteração de contexto é utilizado por qualquer entidade de apresentação iniciadora que tenha recebido uma P_ALTER_CONTEXT *request*. Esta envia uma PPDU AC ("*Alter Context*") contendo informações e parâmetros propostos necessários para a definição de contextos e os identificadores dos contextos propostos para deleção.

A própria entidade de apresentação respondedora pode recusar algumas ou todas as definições de contexto proposta (se, por exemplo, é incapaz de proporcionar as facilidades desejadas). Neste caso, a entidade de apresentação envia uma P_ALTER_CONTEXT *indication* na qual, marca as definições propostas que foram recusadas com o valor "rejeição pelo provedor". Se o respondedor recebe uma P_ALTER_CONTEXT *response*, este envia uma PPDU ACA ("*Alter Context Acknowledge*") indicando a aceitação ou rejeição de cada definição de contexto proposta e de cada deleção de contexto proposta. A entidade de apresentação iniciadora ao receber uma PPDU ACA, envia uma P_ALTER_CONTEXT *confirm*.

Este serviço é o único da camada de apresentação que não possui um serviço semelhante na camada de sessão. Neste caso, a camada de apresentação, para prover tal serviço utiliza-se do serviço S_DATA da camada de sessão.

2.3.4. - Transferência de Informação

O procedimento de transferência de informação é utilizado para transportar unidades de dados do serviço de apresentação originários dos serviços P_DATA, P_TYPED_DATA, P_CAPABILITY_DATA e P_EXPEDITED_DATA.

Quando uma P_DATA *request* é recebida por uma entidade de apresentação, esta envia uma PPDU TD ("*Transfer Presentation*

Data") para transmitir, de acordo com as sintaxes de transferência, a informação expressa na P_DATA *request*. Quando uma entidade de apresentação recebe uma PPDU TD, esta envia uma P_DATA *indication* contendo as informações da P_DATA *request*. A sintaxe abstrata da informação é expressa de acordo com as regras do conjunto de contexto de apresentação corrente. O protocolo do serviço P_TYPED_DATA é semelhante a este serviço, apenas com a substituição da PPDU para TTD ("*Transfer Type Data*").

Quando uma P_EXPEDITED_DATA *request* é recebida por uma entidade de apresentação, esta envia uma PPDU TE ("*Transfer Expedited Data*") na sintaxe de transferência *default*. Quando uma entidade de apresentação recebe uma PPDU TE, esta envia uma P_EXPEDITED_DATA *indication* contendo as informações recebidas.

Quando uma P_CAPABILITY_DATA *request* é recebida por uma entidade de apresentação, esta envia uma PPDU TC ("*Transfer Capability Data*") para transmitir, de acordo com as sintaxes de transferência, a informação expressa na P_CAPABILITY_DATA *request*. Quando uma entidade de apresentação recebe uma PPDU TC, esta envia uma P_CAPABILITY_DATA *indication* contendo as informações recebidas. Se o respondedor então recebe uma P_CAPABILITY_DATA *response*, este envia uma PPDU TCC ("*Transfer Capability Data Acknowledge*") para transmitir, de acordo com as sintaxes de transferência, a informação expressa na P_CAPABILITY_DATA *response*. Quando uma entidade de apresentação recebe uma PPDU TCC, esta envia uma P_CAPABILITY_DATA *confirm* contendo as informações recebidas.

2.3.5. - Controle de Diálogo

Todos os elementos de procedimento dos serviços pertencentes ao grupo de controle de diálogo são utilizados para deixar disponíveis aos usuários de serviços as facilidades de controle de diálogo oferecidas pelos serviços de sessão. Não existem PPDUs definidas explicitamente para estes serviços, mas implicitamente

dadas pelo mapeamento sobre os respectivos serviços de sessão. A Tabela 2.9 mostra todos os serviços da camada de apresentação que são mapeados diretamente nos respectivos serviços da camada de sessão.

Serviço de Apresentação	Serviço de Sessão
P_RELEASE	S_RELEASE
P_TYPED_DATA	S_TYPED_DATA
P_DATA	S_DATA
P_EXPEDITED_DATA	S_EXPEDITED_DATA
P_CAPABILITY_DATA	S_CAPABILITY_DATA
P_TOKEN_GIVE	S_TOKEN_GIVE
P_TOKEN_PLEASE	S_TOKEN_PLEASE
P_CONTROL_GIVE	S_CONTROL_GIVE
P_SYNC_MINOR	S_SYNC_MINOR
P_SYNC_MAJOR	S_SYNC_MAJOR
P_RESYNCHRONIZE	S_RESYNCHRONIZE
P_U_EXCEPTION_REPORT	S_U_EXCEPTION_REPORT
P_P_EXCEPTION_REPORT	S_P_EXCEPTION_REPORT
P_ACTIVITY_START	S_ACTIVITY_START
P_ACTIVITY_END	S_ACTIVITY_END
P_ACTIVITY_INTERRUPT	S_ACTIVITY_INTERRUPT
P_ACTIVITY_RESUME	S_ACTIVITY_RESUME
P_ACTIVITY_DISCARD	S_ACTIVITY_DISCARD

Tabela 2.9. - Serviços de Apresentação Mapeados na Sessão

2.4. - TABELAS DE ESTADO

O protocolo de apresentação também pode ser descrito em termos de TABELAS DE ESTADO. Estas Tabelas de estado não constituem uma definição formal do protocolo, porém proporcionam uma descrição mais precisa dos procedimentos do protocolo, evitando-se

interpretações ambíguas e erradas a partir da descrição textual.

As Tabelas de estado mostram o estado de uma conexão de apresentação, os eventos que ocorrem no protocolo, as ações tomadas e o estado resultante. Em geral, as Tabelas de estado contidas nos documentos OSI possuem os seguintes elementos :

- Eventos de Entrada
- Eventos de Saída
- Estados
- Predicados
- Ações Específicas

No caso do protocolo de apresentação, os eventos de entrada podem ser classificados em três categorias : eventos do usuário de serviços de apresentação, eventos do provedor de serviços de sessão, e eventos caracterizados pela chegada de uma PPDU. De modo semelhante, os eventos de saída podem ser agrupados em três categorias : eventos do usuário de serviços de sessão, eventos do provedor de serviços de apresentação, e eventos caracterizados pelo envio de uma PPDU.

A máquina de protocolo de apresentação (PPM - "*Protocol Presentation Machine*") possui sete estados, como ilustra a Tabela 2.10. Estes estados podem ser classificados em Estado-Presente e Próximo-Estado. O Estado-Presente indica o estado corrente da máquina de protocolo, para uma determinada conexão de apresentação, e o Próximo-Estado o próximo estado que a máquina de protocolo vai assumir após a ocorrência de uma transição de estado.

Os predicados são "atributos" ou condições necessárias para a tomada de uma ação (geração de um evento de saída e/ou execução de ações específicas), mediante o estímulo de um evento de entrada. As ações específicas representam ações executadas sob circunstâncias específicas de cada estado da máquina de protocolo e sempre são acompanhadas de um evento de saída. A ordem de

execução das ações específicas é relevante para a execução correta do protocolo.

Estado	Significado
STAI0	IDLE - Sem Conexão
STAI1	Aguardando PPDU CPA (Connect Pres. Accept)
STAI2	Aguardando P_CONNECT *response*
STAI2	Aguarda P_CONNECT *response*
STAc0	Conectado - Pronto p/ transferência de dados
STAac0	Aguarda PPDU ACA (Alter Context Acknowledge)
STAac1	Aguarda P_ALTER_CONTEXT *response*
STAac2	Aguarda P_ALTER_CONTEXT *response* ou PPDU ACA

Tabela 2.10. - Estados da PPM

Estas Tabelas de estados possuem uma notação própria : Os eventos de entrada, eventos de saída e os estados são representados por seus nomes abreviados, por exemplo, a primitiva de serviço P_CONNECT *request* é representada por P_CON.*req* e a PPDU CP simplesmente por CP; Os Predicados e Ações Especificas são representados pela notação "p-NN" e "[NN]" respectivamente, onde NN corresponde a um número de identificação especificado no documento /ISO-DIS 8823, 1986/. Também são definidos os operadores booleanos AND (&) e NOT (^) utilizados para a especificação de ações condicionais envolvendo expressões de predicados, como por exemplo : (p08 & p12 & ^p24).

Uma entidade de apresentação em qualquer instante no tempo pode estar somente em um único estado (Estado-Presente), dentre um número finito de estados definidos para a PPM. Uma transição de estado é induzida pela chegada de um evento de entrada ocorrido em uma das interfaces da camada e do cumprimento de eventuais condições representadas por predicados e operadores booleanos. Esta transição normalmente resulta na geração de eventos de saída, uma mudança para um novo estado (Próximo-Estado) da máquina de protocolo e, eventualmente, na execução de uma ou mais ações específicas.

No documento /ISO-DIS 8823, 1986/ são definidas tabelas contendo todos os eventos de entrada, eventos de saída, estados, ações específicas e predicados pertinentes ao escopo da camada de apresentação. Adicionalmente, definições das tabelas de estados para o protocolo de estabelecimento de uma conexão, liberação (normal e abrupta) de uma conexão, gerenciamento de contexto, transferência de dados, gerenciamento do *token*, sincronização, ressincronização, gerenciamento de atividades e tratamento de excessões. A título de ilustração, apresenta-se a tabela de estado do serviço de estabelecimento de uma conexão de apresentação (Tabela 2.11).

Estado / Evento	STAI0 Idle sem conexão	STAI1 Await CPA	STAI2 Await P_CON.rsp
CP	~p01 : CPR, STAI0 p01 & p09 : P_CON.ind STAI2		
CPA		p09 : P_CON.cnf(+) [02],[01] STAc0	
CPR		p09 : P_CON.cnf(-) [02], STAI0	
P_CON.req	p05 & p09 : [03] CP , STAI1		
P_CON.rsp(+)			p09 : [02] & [01] CPA, STAc0
P_CON.rsp(-)			p09 : CPR, STAI0
S_CON.rsp(-)		P_CON.cnf(-) STAI0	

Tabela 2.11. - Tabela de Estado do Estabelecimento de uma Conexão

Na Tabela 2.11 podemos observar, por exemplo, que quando ocorre a chegada de um evento de entrada do tipo P_CONNECT *request* (P_CON.req), a PPM está no estado STAI0 ("*Idle*") e os predicados p05 e p09 são satisfeitos, é executada a ação específica [03], gerado um evento de saída que corresponde ao envio de uma PPDU CP e a PPM passa para o estado STAI1. Quando a entidade de apresentação remota recebe a PPDU CP e a PPM se encontra no estado STAI0, duas transições são possíveis : Primeira, se a expressão boolena (^p01) é verdadeira, é enviada uma PPDU CPR, indicando a rejeição do pedido de estabelecimento da conexão e a PPM permanece no estado STAI0; e segunda, se os predicados p01 e p09 são satisfeitos, é enviada uma primitiva de serviço P_CONNECT *indication* (P_CON.ind) e a PPM passa para o estado STAI2. Note-se que a tabela especifica os estados das entidades de apresentação local e remota

2.5. - Escopo de Implementação

A camada de apresentação alvo é voltada para o padrão MAP /GM-MAP, 1987/. Neste padrão somente estão definidos os serviços pertencentes à unidade funcional "*kernel*" de apresentação, uma vez que para a camada de sessão também estão definidos apenas os serviços pertencentes à unidade funcional "*kernel*" de sessão com operação duplex.

Os serviços de apresentação pertencentes à esta unidade funcional estão sempre disponíveis, isto é, são não negociáveis, e porporcionam facilidades para os serviços com conexão. Para o estabelecimento de conexão está definido o serviço P_CONNECT, para a transferência de dados o serviço P_DATA, e para a liberação de conexão os serviços P_RELEASE, P_U_ABORT e P_P_ABORT.

Estes serviços compõem o provedor de apresentação do sistema SISDI-MAP (Figura 2.7). Assim, o protocolo de apresentação, no caso deste trabalho especificamente, possui as seguintes funções :

- Tratar as primitivas de serviços do MMS local, referentes à transferência normal de dados, mapeando-as em PPDUs a serem enviadas ao MMS remoto via primitivas de serviços da camada de sessão;
- Tratar as primitivas de serviços do ACSE local, referentes ao estabelecimento e liberação de conexão, mapeando-as em PPDUs a serem enviadas ao usuário (ACSE) remoto via primitivas de serviços da camada de sessão;
- Tratar as PPDUs válidas e inválidas recebidas do usuário remoto via as primitivas de serviços da camada de sessão.

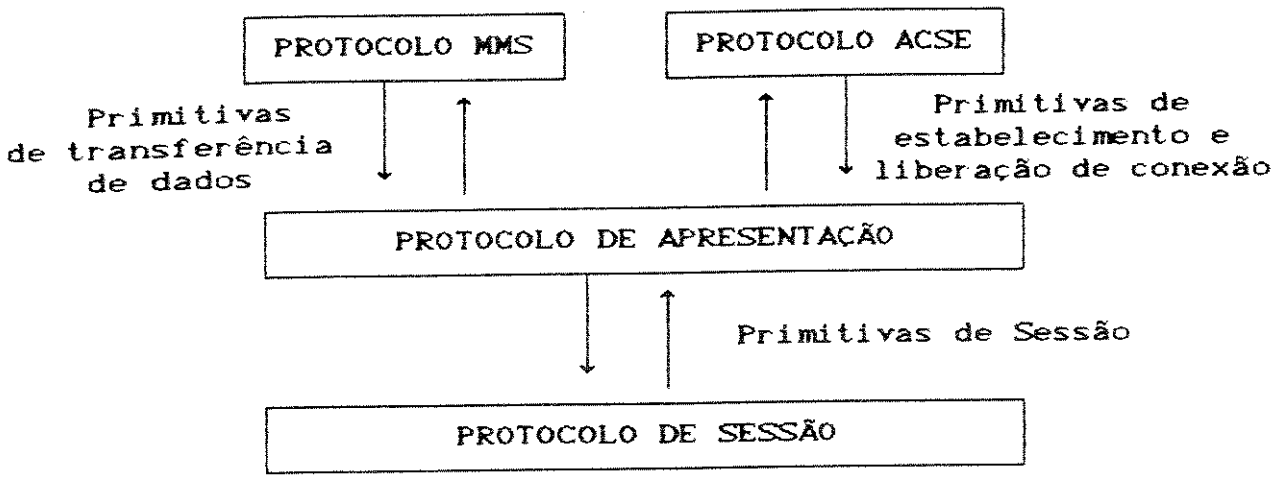


Figura 2.7. - A Camada de Apresentação no SISDI-MAP

A definição das estruturas das PDUs associadas as entidades de aplicação são descritas em sintaxe abstrata, no caso, a ASN.1 ("*Abstract Syntax Notation One*") /ISO-DIS 8824, 1986/. Associada a esta notação de sintaxe abstrata existe um conjunto de regras de codificação e decodificação que permitem transformar as sequências de campos especificados na sintaxe local, em uma sequência de octetos na sintaxe de transferência. Atualmente a notação adotada para a sintaxe de transferência é a BER ("*Basic Encoding Rules*") descrita no documento /ISO-DIS 8825, 1986/.

A realização das respectivas funções de codificação e decodificação para as PDUs definidas no SISDI-MAP não fazem parte do escopo deste trabalho, uma vez que está em desenvolvimento uma

ferramenta /Restovic, 1990/ para proporcionar tais facilidades. Esta ferramenta consiste de um compilador para a notação ASN.1, que à partir da descrição da PDU em sintaxe abstrata produz uma equivalente estrutura de dados em C, e de um gerador de programas codificadores e decodificadores que transformarão os valores dos tipos de dados em sequências de octetos e vice-versa.

2.6. - Especificação dos Requisitos

Neste ítem apresenta-se a especificação dos requisitos referente à implementação dos serviços pertencentes à unidade funcional "*kernel*" da camada de apresentação. A realização desta especificação foi efetuada utilizando-se a linguagem de especificação EPOS-R /EPOS, 1988/. Uma especificação em linguagem EPOS-R (Anexo A) é composta por três ítens :

- Descrição do problema,
- Projeto conceitual e,
- Dicionário de termos.

A seguir apresenta-se um resumo da realização da descrição do problema e do projeto conceitual. A documentação completa da especificação realizada, juntamente com o dicionário de termos pode ser vista no relatório técnico /Inazumi, 1990/

2.6.1. - Descrição do Problema

- Serviços Implementados

A camada de apresentação alvo é voltada para o padrão MAP, que visa aplicações na área de automação industrial. Este padrão MAP, dentre as unidades funcionais definidas pelo modelo de referência OSI/ISO para a camada de apresentação, utiliza-se apenas dos serviços e protocolos pertencentes à unidade funcional "*kernel*", que são :

- Serviço P_CONNECT
- Serviço P_RELEASE
- Serviço P_U_ABORT
- Serviço P_P_ABORT
- Serviço P_DATA

● Documentos ISO

Este trabalho tem como base os seguintes documentos ISO :

- Para a definição dos serviços de apresentação : Draft International Standart ISO/DIS 8822 - (Information Processing Systems - Open Systems Interconnection - Connection Oriented Presentation Service Definition), e
- Para a definição do protocolo de apresentação : Draft International Standart ISO/DIS 8823 - (Information Processing Systems - Open Systems Interconnection - Connection Oriented Presentation Protocol Specification).

● Integração ao sistema SISDI-MAP

Os serviços e protocolos implementados devem ser mapeados em um processo, denominado Proc_Presentation. Este processo será parte integrante do software aplicativo do sistema Didático SISDI-MAP. A integração deste processo Proc_Presentation ao sistema SISDI-MAP se dará pela substituição do simulador das funções da camada de apresentação, hoje existente no sistema, pelo respectivo processo Proc_Presentation. Após esta integração, o processo de simulação fica restrito a simular as funções das camadas inferiores à camada de apresentação, no caso, a camada de sessão e a rede de comunicação, proporcionando o suporte necessário para a comunicação fim-a-fim entre os protocolos deste nível.

● Modularidade

O requisito de modularidade se faz necessário devido ao mapeamento dos serviços e protocolos em um processo, e também a necessidade de proporcionar um aumento progressivo na complexidade de seus

serviços sem afetar seus vizinhos de comunicação (camadas adjacentes).

Assim, uma vez bem definida as interfaces entre a camada superior (Camada de Aplicação), camada inferior (Camada de Sessão) e com a interface de Operação de Usuário/Gerenciamento, possibilita-se a alteração completa/parcial das funções da camada sem a necessidade de reestruturar todo o sistema. Além disso, este processo Proc_Presentation pode ser desenvolvido separadamente dos demais softwares aplicativos do sistema SISDI-MAP, pois todos são funcionalmente independentes.

- Expansibilidade

A concepção do modelo de implementação deve ser realizada de tal maneira a permitir uma expansão "amigável" dos serviços e protocolos pertencentes às demais unidades funcionais da camada de apresentação. Tal modelo deve prever futuras expansões e estar apto a agregar novas funções, sem a necessidade de uma remodelação brusca da estrutura em vigor e assim reduzir o tempo de desenvolvimento de um novo modelo de implementação. Este é um requisito importante, visto que a camada de apresentação dispõe de várias outras primitivas de serviços e protocolos agregados em outras unidades funcionais, além da unidade funcional "*kernel*".

- Ambiente

O ambiente de desenvolvimento e de execução deste projeto será um microcomputador compatível com a linha IBM-PC. Para proporcionar um suporte multitarefa neste tipo de equipamento será utilizado um software básico dedicado e customizado (núcleo de tempo real) para comunicação e sincronização entre tarefas. Tal software básico é baseado no próprio sistema operacional MS-DOS.

- Linguagem de programação

A linguagem de programação utilizada para desenvolver o sistema

SISDI-MAP, tanto do software básico como do software aplicativo, é a linguagem "C". A escolha de tal linguagem de programação deve-se ao fato de ser uma linguagem de alto nível, estruturada, independente da máquina, e que proporciona a produção de códigos eficientes. Além disso, permite a definição de novos tipos abstratos de dados, possibilitando a criação de estruturas de dados que modelam de forma mais adequada os dados da aplicação.

- Ferramentas

As seguintes ferramentas de software serão utilizadas no desenvolvimento deste projeto :

- Sistema de desenvolvimento de software EPOS
- Compilador da linguagem C

- Serviços utilizados da camada de sessão

A camada de apresentação para prover seus serviços às entidades de aplicação faz uso dos serviços da camada de sessão. No caso do escopo deste trabalho, os seguintes serviços de sessão serão utilizados :

- Serviço S_CONNECT
- Serviço S_RELEASE
- Serviço S_U_ABORT
- Serviço S_DATA

2.6.2. - PROJETO CONCEITUAL

O projeto conceitual representa uma solução conceitual para o problema em questão. No caso deste trabalho o projeto conceitual do módulo funcional de apresentação foi realizado utilizando-se a descrição do protocolo em forma de Tabelas de estado, descrita no documento /ISO-DIS 8823, 1986/.

A opção por selecionar tal especificação em Tabelas de estado recaiu sobre o fato de existir um elemento na linguagem EPOS-R

capaz de representar formalmente tais tabelas de estado. No caso, o elemento <DECISION-PROCESS>. Todos os elementos presentes em uma Tabela de estado (item 2.4) foram possíveis de serem representados nos processos de decisão do EPOS-R. A seguir, apresenta-se como foi efetuada a representação das tabelas de estado presentes nos documentos OSI em processos de decisão da linguagem EPOS-R.

A lista de condições (CONDITIONLIST) ficou composta pelos seguintes elementos da tabela de estado : Estado-Presente, Eventos de Entrada e Predicados. Os eventos de entrada foram agrupados em uma lista, que deve conter os eventos do usuário de serviços de apresentação, os eventos do provedor de serviços de sessão, e os eventos caracterizados pela chegada de uma PPDU. Os estados-presente também foram agrupados em uma lista que deve conter todos os estados que são tratados pela máquina de protocolo do serviço em questão.

Os predicados, ao contrário dos elementos anteriores, foram especificados individualmente, ou seja, cada predicado representa no processo de decisão uma condição. Esta opção recaiu sobre dois problemas : primeiro, se fosse construída uma lista de predicados não seria possível selecionar mais do que um predicado por regra, fato que é comum nos documentos OSI, e segundo, um predicado, além de poder estar presente ou ausente em uma determinada "célula" da tabela de estado, se presente pode possuir um operador booleano NOT associado, dificultando a representação dos três estados (presente-verdadeiro, presente-falso e ausente) como um único elemento da lista de predicados.

O segundo problema poderia ser solucionado se considerarmos o predicado e o predicado negado como dois predicados distintos, mas isto não solucionaria o primeiro problema. Diante disto, optou-se por uma solução mais geral, especificando-se individualmente cada predicado com dois valores, T ("*true*") e F ("*false*"), onde a seleção ou não do predicado na lista de condições da regra indica a presença/ausência do mesmo, e se presente, a seleção de um dos valores da lista (T ou F) indica se é presente-verdadeiro ou

presente-falso.

A lista de operações (OPERATIONLIST) ficou composta pelos seguintes elementos : Eventos de Saída, Próximo-Estado, Ações-específicas e, eventualmente, uma situação de erro do protocolo. A especificação destas operações no processo de decisão, ao contrário de algumas condições (CONDITIONLIST), sempre é feita individualmente, isto é, cada elemento corresponde a uma operação da tabela de decisão. Isto se deve ao fato de que na especificação das regras somente é permitido selecionar se uma operação (ação) é executada ou não, e não listar as operações a exemplo de como é feito com as condições.

Não raro, as tabelas de estado apresentam "células" em branco. Tais espaços em branco, onde nenhuma ação e nenhuma transição de estado é definida na intersecção entre um evento de entrada e um estado da máquina de protocolo, correspondem a situações de erro (intersecções inválidas), e para tal foram considerados na especificação desta solução conceitual. No caso, representou-se tais espaços em branco por uma regra de erro (<ELSE RULE>) no processo de decisão.

Se uma tabela de estado é demasiada extensa, a sua representação em processo de decisão pode ser dividida em vários outros processos de decisão. O vínculo entre os vários processos de decisão é então estabelecido na lista de condições através da palavra chave <LOOKAT>.

Outro aspecto importante do processo de decisão se refere à possibilidade de especificar a ordem de execução das ações a serem tomadas, se as condições das regras são cumpridas. Esta situação pode ser exigida durante a execução de mais de uma ação específica, onde a ordem de execução é importante.

A título de ilustração, apresenta-se a representação da tabela de estado do serviço P_CONNECT, utilizado para estabelecer uma conexão de apresentação, em processo de decisão (Figura 2.8). Este

processo de decisão pode ser visto como um requisito (<REQUIREMENT>) do sistema de software a ser especificado. A partir deste processo de decisão, o EPOS gera automaticamente a tabela de decisão correspondente, e que pode ser vista através das tabelas 2.11.a e 2.11.b.

```
DECISION-PROCESS servico-p_connect

CONDITIONLIST :
  Estado-presente(STAI0, STAI1, STAI2),
  Eventos-de-entrada(P_CON_req,P_CON_rsp_pos,P_CON_rsp_neg,
                     PPDU-CPA,PPDU-CPR,PPDU-CP,S_CON_cnf_neg),
  Predicado-01(T, F),
  Predicado-05(T, F),
  Predicado-09(T, F).

OPERATIONLIST :
  LOOKAT eventos-de-saida
  STAI0,
  STAI1,
  STAI2,
  STAc0,
  ERRO-10.

RULES :
  (STAI0, P_CON_req     , -, T, T ; X,-,X,-,-,-),
  (STAI1, PPDU-CPA      , -, -, T ; X,-,-,-,X,-),
  (STAI1, PPDU-CPR      , -, -, T ; X,X,-,-,-,-),
  (STAI1, S_CON_cnf_neg, -, -, - ; X,X,-,-,-,-),
  (STAI0, PPDU-CP       , F, -, - ; X,X,-,-,-,-),
  (STAI0, PPDU-CP       , T, -, T ; X,-,-,X,-,-),
  (STAI2, P_CON_rsp_pos, -, -, T ; X,-,-,-,X,-),
  (STAI2, P_CON_rsp_neg, -, -, T ; X,X,-,-,-,-),
                         (  ,  ,  ,  ,  ; -,-,-,-,-,X).
```

Figura 2.8. - Processo de Decisão do Serviço P_CONNECT

Observa-se que o processo de decisão foi representado em duas tabelas de decisão por motivos de espaço de representação, pois existem 8 regras além da regra ELSE.

Por exemplo, a regra 1 do processo de decisão do serviço P_CONNECT (Figura 2.10) : (STAI0, P_CON_req, -, T, T ; X, -, X, -, -, -) é representada pelas condições :

C1 = STAI0, (Estado Presente STAI0)
C2 = P_CON_req, (Evento de Entrada P_CONNECT *request*)
C3 = - (Condição indiferente)
C4 = T (Predicado 05 com valor *True*)
C5 = T (Predicado 09 com valor *True*)

e pelas seguintes operações :

O1 = X (Gera um evento de saída)
O1 = - (Operação não executada)
O1 = - (Operação não executada)
O4 = X (Passa a PPM para o estado STAI2)
O1 = - (Operação não executada)
O1 = - (Operação não executada)

Assim, foram especificadas as tabelas de estados de todos os serviços pertencentes à unidade funcional "*kernel*" de apresentação. A documentação completa desta especificação consta no relatório técnico /Inazumi, 1990/.

DT: servico-p_connect 1 OF 2		1	2	3	4	5
C 1	Estado-presente	STAI0	STAI1	STAI1	STAI1	STAI0
C 2	Eventos-de-entrada	P_CON_re	PPDU-CPA	PPDU-CPR	S_CON_cn	PPDU-CP
C 3	Predicado-01	-	-	-	-	F
C 4	Predicado-05	T	-	-	-	-
C 5	Predicado-09	T	T	T	-	-
O 1	eventos-de-saida	X	X	X	X	X
O 2	STAI0	-	-	X	X	X
O 3	STAI1	X	-	-	-	-
O 4	STAI2	-	-	-	-	-
O 5	STAc0	-	X	-	-	-
O 6	ERRO-10	-	-	-	-	-

Tabela 2.11.a. - Tabela de Decisão (1/2) do serviço P_CONNECT

DT: servico-p_connect 2 OF 2		6	7	8	ELSE
C 1	Estado-presente	STAI0	STAI2	STAI2	
C 2	Eventos-de-entrada	PPDU-CP	P_CON_rs	P_CON_rs	
C 3	Predicado-01	T	-	-	
C 4	Predicado-05	-	-	-	
C 5	Predicado-09	T	T	T	
O 1	eventos-de-saida	X	X	X	-
O 2	STAI0	-	-	X	-
O 3	STAI1	-	-	-	-
O 4	STAI2	X	-	-	-
O 5	STAc0	-	X	-	-
O 6	ERRO-10	-	-	-	X

Tabela 2.11.b. - Tabela de Decisão (2/2) do serviço P_CONNECT

CAPÍTULO III - IMPLEMENTAÇÃO DA CAMADA DE APRESENTAÇÃO

3.0. - INTRODUÇÃO

Neste capítulo apresenta-se o modelo de implementação, principalmente os mecanismos de troca de mensagens, as portas de comunicação e as estruturas de dados das mensagens. E a seguir, uma visão geral de como foi realizada a respectiva especificação formal da implementação utilizando-se a linguagem EPOS-S. Apresenta-se a estrutura do software implementado e os principais procedimentos de tratamento de mensagens.

A partir da especificação da implementação em EPOS-S realizou-se a geração automática de código utilizando-se a ferramenta C-Composer. Assim, apresentam-se alguns aspectos referentes à transformação de ações, procedimentos, estruturas de controle, etc. para código fonte em linguagem C.

Finalmente, apresenta-se uma parte da documentação gráfica gerada pela ferramenta EPOS-D, baseada na especificação da implementação em EPOS-S. Esta documentação consiste basicamente em diagramas hierárquicos, redes de petri e fluxogramas.

3.1. - O AMBIENTE MULTITAREFA

O projeto SISDI-MAP foi concebido para equipamentos compatíveis com a linha IBM/PC, porém necessita de um ambiente multitarefa para executar os diversos processos que representam os protocolos que o compõem. Para prover tal ambiente multitarefa faz-se uso de um núcleo de tempo real /Zabeu, 1989/ dedicado. Este núcleo é dividido em gerências funcionais. O termo gerência refere-se a um grupo de funções que fornece serviços correlatos. São definidas as seguintes gerências :

- **Gerência de Escalação de Tarefas** : Provê serviços para a execução de tarefas do núcleo. São serviços referentes a criação e término de uma tarefa, reescalonamento, suspensão e ativação de uma tarefa.
- **Gerência de Filas** : Permite a manipulação de filas de forma uniforme por todos os serviços do núcleo.
- **Gerência de Memória** : Aloca e libera áreas de memória para a aplicação.
- **Gerência de Interrupções** : Reprograma o relógio da máquina e fornece funções para o controle de concorrência e para o inicio efetivo da operação multitarefa.
- **Gerência de Sincronização e Comunicação** : Fornece serviços para a comunicação entre tarefas através da manipulação de portas de comunicação com uma temporização associada.
- **Gerência de Semáforos** : Fornece mecanismos para a sincronização de eventos e controle no acesso a recursos que operam em exclusão mútua.
- **Gerência de Entrada/Saída** : Fornece interfaces para memória de massa.

As funções deste núcleo de tempo real são providas aos processos de aplicação na forma de uma biblioteca de funções.

3.2. - O Modelo de Implementação

O módulo funcional de apresentação foi mapeado em um único processo ("*task*") denominado Proc_Presentation, que executa todas as funções dos serviços de apresentação pertencentes a unidade funcional "*kernel*". Assim, existe uma única instância do processo que serve simultaneamente as entidades de aplicação local e remota.

A comunicação entre o processo Proc_Presentation e os demais processos com que troca informações é realizada através de troca de mensagens e de maneira assíncrona. A Figura 3.0 ilustra, de forma esquemática, o modelo de implementação proposto.

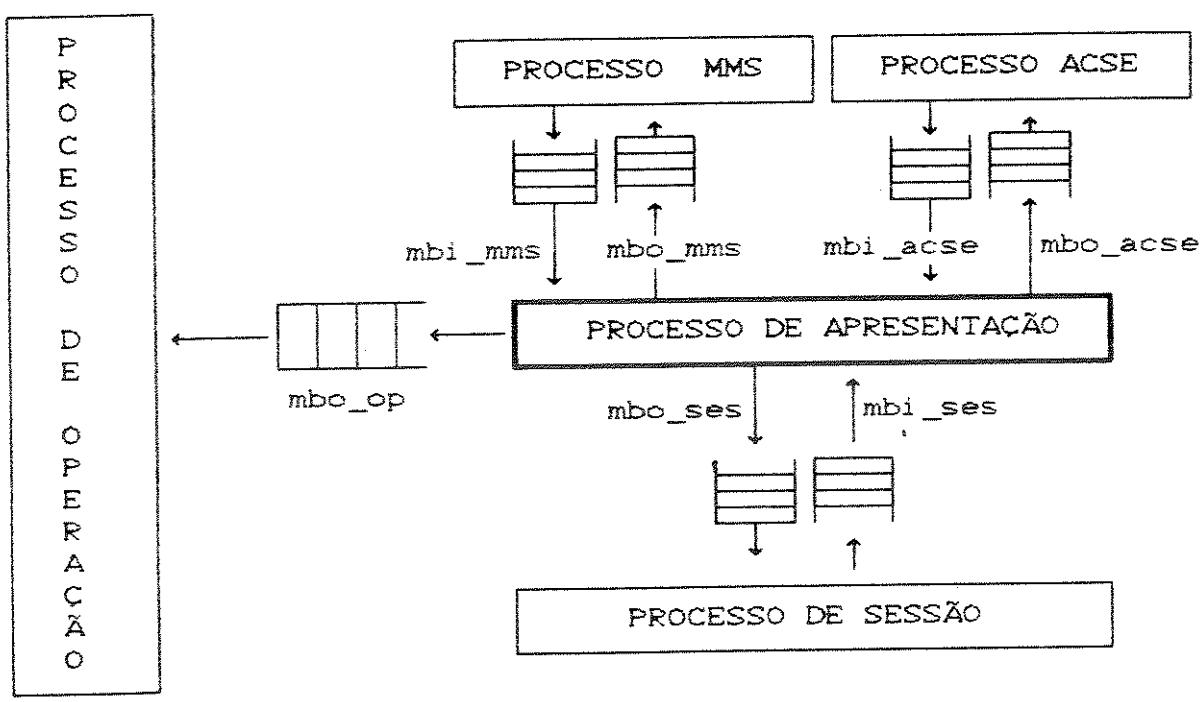


Figura 3.0. - Modelo de Implementação da Camada de Apresentação

A seguir apresentamos uma explanação mais detalhada sobre este modelo de implementação.

3.2.1. - PORTAS DE COMUNICAÇÃO

A troca de mensagens entre os processos é realizada através de "*mail-boxes*" providos funcionalmente na forma de portas de comunicação pelo núcleo de tempo real. Tais portas de comunicação implementam o conceito de SAP ("*Service Access Point*") do modelo de referência OSI/ISO.

Portas de comunicação são providas pelo núcleo de maneira bidirecional e independente do processo que a manipula, porém para facilitar a identificação da origem da mensagem optou-se pela manipulação unidirecional das mesmas e pela associação de forma estática ao processo, de modo a simplificar o modelo de implementação. Assim, para cada processo com quem a apresentação se comunica, à exceção do processo de Operação de Usuário, existem duas portas de comunicação, uma para recepção e outra para o envio de mensagens.

Como se pode observar na Figura 3.0 o processo de apresentação manipula sete portas de comunicação, sendo três portas de entrada e quatro portas de saída. São elas :

- mbi_mms : Porta de Entrada do processo MMS
- mbo_mms : Porta de Saída do processo MMS
- mbi_acse : Porta de Entrada do processo ACSE
- mbo_acse : Porta de Saída do processo ACSE
- mbi_ses : Porta de Entrada do processo de Sessão
- mbo_ses : Porta de Saída do processo de Sessão
- mbo_op : Porta de Saída do processo de Op. de Usuário

Note-se que existe uma única porta de comunicação definida entre o processo de apresentação e o processo de operação de usuário. Isto se deve ao fato de que o processo de operação de usuário não envia mensagens para o processo de apresentação, apenas recebe mensagens de controle visando informar ao usuário do sistema o "*status*" do serviço de apresentação solicitado. As portas definidas entre o

processo de apresentação e o processo MMS (mbi_mms e mbo_mms), e entre o processo de apresentação e o processo ACSE (mbi_acse e mbo_acse) representam os PSAPs da camada de apresentação.

3.2.2. - O Mecanismo de Envio de Mensagens

Durante a comunicação, o mecanismo de envio de mensagens é não bloqueante, isto é, o serviço de envio de mensagens provido pelo núcleo não bloqueia o processo que está enviando a mensagem, apenas a insere na porta de comunicação de saída especificada e após, dependendo da prioridade do processo que recebe a mensagem, retorna o controle para o processo emissor da mensagem ou o suspende se este possuir uma prioridade menor que o processo receptor da mensagem.

Este mecanismo de envio de mensagens é fornecido pelo serviço "envia_mens", onde são especificados a porta de saída e o endereço da mensagem a ser enviada, de modo que apenas este endereço é inserido na porta de comunicação. Isto evita a cópia desnecessária de dados trocados entre os processos comunicantes do sistema.

3.2.3. - O Mecanismo de Recepção de Mensagens

O mecanismo de recepção de mensagens, ao contrário do envio, pode ser temporizado e, neste caso, o processo aguarda pela chegada de uma mensagem por um tempo pré-estabelecido. Caso não exista mensagem na porta especificada, o processo é suspenso até a chegada de uma mensagem ou até expirar o tempo de espera. Esta temporização tem por objetivo evitar esperas infinitas por mensagens.

Este mecanismo de recepção de mensagens é fornecido pelo serviço "espera_mens" com a opção "CONSOME". São passados como parâmetros o identificador da porta onde se busca a mensagem, o tempo em segundos que o processo pode esperar pela chegada da mensagem, e

se a mensagem é recebida dentro do tempo especificado o endereço da mesma é retornado como parâmetro de saída; caso contrário, é gerado um código de tempo expirado.

3.2.4. - TEMPORIZADORES

Foi definido um único temporizador no processo de apresentação. Este temporizador é utilizado pelo serviço "espera_mens" para especificar o tempo máximo de espera por uma mensagem nas portas de comunicação. Possui o valor mínimo permitido pelo núcleo, no caso 1 segundo, pois como o processo de apresentação efetua uma varredura cíclica em apenas três portas de entrada (ítem 3.3.2) não se faz necessário dispender muito tempo na espera destas mensagens, sendo suficiente realizar uma "verificação rápida" se existe ou não mensagens e, se existirem, consumi-las e executar o respectivo tratamento.

3.2.5. - BLOCO DE MENSAGENS DE PROTOCOLO

É definida uma estrutura denominada Bloco de Mensagens de Protocolo /Jacinto, 1989/, que é utilizada na interface entre todos os processos do Sistema Didático, exceto com o processo de operação de usuário. Esta estrutura visa minimizar a alocação de memória, a cópia desnecessária de informações, otimização do mecanismo de identificação das mensagens pelos processos receptores, e pode representar qualquer primitiva de serviço (PCI - "*Protocol Control Information*" + SDU - "*Service Data Unit*") circulante no sistema. Este bloco de mensagens é uma estrutura de dados ("struct" da linguagem C) que contém um cabeçalho comum e um registro específico definido para cada tipo de serviço, como ilustra a Figura 3.1.

O cabeçalho contém :

- Identificador da conexão : Identificador final da conexão,

representando um CEP ("*Connection End Point*").

- Tipo da primitiva de serviço : Pode assumir os seguintes valores : *request*, *indication*, *response_pos*, *response_neg*, *confirm_pos* e *confirm_neg*.
- Tipo de serviço : No caso deste processo de apresentação sempre possui o valor "*presentation*" quando da recepção de uma mensagem, e no envio indica o processo destino da mensagem.
- Tamanho da área de dados : Tamanho do registro específico.

```
typedef struct
    {
    Connection_id     connection_id;
    Primitive_type    primitive;
    Service_type      service;
    sint16            data_area_lenght;
    union  /* Area de Dados de cada tipo de servico */
        {
        Mms_conf_struct        mms_conf_serv;
        Mms_unconf_struct      mms_unconf_serv;
        Mms_cont_struct        mms_cont_serv;
        Acse_struct            acse_serv;
        Presentation_struct    pres_serv;
        Session_struct         ses_serv;
        Transport_struct       transp_serv;
        } data;
    } Block_struct;
```

Figura 3.1. - Bloco de Mensagens de Protocolo

O registro específico é selecionável (declaração "union" da linguagem C) conforme o tipo de serviço especificado no cabeçalho. No caso da apresentação é definida a estrutura "Presentation_struct", mostrada na Figura 3.2.

De modo semelhante ao Bloco de Mensagens de Protocolo, a estrutura de apresentação é composta por um cabeçalho comum a todos os

serviços de apresentação e uma parte específica de acordo com a primitiva de serviço selecionada. O cabeçalho é composto pelo nome do serviço de apresentação, especificado no campo "*service_name*" e pela reserva da área de dados para as demais camadas inferiores definidas no Sistema Didático, no caso a camada de sessão (dummy DS) e a camada de transporte (dummy DT). Note-se que as áreas de dados das camadas inferiores são vistas pelo processo de apresentação através de uma máscara dummy ("áreas Dummy")

```
typedef struct
    {
    Presentation_service  service_name;
    Dummy                 dummy_para_apresentacao[DT+DS];
    union
        {
        P_connection_struct     p_connection_serv;
        P_release_struct        p_release_serv;
        P_U_abort_struct        p_u_abort_serv;
        P_P_abort_struct        p_p_abort_serv;
        P_data_struct           p_data_serv;
        } prim_pres;
    } Presentation_struct;
```

Figura 3.2. - Estrutura de Dados da Apresentação

A parte específica é selecionável (declaração "union" da linguagem C) de acordo com o tipo de serviço de apresentação e assim definiu-se uma estrutura de dados para cada serviço pertencente a unidade funcional "*kernel*" de apresentação, que são : P_CONNECT (P_connection_struct), P_RELEASE (P_release_struct), P_P_ABORT (P_P_abort_struct), P_U_ABORT (P_U_abort_struct) e P_DATA (P_data_struct). A definição destas estruturas de dados consta no relatório técnico /Inazumi, 1990/.

3.2.6. - Estruturas de Dados

Pelo fato de existir um único processo de apresentação para as entidades local e remota, fez-se necessário criar uma estrutura de dados interna ao processo de apresentação de modo a permitir um gerenciamento das várias instâncias do processo. Esta estrutura de dados, denominada Pres_Tab, é mostrada na Figura 3.3.

```
typedef struct
      { Connection_id        connection_id;
        Pres_state_type      pres_state;
        Pres_user_type       user_type;
      } Pres_tab;
```

Figura 3.3. - Estrutura de Dados PRES_TAB

Cada instância do processo de apresentação tem seu "*status*" armazenado em uma estrutura de dados deste tipo. Deste modo, esta estrutura é indexada e limitada pelo número máximo de contextos ("MAX_CONTEXT") definido pelo sistema SISDI-MAP. É composta pelo identificador ("*connection_id*") da conexão de apresentação; pelo estado corrente ("*pres_state*") da Máquina de Protocolo; e pelo tipo de entidade de apresentação (local ou remota) que é indicada pelo tipo de usuário da apresentação ("*user_type*"), e que pode ser "origem" ou "destino".

3.2.7. - Parâmetros de Inicialização

Devido ao fato das portas de comunicação manipuladas pela apresentação estarem associadas ao processo de apresentação, durante a criação deste, devem ser passados de forma parametrizada os endereços das portas de entrada, de onde recebe mensagens, e os endereços das portas de saída, para onde envia mensagens. Tais portas de comunicação são :

- Ingate_Mms, Outgate_Mms : Portas de recepção e envio de mensagens do/para o processo MMS, respectivamente.
- Ingate_Acse, Outgate_Acse : Portas de recepção e envio de mensagens do/para o processo ACSE, respectivamente.
- Ingate_Ses, Outgate_Ses : Portas de recepção e envio de mensagens do/para o processo de Sessão, respectivamente.
- Outgate_Op : Porta de envio de mensagens para o processo de Operação de Usuário.

3.3. - Especificação da Implementação

Para a realização da especificação da implementação do processo de apresentação utilizou-se a linguagem EPOS-S (Anexo B), através de seus elementos formais, denominados objetos de projeto. A seguir apresenta-se um extrato da especificação realizada, descrevendo-se os principais módulos, ações e procedimentos. A documentação completa desta especificação consta no relatório técnico /Inazumi, 1990/.

3.3.1. - O Módulo Principal

O componente de mais alto nível da especificação corresponde a um módulo representando todo o processo implementado, e que no caso corresponde ao provedor de apresentação. Um módulo é uma unidade não executável, e que posteriormente deve ser decomposto em outros sub-módulos e/ou processos, procedimentos, e macros, estabelecendo-se os vários níveis de decomposição.

Este módulo foi definido como uma <ACTION MODULE> e a sua respectiva especificação na linguagem EPOS-S pode ser vista na Figura 3.4. É decomposto ("DECOMPOSITION-Part") no processo de apresentação, denominado Proc_Presentation, e em várias outras ações auxiliares que compõem o processo de apresentação como um todo.

```
ACTION MODULE MAIN_PRES .
DESCRIPTION :
PURPOSE : "
   Este módulo provê os serviços e protocolos definidos na
   Camada de Apresentação. O padrão utilizado é o
   descrito no documento ISO/DIS 8822 (Connection Oriented
   Presentation Service Definition) e ISO/DIS 8823
   (Connection Oriented Presentation Protocol
   Specification). Esta ação representa o módulo principal
   do processo de apresentação. " .
DESCRIPTIONEND
DECOMPOSITION :
   (/ PROC_PRESENTATION , PREDICADO_01 , PREDICADO_03 ,
      PREDICADO_04 , PREDICADO_05 , PREDICADO_07 ,
      PREDICADO_08 , PREDICADO_09 , GET_CONTEXT_TAB_CON /) .
IMPORT-ACTION : STDIO , MASTER_1 , MASTER_P , MASTER_S,
                MASTER_T , MASTER_2 , AMBIENTE,
                PROT_C6 , PROT_NTR .
REALIZATION : SOFTWARE IN 'C' SOURCE 'main.c' .
PROCESSED : MICROCOMPUTADOR .
ACTIONEND
```

Figura 3.4. - Módulo Principal do Processo de Apresentação

O módulo importa ("IMPORT-Part") ações que representam arquivos que devem ser incluidos pelos procedimentos pertencentes ao processo de apresentação. Estes arquivos se referem a :

- **Stdio** : Biblioteca Standard I/O da linguagem C.
- **Master_1** : Definições de estruturas de dados globais ao Sistema Didático, incluindo dos processos MMS e ACSE.
- **Master_P** : Definições das estruturas de dados do processo de apresentação.
- **Master_S** : Definições das estruturas de dados do processo de sessão.

- **Master_T** : Definições das estruturas de dados do processo de simulação da camada de transporte.
- **Master_2** : Definição da estrutura de dados do Bloco de Mensagens de Protocolo.
- **Ambiente** : Definições de estruturas de dados e constantes específicas do núcleo de tempo real.
- **Prot_C6** : Protótipos dos procedimentos que compõem o processo de apresentação.
- **Prot_NTR** : Protótipos das funções do núcleo de tempo real.

A Figura 3.5 ilustra a especificação da inclusão do arquivo "MASTER_P.INC". A categoria 'C_INCLUDE', para um objeto de projeto do tipo ACTION MODULE, representa especificamente para o gerador de código C-Composer um comando "#include" do pré-processador C.

```
ACTION MODULE MASTER_P .
DESCRIPTION :
PURPOSE : "
   Este módulo especifica o comando "#include" do
   pré-processador C. No caso é incluido o arquivo
   "MASTER_P.INC" que contêm as definições das estruturas
   de dados utilizadas no processo de apresentação. " .
CATEGORY : 'C_INCLUDE' .
DESCRIPTIONEND
REALIZATION : SOFTWARE SOURCE 'master_p.inc' .
PROCESSED : MICROCOMPUTADOR .
ACTIONEND
```

Figura 3.5. - Especificação da Inclusão do Arquivo "MASTER_P.INC"

Note-se que na especificação deste módulo principal (Figura 3.4) faz-se a declaração de que a linguagem de programação a ser utilizada para a realização do software ("REALIZATION-Part") é a linguagem C e o nome do arquivo fonte gerado será "main.c". A estrutura hierárquica deste módulo pode ser vista na Figura 3.22.

3.3.2. - O Processo de Apresentação

O Processo de apresentação é quem efetivamente provê todos os serviços pertencentes a unidade funcional "*kernel*" de apresentação. A estrutura deste processo consiste em receber uma mensagem de alguma porta de entrada, executar o tratamento da mensagem, eventualmente um tratamento de erro, e finalmente enviar uma mensagem de resposta para uma porta de saída. A Figura 3.6 ilustra o diagrama de blocos do processo de apresentação.

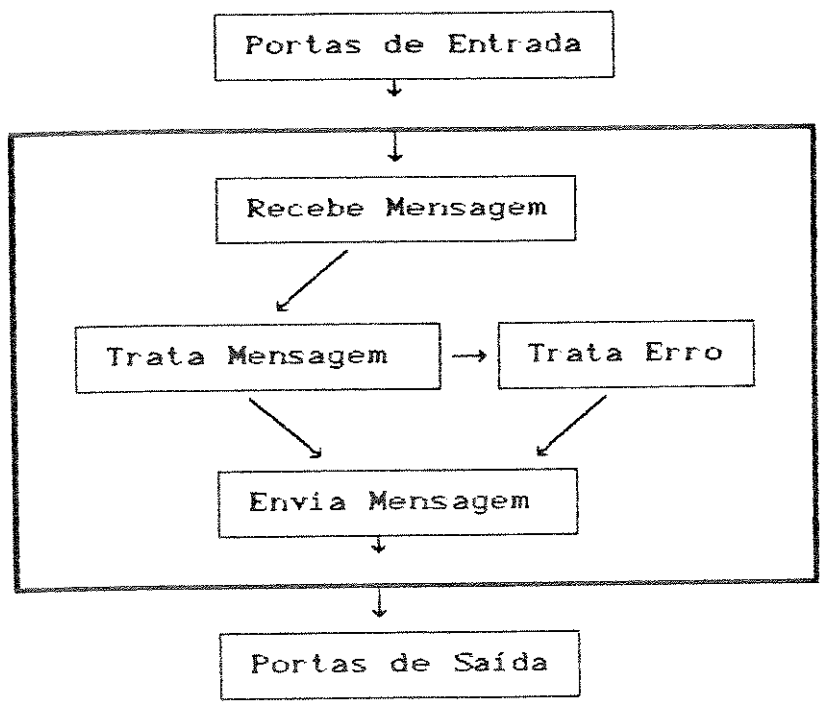


Figura 3.6. - Diagrama de Blocos do Processo de Apresentação

O corpo principal deste processo consiste em fazer varredura ("*polling*") nas portas de entrada dos processos com que troca mensagens, e se existirem mensagens, consumi-las e executar o seu devido tratamento. No caso do sistema SISDI-MAP, o processo de apresentação se comunica com os processos MMS (mbi_mms), ACSE (mbi_acse) e de Sessão (mbi_ses), como ilustra a Figura 3.7.

O procedimento de varredura destas portas de entrada utiliza um temporizador (ítem 3.2.4) visando evitar esperas infinitas por

mensagens. Não existe prioridade na recepção das mensagens enviadas pelos processos à apresentação, pois todas as portas são verificadas com o mesmo valor do temporizador, e esta verificação é executada de forma cíclica. Assim, todas as mensagens são recebidas e tratadas com igual prioridade, de acordo apenas com a ordem de chegada em cada porta de comunicação.

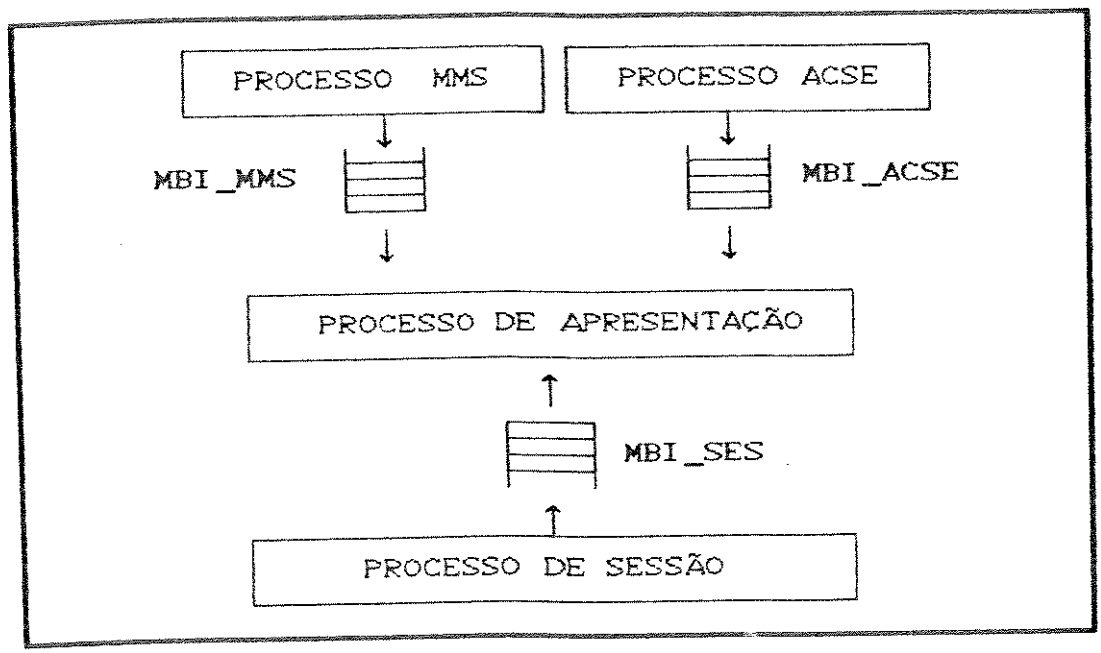


Figura 3.7. - Portas de Entrada do Processo de Apresentação

O processo de apresentação foi especificado como uma <ACTION TASK>, de modo a representar uma tarefa para o núcleo de tempo real. É decomposto em várias outras sub-ações, onde são realizadas ações simples para a declaração das variáveis, temporizadores e estruturas pertencentes ao escopo deste processo (ação esta denominada DECLARE_VARIABLES_LOCAIS); chamada ao procedimento INITIALIZE_TABLE para efetuar a inicialização da estrutura de dados PRES_TAB; e finalmente, ações que especificam o fluxo de controle do corpo principal do processo.

A respectiva especificação na linguagem EPOS-S deste processo pode ser vista na Figura 3.8, e a sua estrutura hierárquica e rede de petri através das Figuras 3.24 e 3.25 respectivamente.

ACTION TASK PROC_PRESENTATION .
DESCRIPTION :
PURPOSE : "
Este processo representa o Provedor de Apresentação, no caso, voltado para o padrão MAP, onde estão definidos apenas os serviços pertencentes a Unidade Funcional KERNEL de Apresentação. Os serviços pertencentes a esta Unidade Funcional são :

> P_CONNECT
> P_RELEASE
> P_U_ABORT
> P_P_ABORT e
> P_DATA

Tais serviços são providos para o processo Proc_Acse (Protocolo ACSE) e para o processo Proc_Mms (Protocolo MMS) da camada de APLICAÇÃO.
Para prover tais serviços de apresentação faz-se uso das primitivas de serviços equivalentes da camada de sessão através da solicitação ao processo de Sessão.
Parâmetros de Entrada :

> ingate_mms - Ponteiro para a porta de entrada do SASE-MMS
> ingate_acse - Ponteiro para a porta de entrada do CASE-ACSE
> ingate_ses - Ponteiro para a porta de entrada da camada de sessão
> outgate_mms - Ponteiro para a porta de saída do SASE-MMS
> outgate_acse - Ponteiro para a porta de saída do CASE-ACSE
> outgate_ses - Ponteiro para a porta de saída da camada de sessão
> outgate_op - Ponteiro para a porta de saída do processo P_OP

Parâmetros de Saída : Não há " .

```
NOTE : "
   Os elementos indicados no campo INPUT são definições de
   novos tipos abstratos de dados e de variáveis globais
   utilizadas pelo processo de apresentação." .
CATEGORY : 'Main-Action' .
DESCRIPTIONEND
DECOMPOSITION :
   DECLARE_VARIABLES_LOCAIS ;
   INITIALIZE_TABLE ;
   REPEAT INFINITE
   DO IF EXIST_MSG_FROM_ACSE THEN
         TREAT_ACSE
      FI ;
      IF EXIST_MSG_FROM_MMS THEN
         TREAT_MMS
      FI ;
      IF EXIST_MSG_FROM_SESSION THEN
         TREAT_SESSION
      FI
   OD .
TRIGGERED : SYSTEMSTART .
INPUT : BLOCK_MSG FROM INGATE_ACSE_INTERFACE ,
        BLOCK_MSG FROM INGATE_MMS_INTERFACE ,
        BLOCK_MSG FROM INGATE_SES_INTERFACE ,
        CON_NULL , P_CON_RESULT , REASON_REJECT ,
        PPDU_IDENTIFIER , ID_PPDU , ABORT_REASON ,
        REASON_ABORT , UINT8 , UINT16 , WARNING_ERROR ,
        INDEX_TAB , PRIMITIVE_TYPE , TYPE_PRIMITIVE ,
        DATA_PRIMITIVE , PRESENTATION_SERVICE ,
        PRESENTATION_SERVICE_NAME , SESSION_SERVICE ,
        SESSION_SERVICE_NAME , PRES_STATE_TYPE ,
        DATA_PRES_TAB_STATE , PRES_USER_TYPE , BLOCK_MSG ,
        PRES_CON_TAB .
ACTIONEND
```

Figura 3.8. - Especificação do Processo de Apresentação

O fluxo de controle do corpo principal do processo representa a varredura nas portas de comunicação. Esta varredura é cíclica e infinita, caracterizando um "*loop*" infinito, que é expresso através da estrutura de controle "REPEAT INFINITE". A varredura na porta de entrada do processo ACSE, por exemplo, é especificada por :

```
IF EXIST_MSG_FROM_ACSE THEN
   TREAT_ACSE
FI ;
```

onde "EXIST_MSG_FROM_ACSE" é um objeto de projeto do tipo CONDITION que descreve a condição de existência de mensagem na porta de entrada associada ao ACSE (Figura 3.9), e "TREAT_ACSE" o correspondente procedimento de tratamento caso exista mensagem.

```
CONDITION EXIST_MSG_FROM_ACSE .
DESCRIPTION :
PURPOSE : "
   Condição que verifica se existe alguma mensagem na
   porta de entrada do CASE-ACSE. Se houver obtêm a
   mensagem para um posterior tratamento, se não houver,
   aguarda por TIME_WAIT_GATE segundos pela chegada de uma
   mensagem. Se neste período de espera chegar uma
   mensagem, a mesma é consumida tornando a condição
   verdadeira (TRUE), caso contrário a condição será falsa
   (FALSE)."
DESCRIPTIONEND
CODE : "
   (espera_mens(mbi_acse,time_wait_gate,CONSOME,&block_msg)
    == OPERACAO_OK) " .
CONDITIONEND
```

Figura 3.9. - Condição de Existência de Mensagens do ACSE

Para descrever a condição de existência de mensagens nas portas de

comunicação fez-se uso do serviço "ESPERA-MENS" do núcleo de tempo real. A Figura 3.9 ilustra a especificação da condição referente ao ACSE, especificação esta onde o próprio código em linguagem C ("CODE-Part") foi inserido.

3.3.3. - Especificação das Portas de Comunicação

Especificaram-se as portas de comunicação utilizando-se os objetos de projeto do tipo INTERFACE. Neste caso, estes objetos representam interfaces "lógicas" entre os processos comunicantes e não interfaces físicas de hardware. A título de ilustração, apresenta-se através da Figura 3.10, a especificação da porta de entrada associada entre os processos de apresentação e ACSE, porta esta denominada INGATE_ACSE_INTERFACE.

```
INTERFACE INGATE_ACSE_INTERFACE .
DESCRIPTION :
PURPOSE : "
   Esta interface representa uma porta de entrada
   (mail-box) para o processo Proc_Pres e através da qual
   são obtidos dados (via mensagens) do processo Proc_Acse
   (CASE-ACSE). Estes dados são relacionados com as
   primitivas "request" e/ou "response" dos serviços
   P_CONNECT, P_U_ABORT e P_RELEASE."
NOTE : "
   Esta é uma interface lógica entre o processo de
   apresentação Proc_Pres e processo Proc_Acse (CASE-ACSE)
   da camada de aplicação."
DESCRIPTIONEND
EXTERNAL .
INTERFACEEND
```

Figura 3.10. - Porta de Comunicação de Entrada do ACSE

3.3.4. - TRATAMENTO DAS MENSAGENS DO ACSE

O processo de apresentação, ao receber uma mensagem do processo ACSE, chama a rotina TREAT_ACSE (Figura 3.11), que executa o respectivo tratamento da mensagem.

```
ACTION PROCEDURE TREAT_ACSE .
DESCRIPTION :
PURPOSE : "
   Este procedimento trata as mensagens provenientes do
   processo Proc_Acse (CASE-ACSE) pertencente a camada de
   aplicação.
   Tais mensagens podem indicar a solicitação de uma das
   seguintes primitivas de serviços providas pela camada
   de apresentação :
       > P_CONNECT request/response (+/-)
       > P_RELEASE request/response (+/-)
       > P_U_ABORT request
   Parâmetros de Entrada : Não há
   Parâmetros de Saída   : Não há ".
DESCRIPTIONEND
DECOMPOSITION :
   SET_DATA_PRIMITIVE ;
   SWITCH DATA_PRIMITIVE IN
      CASE REQUEST : TREAT_ACSE_REQUEST
      CASE RESPONSE_POS : TREAT_ACSE_RESPONSE
      CASE RESPONSE_NEG : TREAT_ACSE_RESPONSE
      OUT : CALL_ERROR USING ( WARNING_ERROR =
            ERROR_ACSE_100 )
   NI .
ACTIONEND
```

Figura 3.11. - Especificação do Procedimento TREAT_ACSE

O processo ACSE pode solicitar (ACSE iniciador) e responder (ACSE respondedor) pedidos para o estabelecimento e liberação (normal ou abrupta) de uma conexão de apresentação, via a porta de comunicação MBI_ACSE. Assim, as seguintes primitivas de serviços de apresentação podem ser solicitadas pelo :

- ACSE iniciador : P_CONNECT *request*,
  P_RELEASE *request*, e
  P_U_ABORT *request*.
- ACSE respondedor : P_CONNECT *response* (+/-), e
  P_RELEASE *response* (+/-).

O procedimento de tratamento inicialmente procura identificar o serviço solicitado. O primeiro passo consiste em identificar o tipo da primitiva do serviço, que pode ser *request*, *response(+)* ou *response(-)*. Se o tipo da primitiva for diferente destes um procedimento de tratamento de erros (CALL_ERROR) é ativado, caso contrário, outro procedimento adequado ao tipo da primitiva é chamado, como ilustra a Figura 3.11.

Uma vez identificado o tipo da primitiva do serviço, o próximo passo consiste em identificar o tipo de serviço associado, que pode ser P_CONNECT, P_RELEASE ou P_U_ABORT, dependendo do tipo da primitiva. A Figura 3.12 mostra o procedimento de identificação do serviço para uma primitiva de serviço do tipo *request*.

Uma vez identificado o tipo da primitiva de serviço e o tipo de serviço, efetua-se o seu respectivo tratamento. Por exemplo, o tratamento da primitiva de serviço P_CONNECT *request* é ilustrado na Figura 3.13. Neste tratamento, se já não existe uma conexão de apresentação com o mesmo identificador ("Connection_Id"), e o número de conexões por entidade de apresentação não estiver excedido, executa-se a ação especifica 03 e o envio de uma PPDU CP para a apresentação remota. Caso contrário, é enviada uma primitiva de serviço P_CONNECT *confirm negative* para o usuário local indicando a não aceitação do pedido de estabelecimento da conexão pelo provedor de apresentação.

```
ACTION PROCEDURE TREAT_ACSE_REQUEST .
DESCRIPTION :
PURPOSE : "
   Tratar as primitivas de servicos do tipo "REQUEST"
   solicitadas pelo processo Proc_Acse (CASE-ACSE) da
   camada de aplicacao.
   Tais servicos solicitados podem ser :
      > P_CONNECT request
      > P_RELEASE request
      > P_U_ABORT request
   Parâmetros de Entrada : Não há
   Parâmetros de Saída   : Não há " .
DESCRIPTIONEND
DECOMPOSITION :
   SET_PRESENTATION_SERVICE_NAME ;
   SWITCH PRESENTATION_SERVICE_NAME IN
      CASE P_CONNECTION : TREAT_P_CONNECTION_REQUEST
      CASE P_RELEASE : TREAT_P_RELEASE_REQUEST
      CASE P_U_ABORT : TREAT_P_U_ABORT_REQUEST
      OUT : CALL_ERROR USING ( WARNING_ERROR =
            ERROR_ACSE_105 )
   NI .
ACTIONEND
```

Figura 3.12. - Especificação do Procedimento TREAT_ACSE_REQUEST

Observe-se que o procedimento "Treat_P_Connection_Request" cumpre (FULFILS) parcialmente (PARTLY) com os requisitos 10(0) (REQUIREMENT 10 (0)) e 10(1) (REQUIREMENT 10 (1)) especificados no projeto conceitual.

```
ACTION PROCEDURE TREAT_P_CONNECTION_REQUEST .
DESCRIPTION :
PURPOSE : "
   Tratar o serviço P_CONNECT.request solicitado pelo
```

```
   processo Proc_Acse (CASE-ACSE) da camada de aplicação.
   O tratamento consiste em enviar uma PPDU CP (Connect
   Presentation) para a apresentação remota, e para tal
   faz-se uso do serviço S_CONNECT.request da camada de
   sessão.
   Parâmetros de Entrada : Não há
   Parâmetros de Saída   : Não há " .
FULFILS : REQUIREMENT 10 ( 0 ) PARTLY ,
          REQUIREMENT 10 ( 1 ) PARTLY .
DESCRIPTIONEND
DECOMPOSITION :
   GET_INDEX_CONNECTION_ID ;
   IF NOT NOT_EXIST_CONNECTION_ID THEN
      TREAT_INVALID_INTERSECTIONS_USER USING (
         WARNING_ERROR = ERROR_ACSE_115 )
   ELSE
      GET_INDEX_CONNECTION_NULL ;
      IF NOT_EXIST_CONNECTION_ID THEN
           SET_PRIMITIVE_CONFIRM_NEG ;
           SET_REASON_PROVIDER_REJECTION ;
           SEND_P_CONNECTION_CONFIRM USING (
              TYPE_PRIMITIVE , REASON_REJECT )
       ELSE
           IF IS_PREDICADO_05_AND_09 THEN
              ACAO_ESPECIFICA_03 ;
              SEND_PPDU_CP ;
              SET_TAB_CONNECTION_REQUEST
           ELSE
              TREAT_INVALID_INTERSECTIONS_USER USING (
                 WARNING_ERROR = ERROR_ACSE_120 )
           FI
       FI
    FI .
ACTIONEND
```

Figura 3.13. - Especificação do Procedimento TREAT_P_CONNECTION_REQUEST

3.3.5. - TRATAMENTO DAS MENSAGENS DO MMS

O processo de apresentação, ao receber uma mensagem do processo MMS, chama a rotina TREAT_MMS, que executa o respectivo tratamento da mensagem recebida. O processo MMS pode solicitar pedidos para a transferência normal de dados dentro de uma conexão de apresentação, via a porta de comunicação MBI_MMS. Assim, o processo MMS somente pode enviar primitivas de serviços do tipo P_DATA *request* ao processo de apresentação, pois este serviço não é confirmado.

O tratamento de uma primitiva P_DATA *request* consiste inicialmete em verificar se a transferência de dados está sendo solicitada dentro de uma conexão já estabelecida, onde neste caso, a Máquina de Protocolo pode estar nos estados STAac0, STAac1, STAac2 ou STAc0. Se estas condições forem satisfeitas, e os predicados associados a cada transição de estado forem verdadeiros procede-se o envio de uma PPDU TD para a entidade de apresentação remota, mantendo-se o mesmo estado da Máquina de Protocolo. A Figura 3.14 ilustra a especificação deste procedimento em EPOS-S.

```
ACTION PROCEDURE TREAT_P_DATA_REQUEST .
DESCRIPTION :
PURPOSE : "
   Tratar a solicitação da primitiva de serviço
   P_DATA.REQUEST efetuada pelo processo Proc_Mms
   (SASE-MMS) da camada de aplicação.
   Parâmetros de Entrada : Não há
   Parâmetros de Saída   : Não há  " .
FULFILS : REQUIREMENT 50 PARTLY .
DESCRIPTIONEND
DECOMPOSITION :
   GET_INDEX_CONNECTION_ID ;
   IF NOT_EXIST_CONNECTION_ID THEN
      TREAT_INVALID_INTERSECTIONS_USER USING (
```

```
          WARNING_ERROR = ERROR_MMS_210 )
   ELSE   SET_DATA_PRES_TAB_STATE ;
          SWITCH DATA_PRES_TAB_STATE IN
          CASE STAACO :
             IF IS_PREDICADO_03_AND_07 THEN SEND_PPDU_TD
             ELSE TREAT_INVALID_INTERSECTIONS_USER USING (
                    WARNING_ERROR = ERROR_MMS_215 )
             FI
          CASE STAAC2 :
             IF IS_PREDICADO_03_AND_07 THEN SEND_PPDU_TD
             ELSE TREAT_INVALID_INTERSECTIONS_USER USING (
                    WARNING_ERROR = ERROR_MMS_215 )
             FI
          CASE STAAC1 :
             IF IS_PREDICADO_08 THEN SEND_PPDU_TD
             ELSE TREAT_INVALID_INTERSECTIONS_USER USING (
                    WARNING_ERROR = ERROR_MMS_220 )
             FI
          CASE STACO :
             IF IS_PREDICADO_07 THEN SEND_PPDU_TD
             ELSE TREAT_INVALID_INTERSECTIONS_USER USING (
                    WARNING_ERROR = ERROR_MMS_225 )
             FI
          OUT :
             TREAT_INVALID_INTERSECTIONS_USER USING (
                WARNING_ERROR = ERROR_MMS_230 )
      NI
   FI .
INPUT : INDEX_TAB ,
        PRES_CON_TAB .
ACTIONEND
```

Fig. 3.14. - Especificação do Procedimento TREAT_P_DATA_REQUEST

3.3.6. - TRATAMENTO DAS MENSAGENS DA SESSÃO

O processo de apresentação, ao receber uma mensagem do processo de Sessão, chama a rotina TREAT_SESSION, que executa o respectivo tratamento da mensagem. O processo de Sessão envia para a porta de comunicação MBI_SES primitivas de serviços de sessão que transportam PPDUs provenientes da apresentação par. Além de PPDUs, também podem ser recebidos eventos do próprio provedor de sessão, como por exemplo uma primitiva de aborto S_P_ABORT. Assim, as seguintes primitivas de serviços de sessão são tratadas pelo processo de apresentação :

- S_CONNECT *indication* (PPDU CP)
- S_CONNECT *confirm*(+) (PPDU CPA)
- S_CONNECT *confirm*(-) (PPDU CPR)
- S_DATA *indication* (PPDU TD)
- S_U_ABORT *indication* (PPDU ARU ou ARP)
- S_RELEASE *indication*
- S_RELEASE *confirm*
- S_P_ABORT *indication*

De maneira análoga ao procedimento de tratamento de mensagens do ACSE, o primeiro passo consiste em identificar o serviço solicitado. O corpo principal deste procedimento é mostrado na Figura 3.15.

Uma vez identificado o tipo de serviço, o próximo passo consiste em verificar se a PPDU recebida está correta, pois a mesma pode ter recebido a inserção de um erro através do processo de simulação da camada de transporte no momento da comutação de nós da rede (simulação fim-a-fim da comunicação). Se a PPDU é inválida um procedimento de tratamento de erros é chamado, caso contrário efetua-se o tratamento da primitiva recebida provocando o envio de uma primitiva de serviço de apresentação do tipo *indication* ou *confirm* para o usuário local de acordo com o tipo de serviço recebido.

```
ACTION PROCEDURE TREAT_SESSION .
DESCRIPTION :
PURPOSE : "
   Este procedimento trata as mensagens provenientes do
   processo Proc_Ses de Sessão. Tais mensagens podem
   indicar uma das seguintes primitivas de serviços da
   camada de sessão :
             > S_CONNECT indication/confirm (+/-)
             > S_RELEASE indication/confirm
             > S_U_ABORT indication
             > S_P_ABORT indication
             > S_DATA indication
   Parâmetros de Entrada : Não há
   Parâmetros de Saída   : Não há " .
DESCRIPTIONEND
DECOMPOSITION :
   SET_DATA_PRIMITIVE ;
   SWITCH DATA_PRIMITIVE IN
      CASE INDICATION  : TREAT_SESSION_INDICATION
      CASE CONFIRM_POS : TREAT_SESSION_CONFIRM_POS
      CASE CONFIRM_NEG : TREAT_SESSION_CONFIRM_NEG
      OUT : CALL_ERROR USING ( WARNING_ERROR =
            ERROR_SESSION_300 )
   NI .
INPUT : BLOCK_MSG .
ACTIONEND
```

Figura 3.15. - Especificação do Procedimento TREAT_SESSION

3.3.7. - Tratamento de Erros

Nas tabelas de estado definidas nos documentos OSI, se nenhuma transição de estado é definida para um determinado evento de entrada, a intersecção estado/evento é dita inválida, e uma das seguintes ações é tomada :

● Se o evento de entrada está relacionado com a recepção de uma PPDU ou um evento do provedor de serviços de sessão, a PPM deve enviar uma PPDU ARP para o usuário remoto e uma primitiva de serviço P_P_ABORT *indication* para o usuário local. Esta ação foi especificada no procedimento TREAT_INVALID_INTERSECTIONS_PROVIDER como ilustra a Figura 3.16.

● Se o evento de entrada é proveniente do usuário de serviços de apresentação, qualquer ação tomada pela PPM é definida como uma "*local matter*". No caso desta implementação, o tratamento adotado consiste em emitir uma mensagem de erro para o usuário do sistema.

3.3.8. - Ações Terminais

A medida que os refinamentos dos procedimentos atingem um pequeno nível de complexidade, não se fazendo necessário mais decomposições, caracteriza-se o que se denomina uma "ação terminal". Nas especificações desta ações pode-se incluir o respectivo código fonte da implementação na linguagem de programação selecionada.

A título de ilustração, apresenta-se a especificação da ação "DECLARE_VARIABLES_LOCAIS", utilizada no corpo principal do processo de apresentação, através da Figura 3.17.

3.4. - A Geração de Código

A partir da especificação da implementação realizada na linguagem EPOS-S efetuou-se a geração automática de código utilizando-se a ferramenta C-Composer (Anexo C). As ações são trocadas por suas respectivas decomposições, iniciando-se das ações de mais alto nível, o fluxo de controle é substituido pela respectiva estrutura de controle da linguagem C e a ações terminais são incorporadas ao fluxo de controle de forma inalterada.

```
ACTION PROCEDURE TREAT_INVALID_INTERSECTIONS_PROVIDER .
DESCRIPTION :
PURPOSE : "
   Tratar as intersecções inválidas das tabelas de estado.
   Tais intersecções inválidas são provocadas por eventos
   provenientes da camada de sessão.
   Se nenhuma transição de estado é definida por um
   evento de entrada então a intersecção estado/evento é
   inválida, e a seguinte ação é tomada :
            > Se o evento de entrada está relacionado com
            a chegada de uma PPDU ou é um evento
            proveniente da camada de sessão, a PPM envia
            uma PPDU ARP e um P_P_ABORT.indication.
   Parâmetros de Entrada :
      > Reason - Razao do aborto
      > Ppdu_Id - Se a razão do aborto foi provocada pela
      chegada de um PPDU, este parâmetro identifica a ppdu
      que originou o aborto.
   Parâmetros de Saída : Não há. " .
CATEGORY : 'Error-Action' .
DESCRIPTIONEND
DECOMPOSITION :
   GET_INDEX_CONNECTION_ID ;
   ABORT_CONNECTION_LOCAL USING ( INDEX_TAB , REASON ,
      PPDU_ID ) ;
   SET_REMOTE_ABORT_CONNECTION ;
   SEND_PPDU_ARP USING ( REASON , PPDU_ID ) .
INPUT : REASON , PPDU_ID .
ALT : REASON = REASON_ABORT , PPDU_ID = ID_PPDU .
ACTIONEND
```

Figura 3.16. - Especificação do Procedimento TREAT_INVALID_INTERSECTIONS_PROVIDER

```
ACTION DECLARE_VARIABLES_LOCAIS .
DESCRIPTION :
PURPOSE : "
    Esta ação consiste em declararar e se necessário
    inicializar as variáveis locais ao programa principal
    do processo de apresentação. A principal inicialização
    diz respeito as portas de entrada e de saída
    relacionadas com a camada de apresentação. " .
 DESCRIPTIONEND
 CODE : "
    int     time_wait_gate = 1;
    mbi_mms   = ingate_mms;
    mbi_acse  = ingate_acse;
    mbi_ses   = ingate_ses;
    mbo_mms   = outgate_mms;
    mbo_acse  = outgate_acse;
    mbo_ses   = outgate_ses;
    mbo_op    = outgate_op;   " .
 ACTIONEND
```

Figura 3.17. - Especificação do Procedimento DECLARE_VARIABLES_LOCAIS

3.4.1. - O Processo Proc_Presentation

Selecionou-se como objeto inicial para efetuar a geração de código a ACTION MODULE MAIN_PRES (Figura 3.4). Este módulo representa o maior nível de abstração da especficação proposta e engloba o processo de apresentação. A Figura 3.18 ilustra a tradução para a linguagem C da ACTION TASK PROC_PRESENTATION (Figura 3.8).

A estrutura de controle "REPEAT INFINITE" foi traduzida para um comando "for (; ;)" em linguagem C e representa o "*polling*" infinito nas portas de entrada dos processos com quem troca informações. A ação "DECLARE_VARIABLES_LOCAIS" (Figura 3.17) e a

condição "EXIST_MSG_FROM_ACSE" (Figura 3.9) foram transladadas de forma inalterada para o programa fonte C, pois foram especificadas com uma "CODE-Part". Já as ações "INITIALIZE_TABLE", "TREAT_ACSE", "TREAT_MMS" e "TREAT_SESSION" foram transformadas em chamadas de procedimento, pois foram definidas como objetos de projeto do tipo ACTION com o atributo <PROCEDURE>.

```
main(argc, argv)    /*  PROC_PRESENTATION  */
int argc;
char *argv[];
{  int     time_wait_gate = 1;
   mbi_mms   = ingate_mms;
   mbi_acse  = ingate_acse;
   mbi_ses   = ingate_ses;
   mbo_mms   = outgate_mms;
   mbo_acse  = outgate_acse;
   mbo_ses   = outgate_ses;
   mbo_op    = outgate_op;
   INITIALIZE_TABLE ();
   for ( ; ; )
   {  if (espera_mens(mbi_acse,time_wait_gate,CONSOME,
            &block_msg) == OPERACAO_OK
      {  TREAT_ACSE ();
      }
      if (espera_mens(mbi_mms,time_wait_gate,CONSOME,
            &block_msg) == OPERACAO_OK)
      {  TREAT_MMS ();
      }
      if (espera_mens(mbi_ses,time_wait_gate,CONSOME,
            &block_msg) == OPERACAO_OK)
      {  TREAT_SESSION ();
      }
   }
} /*  main  */
```

Figura 3.18. - O Processo de Apresentação

Todos os módulos da especificação que possuem a categoria "C_INCLUDE" são transformados em comandos do tipo "#include" do pré-processador C. Por exemplo, o objeto de projeto ACTION MODULE MASTER_P (Figura 3.5) é traduzido para o comando '#include "MASTER_P.INC"'.

Os objetos de projeto do tipo INTERFACE utilizados na especificação das portas de comunicação, como por exemplo INGATE_ACSE_INTERFACE (Figura 3.10), não geram nenhum código fonte, pois não são transformáveis para C pelo C-Composer, sendo simplesmente ignorados no momento da geração de código.

3.4.2. - PROCEDIMENTOS

Ações que possuem um atributo <PROCEDURE> são transformados em procedimentos da linguagem C. A Figura 3.19 ilustra a ACTION PROCEDURE "TREAT_ACSE" (Figura 3.11) que foi traduzida para o procedimento "TREAT_ACSE()" em linguagem C.

```
TREAT_ACSE()
{  DATA_PRIMITIVE = block_msg->primitive;
   switch (DATA_PRIMITIVE) {
         case REQUEST :
            TREAT_ACSE_REQUEST (); break;
         case RESPONSE_POS :
            TREAT_ACSE_RESPONSE (); break;
         case RESPONSE_NEG :
            TREAT_ACSE_RESPONSE (); break;
         default:
            CALL_ERROR ( ERROR_ACSE_100 );
      }
} /*  TREAT_ACSE  */
```

Figura 3.19. - Código do Procedimento TREAT_ACSE

Nesta transformação pode-se notar a tradução da estrutura SWITCH da linguagem EPOS-S para a estrutura de controle SWITCH-CASE da linguagem C. Este procedimento em particular não possui parâmetros, porém esta tradução de ações para procedimentos também gera os parâmetros formais dos mesmos, se existirem. Por exemplo, a ação TREAT_INVALID_INTERSECTIONS_PROVIDER (Figura 3.16) possui dois parâmetros de entrada : REASON e PPDU_ID. Estes parâmetros são especificados como objetos de projeto do tipo "DATA" com o atributo <FORMAL>, e no momento da geração do código fonte, são definidos como parâmetros formais do procedimento que foi especficado em seu escopo. A Figura 3.20 ilustra o código gerado para a ACTION PROCEDURE TREAT_INVALID_INTERSECTIONS _PROVIDER.

```
TREAT_INVALID_INTERSECTIONS_PROVIDER(REASON,PPDU_ID)
ABORT_REASON REASON;
PPDU_IDENTIFIER PPDU_ID;
{   INDEX_TAB = GET_CONTEXT_TAB_CON(block_msg->connection_id);
    ABORT_CONNECTION_LOCAL ( INDEX_TAB, REASON, PPDU_ID );
    pede_mem(sizeof(Block_struct),&block_msg);
    block_msg->connection_id = INDEX_TAB;
    SEND_PPDU_ARP ( REASON, PPDU_ID );
} /*  TREAT_INVALID_INTERSECTIONS_PROVIDER  */
```

Figura 3.20. - Código do Procedimento
TREAT_INVALID_INTERSECTIONS_PROVIDER

A geração completa do código fonte C do processo de apresentação consta no relatório técnico /Inazumi, 1990/.

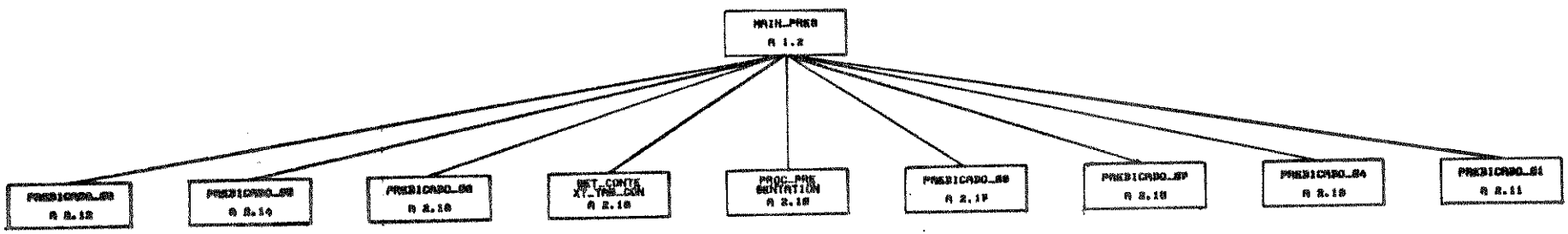
3.5. - DOCUMENTAÇÃO GRÁFICA

O ambiente EPOS, através da sua ferramenta EPOS-D, gera documentação gráfica baseada na especificação EPOS-S. Neste ítem apresenta-se uma coletânea da documentação gráfica gerada e que se baseia principalmente em Diagramas Hierárquicos, Redes de Petri e Fluxogramas.

As Figuras 3.21 e 3.22 representam o Diagrama Hierárquico e a respectiva Rede de Petri do módulo principal do sistema. Na Figura 3.21 observa-se a decomposição do Módulo Main_Pres no processo de apresentação e em ações auxiliares que o compõem. A Figura 3.22 representa a rede de petri correspondente ao módulo principal, onde no caso, é composto por um único processo, o proc_presentation.

As Figuras 3.23, 3.24 e 3.25 referem-se ao processo de apresentação proc_presentation. A Figura 3.23 ilustra, através de um diagrama hierárquico, os dois primeiros níveis de decomposição do processo de apresentação, e onde nota-se no primeiro nível os procedimentos TREAT_SESSION, TREAT_ACSE e TREAT_MMS que efetuam o tratamento das mensagens recebidas pelo processo. As Figuras 3.24 e 3.25 ilustram através de uma rede de petri e um fluxograma, respectivamente, o corpo principal do processo de apresentação. Nestas Figuras observa-se o ciclo de varredura infinita nas três portas de comunicação e as chamadas aos respectivos procedimentos de tratamento.

As Figuras 3.26, 3.27 e 3.28 referem-se ao procedimento TREAT_ACSE. A Figura 3.26 apresenta o diagrama hierárquico, com dois níveis de decomposição, deste procedimento de tratamento das mensagens provenientes do processo ACSE. As Figuras 3.27 e 3.28 representam respectivamente a rede de petri e o fluxograma do corpo principal do procedimento em questão.



UNICAMP	Principal Project: Apresentacao	EPOS84 V. 5.0.6
	Date: 11-10-90 User: P.Inazumi	

Figura 3.21. - Diagrama Hierárquico Principal

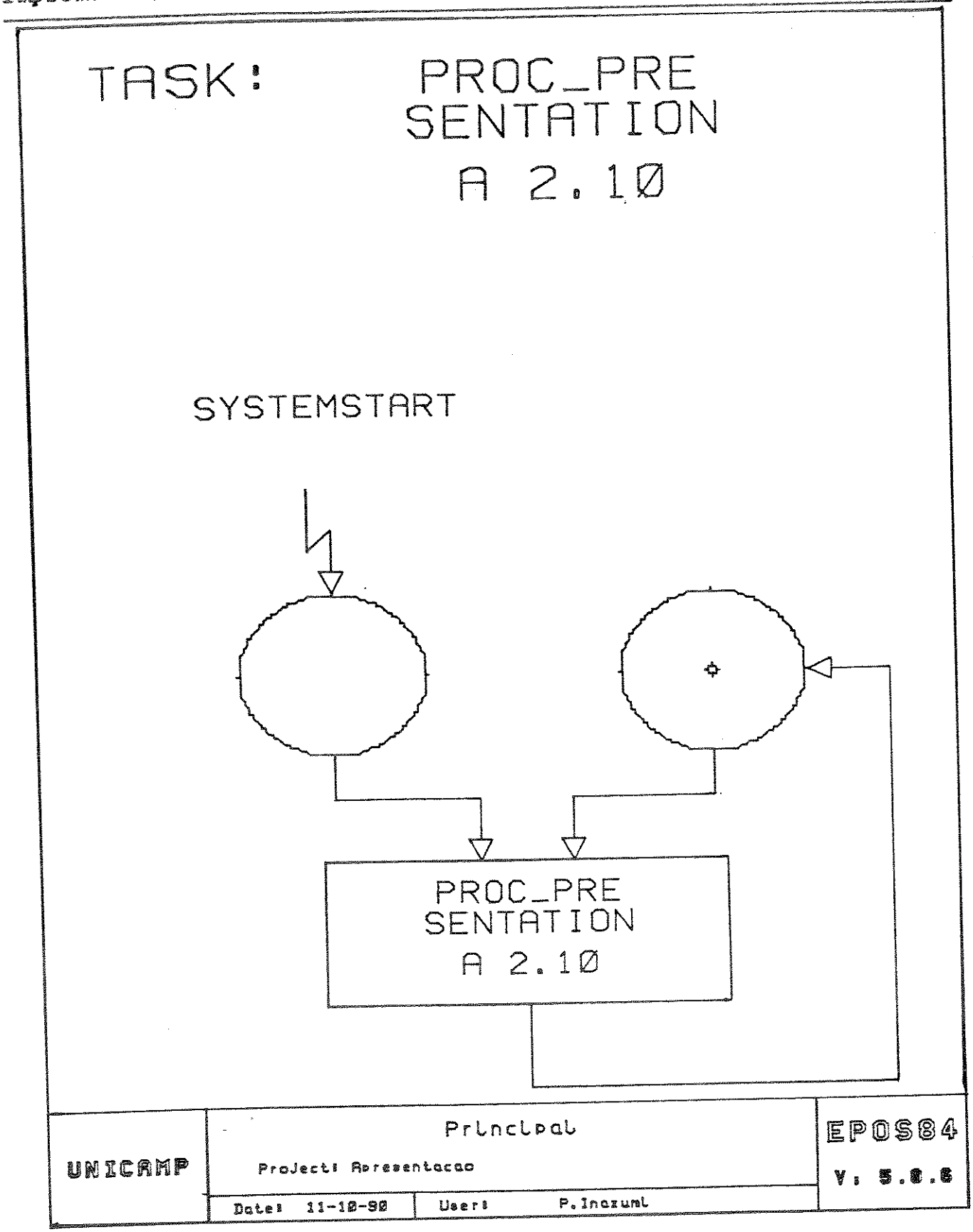


Figura 3.22. - Rede de Petri do Módulo de Apresentação

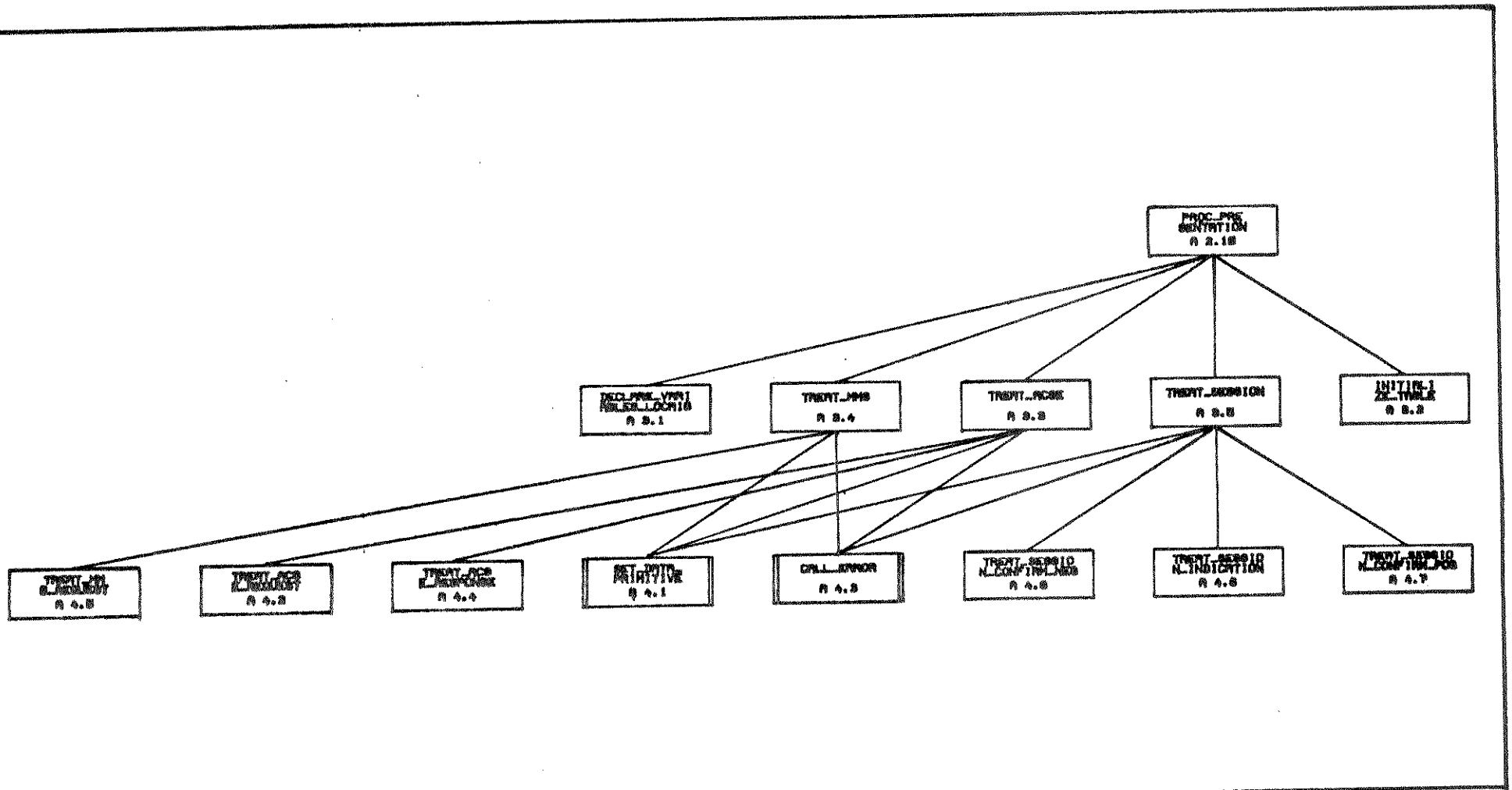


Figura 3.23. - Diagrama Hierárquico do Processo de Apresentação

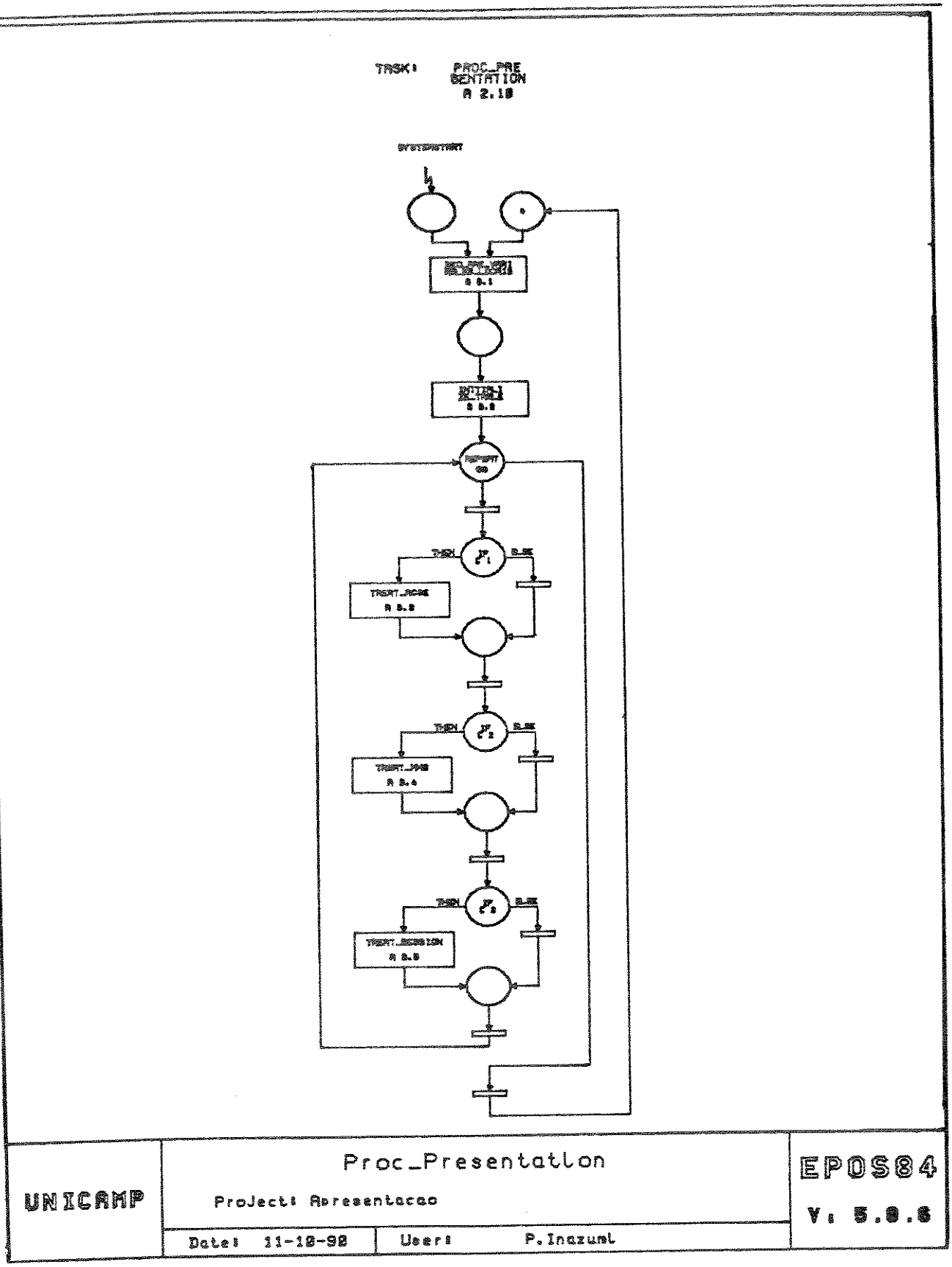


Figura 3.24. - Rede de Petri do Processo de Apresentação

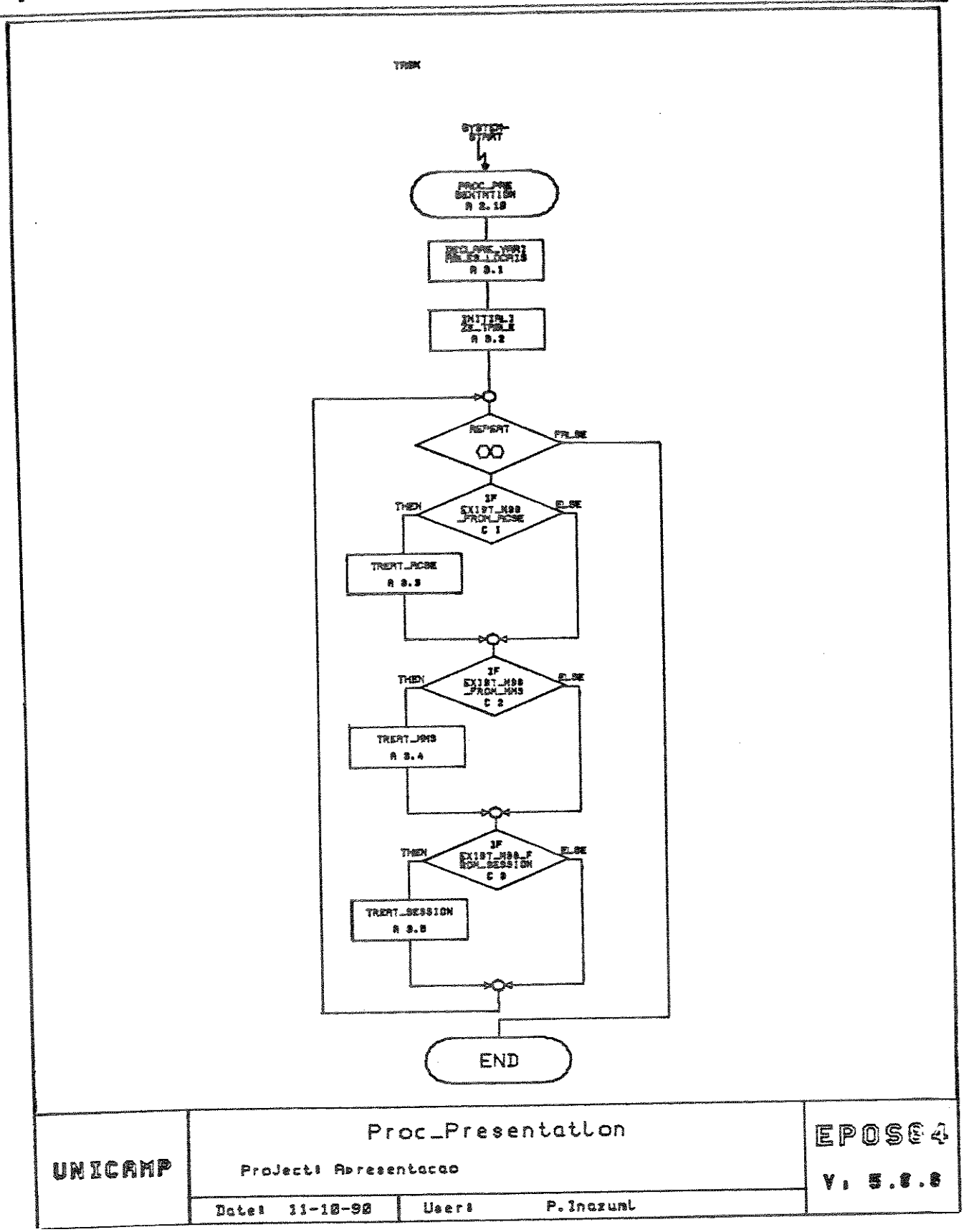


Figura 3.25. - Fluxograma do Processo de Apresentação

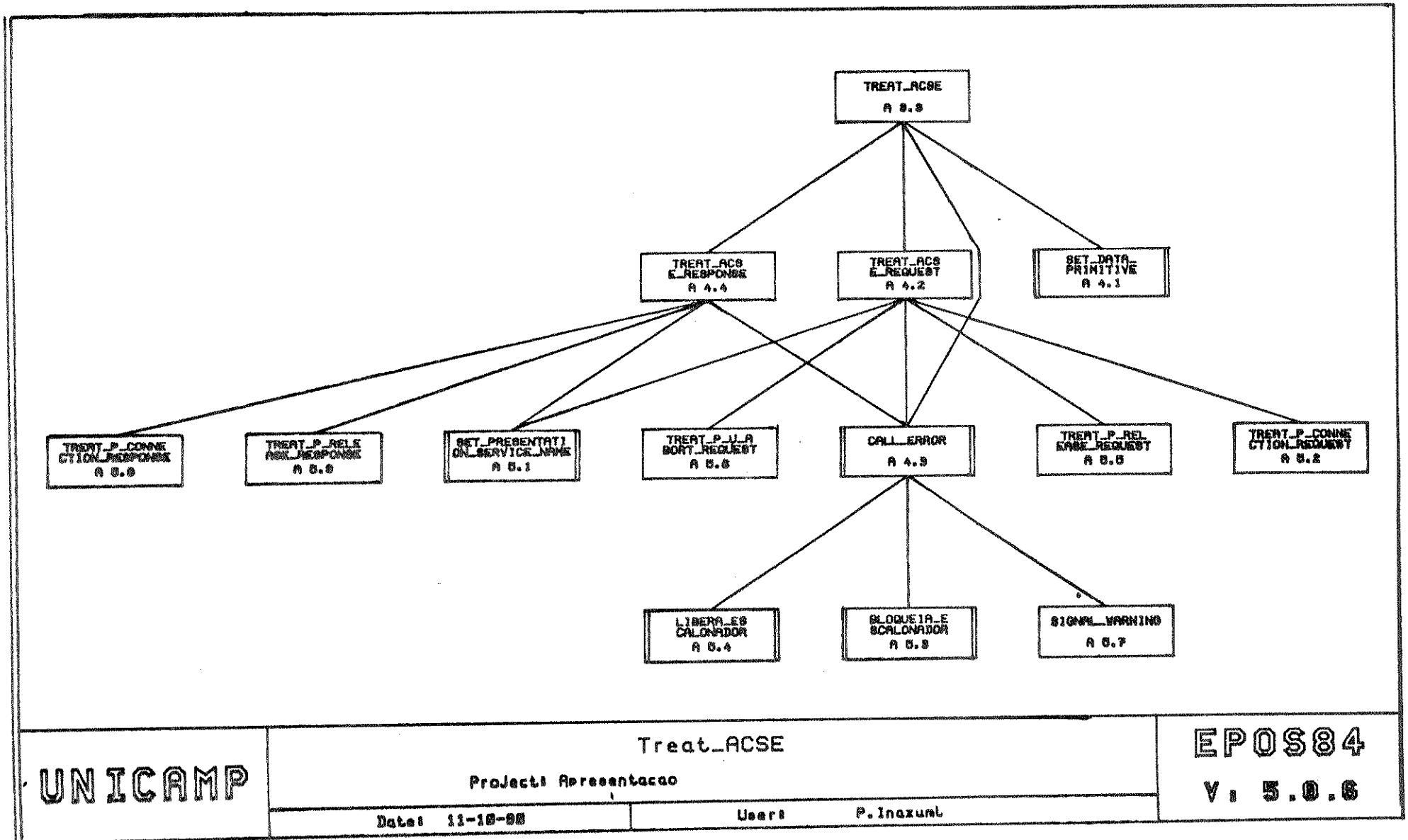
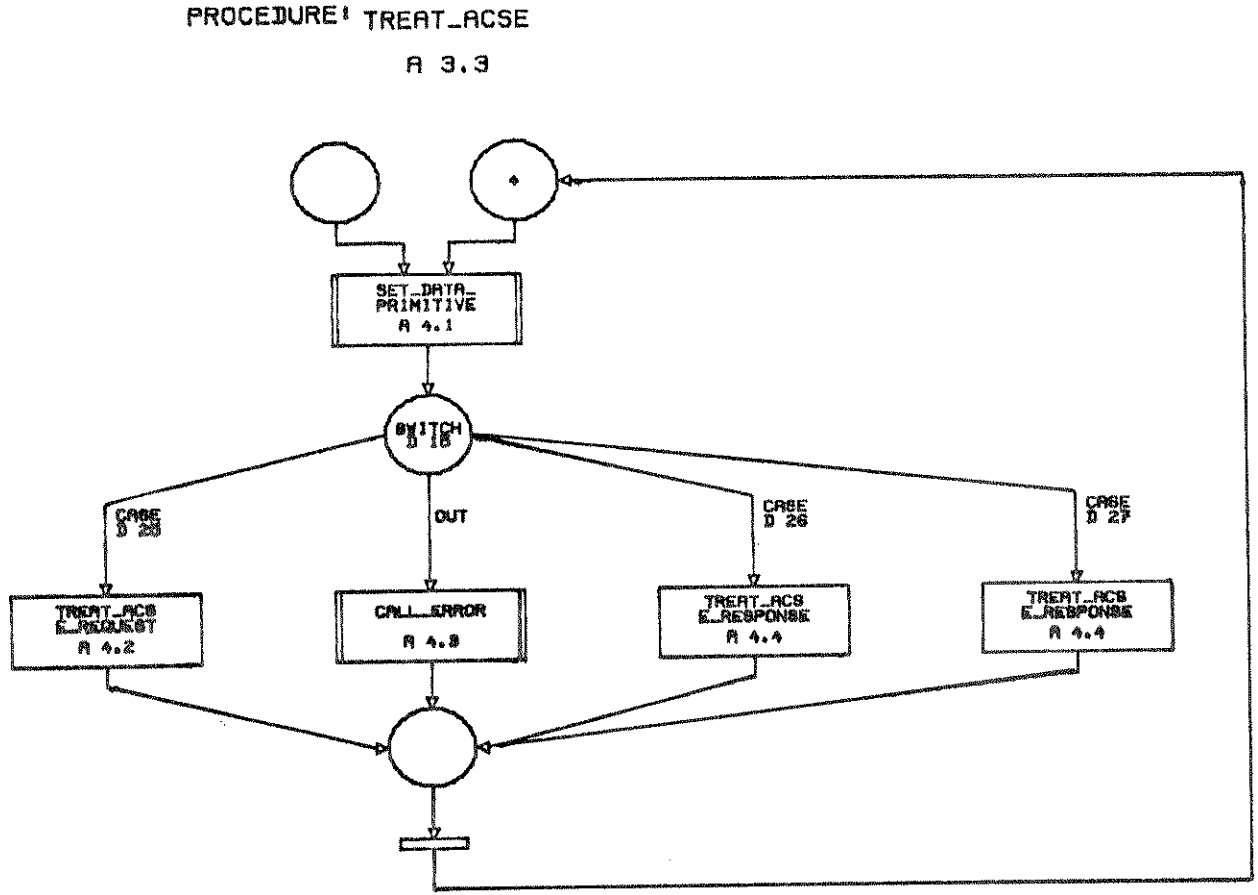


Figura 3.26. - Diagrama Hierárquico do Procedimento TREAT_ACSE



UNICAMP	Treat_Acse Project: Apresentacao	EPOS84 V: 5.0.6
	Date: 11-10-90 User: P.Inazumi	

Figura 3.27. - Rede de Petri do Procedimento TREAT_ACSE

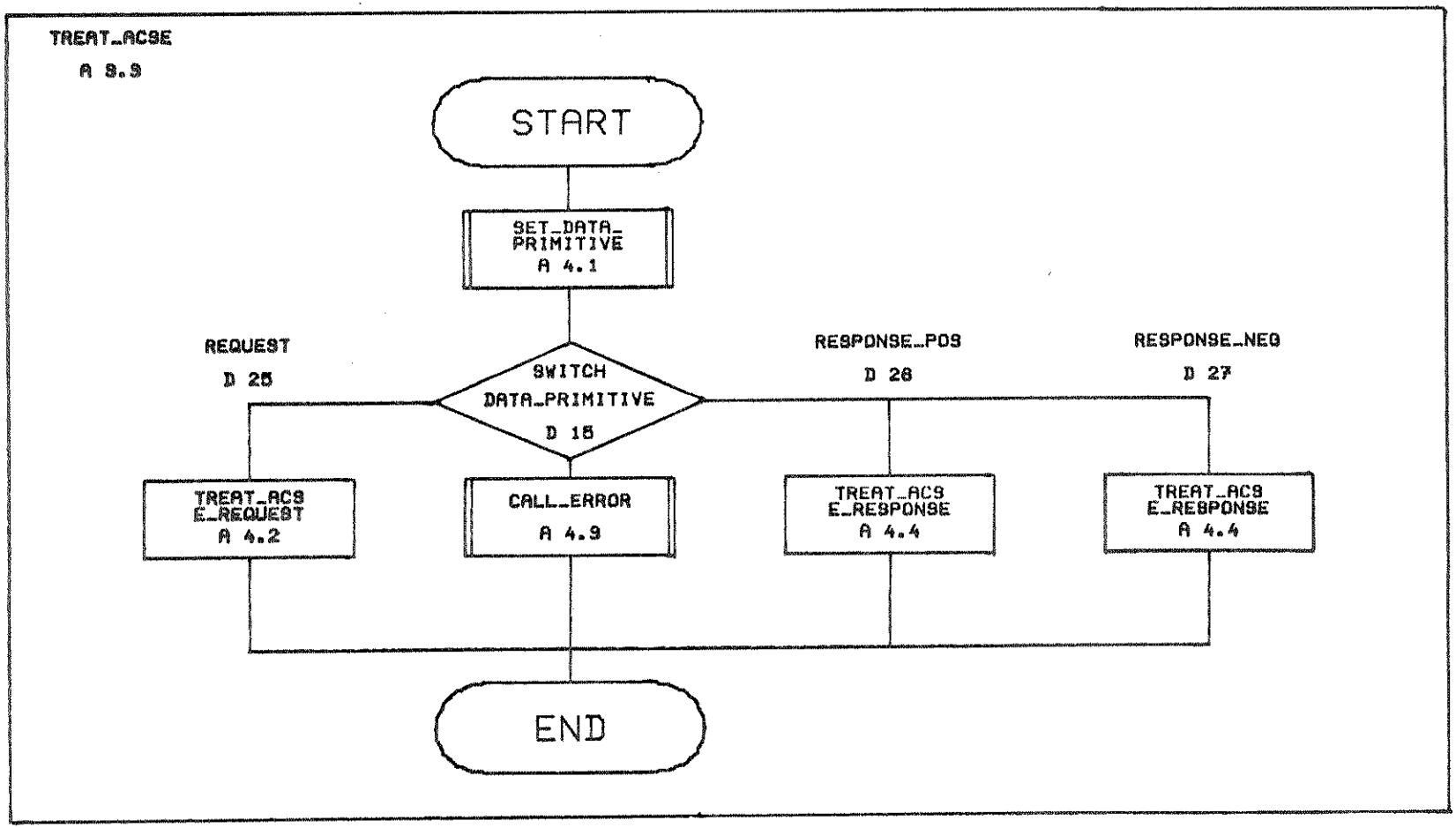


Figura 3.28. - Fluxograma do Procedimento TREAT_ACSE

CAPÍTULO IV - EXEMPLO DE EXECUÇÃO

4.0. - INTRODUÇÃO

Neste capítulo apresenta-se um exemplo de execução dos serviços de apresentação. Dentre os serviços pertencentes a unidade funcional *kernel* optou-se pelo serviço de estabelecimento de conexão (P_CONNECT) por ser um serviço confirmado e corresponder ao primeiro passo, referente a camada de apresentação, para que dois nós da rede possam trocar informações.

Procurou-se ilustrar o procedimento passo a passo para o estabelecimento de uma conexão, do ponto de vista das entidades de apresentação local e remota. Na medida do possível foi realizada uma analogia entre os procedimentos do protocolo de apresentação e os procedimentos do processo de apresentação que foram implementados.

4.1. - ESTABELECIMENTO DE CONEXÃO

Na configuração atual do SISDI-MAP, o processo ACSE é responsável pelo estabelecimento de associações da aplicação, logo é quem solicita o estabelecimento de uma conexão de apresentação ao processo de apresentação. Esta solicitação é feita através do envio de primitivas de serviços para a apresentação, via a porta MBI_ACSE, e da recepção de uma resposta via a porta MBO_ACSE. O processo de apresentação, por sua vez, faz solicitações aos serviços de sessão via a porta MBO_SES e recebe indicações e confirmações destes serviços via a porta MBI_SES.

A figura 4.0 ilustra as interações entre as primitivas da camada de apresentação e da camada de sessão envolvidas no procedimento

de estabelecimento de uma conexão de apresentação.

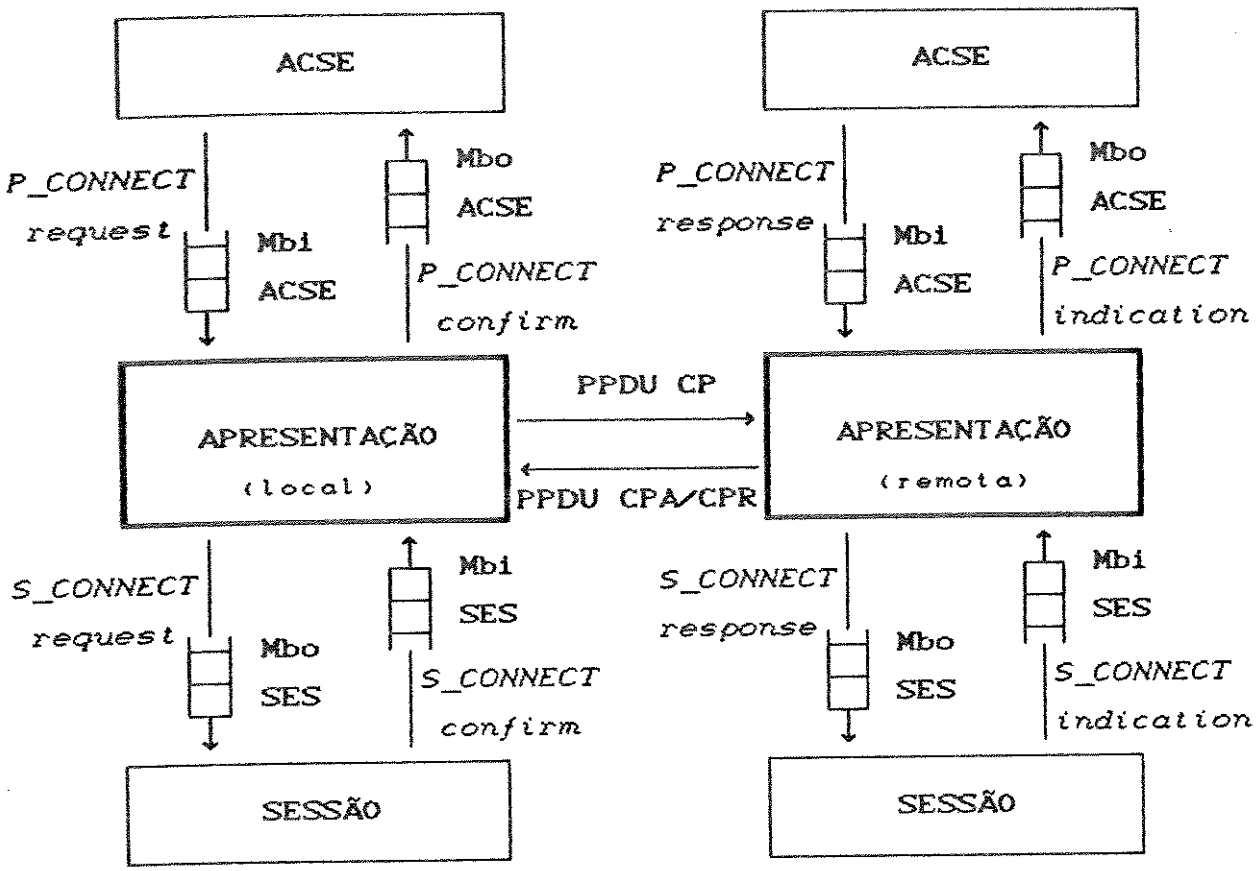


Figura 4.0. - Estabelecimento de uma Conexão de Apresentação

4.2. - PRIMITIVA "P_CONNECT REQUEST"

O processo ACSE solicita um pedido de estabelecimento de uma conexão ao processo de Apresentação. Esta solicitação se reflete no envio de uma primitiva de serviço P_CONNECT *request* para o processo de apresentação, via a porta de comunicação MBI_ACSE. Esta primitiva de serviço é representada por uma instância da estrutura de dados Bloco de Mensagens de Protocolo, onde estão representados os parâmetros da primitiva de serviço, e os dados do usuário (Figura 4.1).

Uma vez que esta mensagem é inserida na porta MBI_ACSE, o processo

de apresentação, durante a varredura ("*polling*") nas portas de comunicação, ao verificar que existe uma mensagem, obtêm o endereço do Bloco de Mensagens de Protocolo contido na porta, e passa para o respectivo tratamento chamando a rotina TREAT_ACSE.

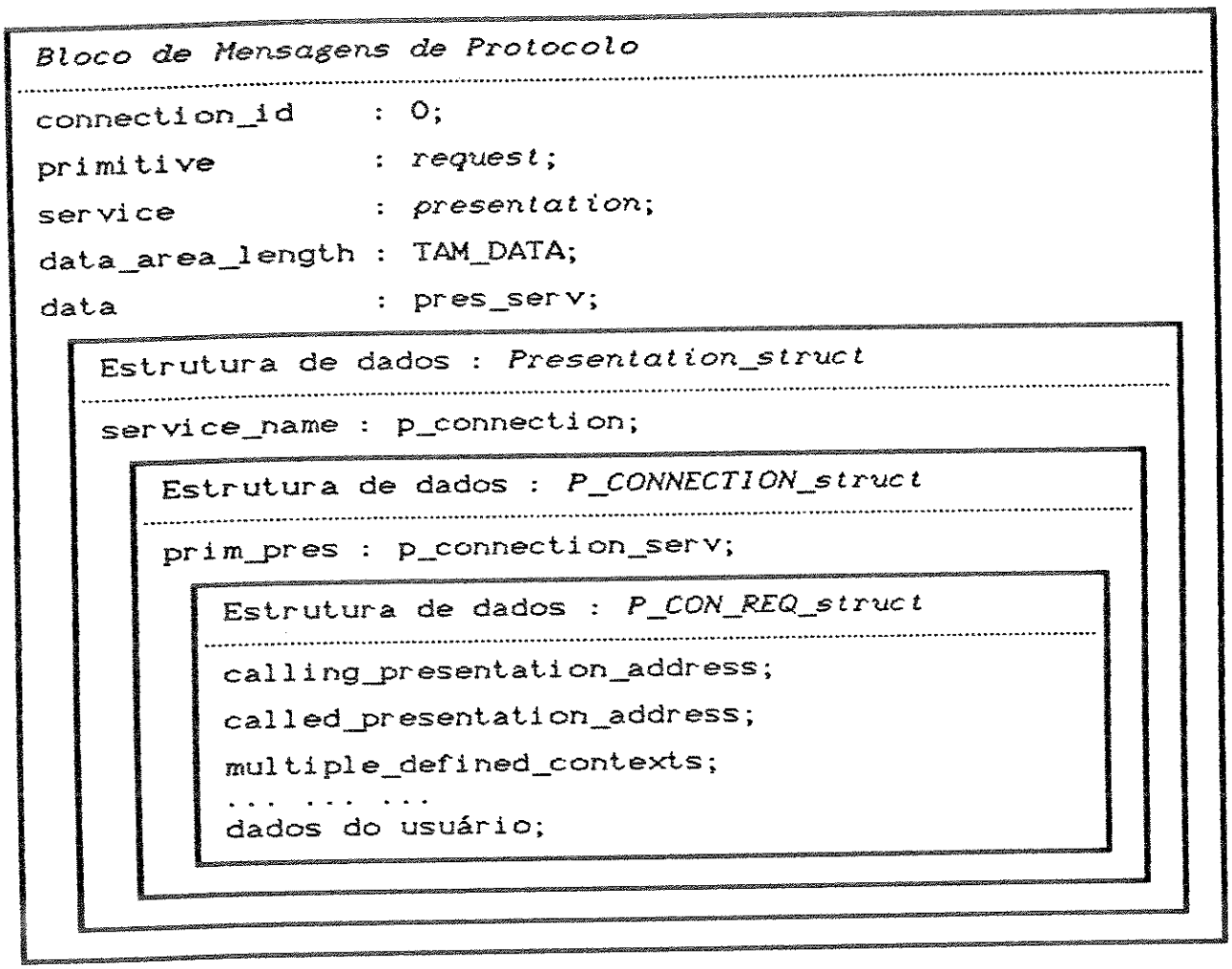


Figura 4.1. - Bloco de Mensagens de Protocolo para a Primitiva P_CONNECT Request

O tratamento inicial consiste em identificar o tipo da primitiva de serviço recebido, no caso *request*, e chamar a rotina TREAT_ACSE_REQUEST. Esta, por sua vez, identifica tratar-se do serviço P_CONNECT e então chama a rotina TREAT_P_CONNECTION _REQUEST, que finalmente executa o tratamento propriamente dito da primitiva P_CONNECT *request*.

4.3. - PPDU CP ("CONNECT PRESENTATION")

Neste momento uma nova instância do processo de apresentação deve ser criada para tratar da conexão solicitada. Se o número máximo de contextos (MAX_CONTEXT) já foi excedido, o processo de apresentação envia uma P_CONNECT *indication* para o processo ACSE indicando que não é capaz de suportar o pedido de estabelecimento da conexão. Caso contrário, armazena o status desta nova instância na estrutura PRES_TAB, que consiste no estado STAI1 (aguardando uma PPDU CPA) da máquina de protocolo, no identificador "connection_id" da conexão, e no tipo "origem" indicando tratar-se de uma entidade de apresentação local.

Finalmente, o tratamento da primitiva P_CONNECT *request* encerra-se no envio de uma PPDU CP para a entidade de apresentação remota contendo as informações e os parâmetros necessários para o estabelecimento da conexão. Para tal, chama-se a rotina SEND_PPDU_CP que preenche os parâmetros da primitiva S_CONNECT *request* da camada de sessão e passa como dados do usuário a PPDU CP. Após o preenchimento destes parâmetros no mesmo bloco de mensagens de protocolo recebido, este é inserido na porta de saída associado ao processo de sessão (MBO_SES) através da função "envia_mens" do núcleo de tempo real.

4.4. - PRIMITIVA "S_CONNECT INDICATION"

O processo de sessão ao receber a primitiva S_CONNECT *request* envia uma primitiva S_CONNET *indication* para o processo de apresentação remoto. O processo de apresentação, ao receber uma mensagem do processo de sessão via a porta MBI_SES, passa para o respectivo tratamento através da rotina TREAT_SESSION.

Ao identificar tratar-se de uma primitiva de serviço S_CONNECT *indication*, passa-se para o tratamento desta através da rotina

TREAT_S_CONNECTION_INDICATION. Inicialmente o tratamento consiste em verificar se a PPDU CP recebida é válida. Se não for, devido a inserção de erros pelo processo de simulação da camada de transporte, envia-se uma PPDU CPR para a entidade de apresentação origem indicando a rejeição do pedido de estabelecimento da conexão pelo provedor remoto, caso contrário, executa-se o tratamento da PPDU CP recebida.

Novamente uma instância do processo de apresentação, agora remoto, deve ser criada para tratar da conexão solicitada. Se o número máximo de contextos (MAX_CONTEXT) já foi excedido, o processo de apresentação envia uma PPDU CPR para a entidade de apresentação origem indicando que o provedor remoto não é capaz de suportar o pedido de estabebecimento da conexão. Caso contrário, armazena o "*status*" desta nova instância na estrutura PRES_TAB, que agora consiste no estado STAI1 (aguardando uma primitiva P_CONNECT *response*), no identificador "connection_id", e no tipo "destino" indicando tratar-se de uma entidade de apresentação remota. Finalmente, o processo de apresentação, através da rotina SEND_P_CONNECTION_INDICATION, envia uma primitiva P_CONNECT *indication* para o processo ACSE via a porta MBO_ACSE.

4.5. - PRIMITIVA "P_CONNECT RESPONSE"

O processo ACSE (remoto) ao receber a primitiva P_CONNECT *indication* executa os procedimentos adequados de seu protocolo e envia uma resposta referente a este pedido de estabelecimento de conexão. Para tal envia uma primitiva P_CONNECT *response positive* se aceitou o pedido de conexão, ou uma primitiva P_CONNECT *response negative* caso contrário. Estas primitivas de serviços são enviadas à camada de apresentação via a porta MBI_ACSE através de uma nova instância do bloco de mensagens de protocolo.

O processo de apresentação (remoto) ao receber esta mensagem chama a rotina TREAT_ACSE para efetuar o respectivo tratamento. Ao

identificar tratar-se de uma primitiva P_CONNECT *response* chama a rotina TREAT_P_CONNECTION_RESPONSE onde, se a máquina de protocolo está no estado STAI2 e recebe uma primitiva *response positive* então envia-se uma PPDU CPA indicando a aceitação do pedido de conexão e muda-se o estado da máquina de procotolo para conectado (STAc0), ou se receber uma primitiva *response negative* envia-se uma PPDU CPR indicando a não aceitação do pedido de conexão pelo usuário remoto e retorna a máquina de protocolo para o estado disconectado (STAI0).

4.6. - PPDU CPA ("CONNECT PRESENTATION ACCEPT")

O processo de apresentação envia uma PPDU CPA para a entidade de apresentação par indicando a aceitação do pedido de estabelecimento de uma conexão. Esta PPDU é enviada no campo "Dados do Usuário" da primitiva de serviço S_CONNECT *response positive* da camada de sessão. O procedimento SEND_PPDU_CPA é responsável pelo preenchimento dos parâmetros da primitiva S_CONNECT *response* e pelo envio desta mensagem para o processo de sessão, via a porta de saída MB0_SES, através da função "envia_mens" do núcleo de tempo real.

4.7. - PPDU CPR ("CONNECT PRESENTATION REJECT")

As únicas possibilidades para uma entidade de apresentação enviar uma PPDU CPR referem-se ao fato ou do provedor ou do usuário rejeitarem o pedido de estabelecimento de uma conexão. Em ambos os casos o procedimento SEND_PPDU_CPR é chamado recebendo como parâmetro qual das partes comunicantes recusou o pedido de estabelecimento da conexão, e que pode ser "rejeição pelo provedor" ou "rejeição pelo usuário". De maneira análoga a rotina SEND_PPDU_CPA, são preenchidos os parâmetros da primitiva S_CONNECT *response* e então enviada para o processo de sessão via a porta de saída MB0_SES.

4.8. - PRIMITIVA "S_CONNECT CONFIRM"

Finalmente, o processo de sessão ao receber uma primitiva S_CONNECT *response* (*positive/negative*) envia uma correspondente primitiva S_CONNECT *confirm* para o processo de apresentação par, transportando ou uma PPDU CPA ou uma PPDU CPR. O processo de apresentação ao receber a primitiva S_CONNECT *confirm* da camada de sessão, via a porta MBI_SES, chama a rotina TREAT_SESSION para executar o devido tratamento.

Se a máquina de protocolo está no estado de aguardando uma PPDU CPA (STAI1) e foi recebida uma primitiva S_CONNECT *confirm positive*, o tratamento consiste em enviar uma correspondente P_CONNECT *confirm positive* e passar a máquina de protocolo para o estado conectado (STAc0), ou se foi recebida uma primitiva S_CONNECT *confirm negative*, enviar uma primitiva P_CONNECT *confirm negative* e retorna-se a máquina de protocolo para o estado disconectado (STAI0).

CAPÍTULO V - CONCLUSÕES

5.0. - A IMPLEMENTAÇÃO

Para a realização desta implementação fez-se necessário um estudo profundo dos documentos da camada de apresentação de modo a sanar as ambiguidades de interpretações. Após um investimento inicial de tempo no entendimento da especificação, a passagem para uma implementação implicou na definição dos ítens descritos como "*local matter*" nos documentos e em propor um modelo de implementação, no caso utilizando-se "*mail-boxes*" e baseado no envio e recepção de mensagens.

Outro ítem a ser observado se refere ao fato da existência de uma única instância do processo de apresentação. No caso do sistema SISDI-MAP, onde tem-se como objetivo propiciar mecanismos para o entendimento do ambiente OSI, tal opção mostrou-se suficiente, pois a camada de apresentação implementada não possui um protocolo muito complexo, e a estrutura de dados de gerenciamento interno PRES_TAB foi suficiente para a manipulação e gerenciamento correto das várias instâncias do processo.

A estrutura de dados Bloco de Mensagens de Protocolo mostrou-se pouco flexível a alterações, visto que a cada mudança na estrutura funcional do SISDI-MAP implica em alterações nas máscaras ("*dummy*") de cada processo do sistema. Estas alterações foram necessárias devido a troca do simulador da camada de apresentação pelo respectivo protocolo de apresentação e a inserção do protocolo de sessão e da simulação da camada de transporte e inferiores.

5.1. - O AMBIENTE EPOS

Uma grande vantagem do sistema EPOS se refere ao fato de ser um "ambiente integrado", proporcionando suporte em todo o ciclo de vida do software, abrangendo desde as fases da especificação do problema, da solução conceitual, da especificação da implementação (projeto) até as fases de geração de código e planejamento dos testes.

Observou-se que o tempo gasto inicialmente com a realização da especificação formal da implementação foi compensado nas fases seguintes de codificação e testes do software. Isto se deve ao fato de que a geração automática de código produz um código fonte sintáticamente correto, e se a especificação também é correta e completa, os erros de programação e a necessidade de alterações no código fonte gerado são minimizados.

Proporciona a geração automática de documentação a partir das especificações realizadas em EPOS-R e EPOS-S. Documentação esta que pode ser em forma textual através de relatórios ou listas de ítens; ou gráfica, como por exemplo, diagramas hierárquicos, diagramas nassi-shneiderman, redes de petri, flowcharts, etc.

5.1.1. - A LINGUAGEM EPOS-R

A especificação semi-formal do problema ajuda a expor na forma de um projeto básico em linguagem natural os objetivos e a descrição do protocolo, e que posteriormente pode ser incorporado à documentação final do projeto. Porém não possui meios para os testes das especificações informais em texto livre. Típicas são os elementos de requisitos identificáveis, onde apenas é verificado se os requisitos e/ou restrições foram mencionados na solução conceitual do projeto e não se a sua especificação é correta.

A linguagem de especificação EPOS-R, através do elemento formal

"Processo de Decisão", permite uma representação de todos os elementos relacionados as tabelas de estado descritas nos documentos ISO. E como estes documentos (Serviços e Protocolos) já estabelecem todos os requisitos (eventos, estados, ações, etc.) do protocolo, esta ferramenta nos permite converter todas as tabelas de estado definidas no protocolo em processos de decisão, e verificar sua consistência, redundância e completeza.

Esta conversão de tabelas de estado em processos de decisão se aplica principalmente a documentos onde ainda não estão definidas tais tabelas de estado, pois permite efetuar verificações quanto a consistência, redundância e completeza das eventuais tabelas que forem confeccionadas. Nos documentos ISO, onde tais tabelas já passaram por vários processos de padronização e assumindo estarem corretas, tal fato é útil para documentação, visto que se trata de um elemento formal, e para tal elimina ambiguidades de interpretação, e também como apresentação de uma solução conceitual para o protocolo.

5.1.2. - A LINGUAGEM EPOS-S

A linguagem EPOS-S permite que a especificação da implementação seja realizada por meio de um refinamento passo a passo do sistema, através da decomposição dos módulos e ações, e assim caracterizando um desenvolvimento "*top-down*". Durante o desenvolvimento da especificação da implementação, o fato de não se conhecer muito bem a função de um módulo, não impedirá o desenvolvimento normal do projeto, devido ao fato de que os módulos são definidos de forma independente no sistema, permitindo que em uma fase posterior possa especificar-se claramente o módulo em questão.

Esta especificação pode ser realizada de maneira independente da linguagem de programação, à exceção talvez das ações terminais, onde o próprio código da ação na linguagem de programação selecionada é inserido. Pode-se perceber que, quanto mais

detalhada é a especificação da implementação, menor e menos complexo resultam os códigos das ações terminais, e vice-versa.

Não existe o operador de atribuição. Assim, a mais simples atribuição de um valor para uma variável tem que ser uma ação (ACTION) simples, onde se especifica uma "CODE-Part" com a operação de atribuição desejada. Também não existe um atributo para ações que as caracterizem como funções, apenas como procedimentos.

5.1.3. - O GERADOR DE CÓDIGO C-COMPOSER

A geração automática de código em C a partir da especificação da implementação em EPOS-S não é total, pois existem elementos formais que não possuem uma correspondente implementação em linguagem C, como por exemplo os objetos de projeto DEVICE, INTERFACE e EVENT. Também não é otimizada, no sentido de que é difícil explorar vantagens e artifícios permitidos na linguagem C, como por exemplo, a conversão de tipos de dados ("*cast*"), a manipulação de variáveis com identificadores em letra maiúscula e minúscula, uma vez que todas são geradas em letras maiúsculas, etc.

O código gerado de forma automática representou uma grande porcentagem do código final da implementação, cerca de 75%. Estas se referem principalmente às estruturas de controle da linguagem C, como IF-THEN-ELSE, SWITCH-CASE, FOR, REPEAT, etc. e a definição de procedimentos. Os demais 25% referem-se à definição de estruturas de dados de apresentação que foram definidas no arquivo "MASTER_P", alterações em algumas estruturas do programa, como a mudança do cabeçalho do programa principal de "MAIN" para "Proc_Presentation", a inserção dos protótipos dos procedimentos implementados, etc.

Destes 75% do código final gerado, 56% correspondem a especificações de objetos de projeto com uma "CODE-Part", isto é,

que possuem uma codificação própria na linguagem C, e 44% de objetos de projeto com uma "DECOMPOSITION-Part". Porém, destes 56% especificados diretamente em linguagem C, uma grande parte corresponde a comandos de atribuição, referentes principalmente aos comandos de envio de uma PDU. Note-se que na transmissão de uma PDU é necessário o preenchimento dos parâmetros da primitiva de serviço que a transportará e da própria PDU.

5.2. - O AMBIENTE MULTITAREFA

Um fator a ser considerado é a portabilidade da implementação para outros sistemas operacionais, como por exemplo, o sistema operacional UNIX. Se não considerarmos os passos para a criação da tarefa e das portas de comunicação, a implementação realizada é dependente do núcleo de tempo real somente no recebimento e no envio de mensagens via as portas de comunicação e de funções de alocação de memória.

Uma vez recebida a mensagem, o tratamento independe de chamadas explicitas das funções do núcleo até o momento do envio de uma mensagem de resposta. Assim, seria necessário adaptar apenas estas funções de recepção/envio de mensagens e as solicitações de memória para as funções do novo sistema operacional.

5.3. - O FUTURO

Os próximos passos propostos para esta camada de apresentação são basicamente dois :

Primeiro, expandir os serviços além destes pertencentes a unidade funcional *"kernel"*, especialmente os serviços referentes ao gerenciamento de atividades e sincronização. Esta expansão não deve ser muito complexa visto que a implementação foi realizada de forma modular e aberta à agregação de novos serviços.

E segundo, inserção das funções referentes à CODIFICAÇÃO e DECODIFICAÇÃO de PDUs. Estas funções serão fornecidas por uma ferramenta /Restovic, 1990/ que está em desenvolvimento e que engloba além de um COMPILADOR para ASN.1, um gerador de código para as funções de codificação e decodificação das PDUs.

CAPÍTULO VI - REFERÊNCIAS BIBLIOGRÁFICAS

6.0. - REFERÊNCIAS BIBLIOGRÁFICAS

[01] - Barros, Ruy C.
Tese de mestrado em fase de conclusão - FEE - UNICAMP

[02] - Bartioli, Paul D.
"The Application and Presentation Layers of the Reference Model for Open Systems Interconnection". Proceedings of INFOCOM, 1983.

[03] - Biewald, J.; Joho, E.; Jovalekic, S.; Shelling, H.
"Application of the specification and design technique EPOS to a process control problem". Proc. IFAC/IFIP, 1980.

[04] - EPOS C-Composer, Manual "Code Generation"
Version 5.0, Release 0. GPP - Janeiro/1988.

[05] - EPOS-R, Manual "Specification Language"
Version 5.0, Release 0. GPP - Janeiro/1988.

[06] - EPOS-S, Manual "Specification Language"
Version 5.0, Release 0. GPP - Janeiro/1988.

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

[08] - GM - MAP, Versão 3.0
"Manufacturing Automation Ptotocol", Julho 1987.

[09] - Halsall, Fred.
"Data Communications, Computer Networks and OSI". Addison-Wesley Publishing Conpany. Workingham, England 1988.

[10] - Hollis, Lloyd L.
"OSI Presentation Layer Activities". Proceedings of the IEEE, Vol. 71, No 12, December 1983.

[11] - Inazumi, P.C.M.
Relatório Técnico referente a implementação da camada de apresentação do SISDI-MAP. FEE-UNICAMP, Novembro/1990.

[12] - ISO/DIS 8823
"Connection Oriented Presentation Protocol Specification". Junho/1986.

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

[14] - ISO/DIS 8825
"Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)". Agosto/1986

[15] - ISO/DP 7498
"Data Processing - Open Systems Interconnection - Basic Reference Model". Dezembro/1982.

[16] - ISO/DP 9506
"Manufacturing Message Specification - Protocol Specification". Maio/1987.

[17] - ISO/IS 8327
"Basic Connection Oriented Session Protocol Specification". Agosto/1987.

[18] - ISO/IS 8649
"Protocol Specification for the Association Control Service Element". Dezembro/1988.

[19] - Jacinto, D.C.A.
"Aspectos de especificação e implementação da estrutura de mensagens de um Sistema Didático do protocolo MMS".
Tese de mestrado - FEE - UNICAMP. Novembro/1989.

[20] - Lauber, R.J. e Lempp, P.R.
EPOS Overview. GPP and University of Stuttgart. Stuttgart, Setembro/1986.

[21] - Mendes, Manuel J.
"Comunicação Fabril e o projeto MAP/TOP", Escola Brasileiro Argentina de Informática - IV EBAI. KAPELUSZ S.A., Buenos Aires, Janeiro/1989.

[22] - Mendez, C.R.A.
Tese de mestrado em fase de conclusão - FEE - UNICAMP

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

[24] - Paglioli, A.J.; Jacinto, D.C.A; Madeira, E.R.M.; Fernandes, I.A.; Mendes, M.J.; Coreaa, J.N.; Lima, J.M.S; Zabeu, M.C.; Souza, V.L.P.
"SISDI-MAP : Sistema didático do protocolo e da interface de aplicação do MMS do MAP", Seminário Franco-Brasileiro em Sistemas Informáticos Distribuidos. Florianópolis-SC, Setembro/1989.

[25] - Restovic, M.I.V.
Tese de mestrado em andamento - FEE - UNICAMP.

[26] - Rose, Marshall T.
"The Open Book - A Practical Perspective on OSI"
Prentice Hall, New Jersey, 1990.

[27] - Svobodova, Liba; Janson, P.A.; Mumprecht, E.
"Heterogeneity and OSI", IEEE Journal on Selected Areas in Communications. Vol. 8, No 1, January/1990.

[28] - Svobodova, Liba
"Implementing OSI Systems", IEEE Journal on Selected Areas in Communications. Vol. 7, No 7, September/1989.

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

[30] - Zabeu, M.C.A.
"Um modelo para configuração de núcleos para tempo real"
Tese de mestrado - FEE - UNICAMP. Novembro/1989.

ANEXO A : A LINGUAGEM DE ESPECIFICAÇÃO EPOS-R

A linguagem de especificação EPOS-R é utilizada durante as fases de definição das necessidades do usuário (requisitos do sistema) e do modelamento da solução conceitual do problema. Trata-se de uma linguagem semi-formal, onde as regras de sintaxe correspondem a uma combinação de semântica formal e texto em linguagem natural. Em geral, a especificação em EPOS-R permite um extensivo uso da descrição verbal informal, ficando a cargo do usuário da linguagem determinar o grau de formalidade.

Uma especificação em EPOS-S é composta por :

- Descrição do problema,
- Projeto conceitual, e
- Dicionário de termos.

A descrição do problema é realizada através de uma descrição verbal informal, estruturada em capítulos e seções. O projeto conceitual, que representa uma solução conceitual para o problema, pode ser especificado de modo mais formal, através de tabelas de decisão.

Em paralelo a especificação EPOS-R pode-se criar um dicionário para a definição de termos e designadores. Este dicionário, denominado LEXICON, possui função semelhante à de um glossário, e permite definir, referenciar e completar termos em qualquer ponto durante à especificação do projeto. Pode englobar termos pertinentes a especificação do problema e ao projeto conceitual, bem como termos mencionados nas especificações realizadas nas linguagens EPOS-S e EPOS-P. Este dicionário auxilia a comunicação entre os membros do projeto e aumenta a compreensão da documentação posteriormente gerada.

A estruturação da descrição do problema e do projeto conceitual é formalmente organizada em capítulos, e que por sua vez são divididos em seções <SECTION>. As seções possuem uma classificação decimal que consiste de uma série de números positivos separados e concluídos por pontos. Capítulos e seções podem ser compostos dos seguintes elementos :

- Textos;
- Referências a termos do dicionário;
- Descrição de componentes de requisitos identificáveis;
- Reconhecimento de componentes de requisitos identificáveis;
- Troca de componentes de requisitos identificáveis.

Os textos descrevem o atual conteúdo da especificação. Podem ser definidos, ou como textos em linguagem natural delimitados por aspas (") e com a opção de comandos de formatação de textos específicos da linguagem EPOS-R, ou por arquivos ASCII. Esta descrição não é verificada formalmente e não necessita satisfazer qualquer requisito sintático.

O projetista pode fazer referências a termos que estão definidos no dicionário, por exemplo, termos que são utilizados na seção. Esta referência permite ao usuário da documentação da especificação saber que o termo está definido no dicionário de termos. Note-se que o termo é apenas referenciado e a definição do mesmo somente é possível no dicionário de termos (LEXICON). Esta referência é indicada pela palavra chave <TERM>.

Componentes de requisitos identificáveis são nomes atribuidos às asserções sobre as características requisitadas pelo problema e à capacidade técnica do sistema a ser desenvolvido. Estes componentes podem estar presentes tanto na especificação do problema como no projeto conceitual. No caso do EPOS-R são definidos dois componentes de requisitos identificáveis :

- **Requisitos**, indicados pela palavra chave <REQUIREMENT>, são condições externas do sistema e são estabelecidos pelo cliente durante a descrição do problema;

- **Restrições**, indicados pela palavra chave <CONSTRAINT>, são fatores limitantes baseados na análise da situação do projeto e que podem surgir, por exemplo, de demandas técnicas e organizacionais.

Alguns destes componentes de requisitos identificáveis podem ser satisfeitos na própria especificação EPOS-R. Neste caso o reconhecimento dos requisitos e/ou restrições deve ser indicado na seção apropriada da especificação através da palavra chave <FULFILS>. Um componente de requisito identificável pode ser satisfeito completamente em uma única seção, ou parcialmente em várias seções, e neste último caso deve-se agregar o atributo <PARTILY> em cada seção.

Assim, se uma seção contém uma notação <FULFILS> deve também conter a solução para o respectivo requisito e / ou restrição. Note-se que a verificação se o componente de requisito identificável é realmente satisfeito não pode ser realizada de forma automática, pois o mesmo é descrito em linguagem natural ou através de tabelas de decisão, ficando a cargo do projetista tal responsabilidade.

Durante a confecção do projeto conceitual o EPOS-R permite uma substituição dos componentes de requisitos identificáveis definidos na fase de descrição do problema. Esta substituição proporciona um refinamento e uma descrição mais precisa dos requisitos e/ou restrições do projeto. Esta substituição é indicada pela palavra chave <SUBSTITUTE> e pode não somente substituir componentes de requisitos identificáveis definidos na descrição do problema como também no próprio projeto conceitual.

A descrição do conteúdo de um componente de requisito identificável pode ser feita de maneira informal, utilizando-se de textos em linguagem natural ou formalmente, com o auxilio de tabelas de decisão.

Uma tabela de decisão é um método de descrição tabular para processos de decisão que podem ser formalizados. Ela inclui relacionamentos que são indicados por regras. Uma regra é a determinação precisa do relacionamento entre condições e a correspondente operação (ação) a ser tomada. Uma tabela de decisão é descrita no EPOS-R como ilustra a Figura A.0.

DECISION-PROCESS:		Nome da tabela de decisão.
CONDITIONLIST	:	Lista de condições e seus possíveis valores.
OPERATIONLIST	:	Lista de operações e referências a outras tabelas.
RULES	:	Lista dos valores das condições e valores das operações para cada regra.
COMMENT	:	Seção opcional para comentários.

Figura A.0. - Tabela de Decisão do EPOS-R

● <CONDITIONLIST> - Define variáveis, e os possíveis valores que estas variáveis podem assumir. Estes possíveis valores são em quantidade finita. Assim cada condição é composta por um identificador e uma lista com todos os valores aceitáveis.

● <OPERATIONLIST> - Define as operações que podem ocorrer sob determinadas condições. Estas operações são utilizadas para descrever as regras do processo de decisão. Para as operações que não terão uma definição formal mais adiante basta incluir o identificador da mesma; caso contrário, se a operação será formalizada em um outro processo de decisão o identificador deve ser precedido da palavra chave <LOOKAT>.

● <RULES> - Corresponde a especificação real da tabela de decisão, e contém um grupo de regras. Uma regra especifica quais operações devem ser executadas sob determinadas condições. As condições das regras podem assumir, ou um dos possíveis valores especificados no ítem <conditionlist>, ou um hífen (-) indicando que a regra é independente desta condição. As operações podem assumir :

- O valor "X", indicando que a operação especificada ocorrerá sob as condições especificadas, e neste caso a sequência de execução das operações não é definida, sendo executadas em ordem aleatória;
- Um valor inteiro, indicando que a operação apropriada também deverá ser executada sob as condições especificadas, porém a sua sequência é especificada de acordo com a ordem crescente dos números inteiros;
- Um hífen (-), indicando que a operação não será executada se as condições relativas à regra são satisfeitas.

Existe uma regra opcional especial, denominada <ELSE-RULE>, que especifica as operações que devem ocorrer se nenhuma regra do processo de decisão se aplica. Esta regra não possui nenhum valor para as condições, apenas para as operações a serem executadas. Pode estar presente uma única vez em cada processo de decisão, e se presente implica que o processo de decisão é completo.

● <COMMENT> - É útil para o projetista inserir informações sobre o processo de decisão, como o propósito do processo de decisão, o relacionamento das condições e operações, etc.

Tabelas de decisão são descritas formalmente e permitem testes de consistencia e completeza, além disso são de fácil entendimento tornando-se um bom meio de comunicação entre as pessoas que estão envolvidas no projeto.

ANEXO B : A LINGUAGEM DE ESPECIFICAÇÃO EPOS-S

O EPOS-S é uma linguagem de especificação formal utilizada para descrever a implementação do sistema de software e hardware, a partir do projeto conceitual descrito na linguagem EPOS-R. A palavra formal implica que determinadas regras de sintaxe devem ser seguidas durante a descrição das relações funcionais e dos componentes específicos do processo.

Para descrever os diferentes níveis de projeto do sistema são definidos o que se designou de **"Objetos de Projeto"**. Todos os objetos de projeto definidos pelo EPOS-S iniciam com uma palavra chave para indicar o tipo de descrição (ACTION, DATA, etc.), seguido de um nome arbitrário para identificar o objeto de projeto. O término de uma descrição é indicada pela palavra chave acrescida de um sufixo 'END' (ACTIONEND, DATAEND, etc.), como ilustra a Figura B.0.

Palavra chave	Nome do objeto de projeto
Descrição	[Mandatório]
Decomposição	[Opcional]
Relação	[Opcional]
Palavra chave + 'END'	

Figura B.0. - Estrutura Geral de um objeto de projeto do EPOS-S

São definidos sete objetos de projeto : ACTION, MODULE, DATA, CONDITION, DEVICE, INTERFACE e EVENT. Todos estes objetos de projeto possuem uma parte obrigatória comum, denominada "DESCRIPTION-Part". Esta parte é delimitada pelas palavras chaves <DESCRIPTION> e <DESCRIPTIONEND>, e é propícia para a inclusão de características relacionadas a cada objeto de projeto, como por exemplo, uma descrição verbal da função do objeto de projeto, informações adicionais, data de testes, requisitos de desempenho,

requisitos da especificação EPOS-R que são cumpridos pelo objeto de projeto, etc. A seguir apresenta-se uma breve descrição de cada um destes objetos de projeto definidos pela linguagem EPOS-S.

B.1. - OBJETO DE PROJETO DO TIPO AÇÃO

São indicados pela palavra chave <ACTION> e descrevem atividades na qual dados são operados, atualizados, trocados ou processados, em um certo período de tempo. Pode ou representar um processamento puramente funcional não relacionado a uma realização específica, ou o processamento de dados por um programa, dispositivo ou pessoa. Se necessário, pode ser decomposta em várias outras subações. Uma ação pode possuir os seguintes atributos :

- <TASK> : uma "*task*" é uma atividade independente executada assincronamente.
- <**MACRO**> : corresponde a uma sequência de instruções que se repetem com frequência durante a especificação do sistema, mas não correspondem a chamadas de procedimento. Macros não possuem parâmetros.
- <**PROCEDURE**> : corresponde a uma sequência de instruções que são chamadas de vários pontos da especificação com a opção de passagem de parâmetros.

Uma ação que não possui atributos é denominada "ação simples". É funcionalmente semelhante a uma <ACTION MACRO>, exceto que não se repete com frequência na especificação. Após determinar o nome de uma ação, uma série de informações opcionais podem ser acrescentadas. A seguir apresenta-se as mais usuais.

- <**DECOMPOSITION**> : As ações podem ser decompostas passo-a-passo criando uma especificação "*top-down*" do sistema. Isto significa que uma ação é dividida em várias subações, que em suas combinações e inter-relações, reproduzem exatamente a ação em questão. As subações podem ser decompostas em outras subações ou em fluxos de controle. Para representar o fluxo de controle

diferentes construções e operações elementares estão disponíveis na linguagem EPOS-S. A seguir apresenta-se tais construções de fluxo de controle, onde as palavras em maiúsculas correspondem a palavras reservadas da linguagem EPOS-S e as palavras em minúsculas a objetos de projeto definidos na especificação.

- Sequência de Ações

```
ação-1; ação-2; ação-3; ...;
```

- Ações Paralelas

```
PARALEL(ação-1, ação-2, ...);
```

- Ramificação Condicional

```
IF condição
     THEN ações-1
     ELSE ações-2
FI
```

- Múltiplas Ramificações

```
SWITCH dado IN
     CASE valor-1 : ações-1
     ... ... ... ... ... ...
     CASE valor-n : ações-n
     OUT          : ações-erro
NI
```

- "Loop" Contador

```
REPEAT nro-vezes DO
     ações
OD
```

- "Loop" Condicional

```
WHILE condição DO
     ações
OD
```

- Dado dependente de "delay"

```
WAIT UNTIL condição WITHIN delay
     THEN ações-1
     ELSE ações-2
WAITEND
```

- Evento dependente de "delay"

```
WAIT UNTIL evento WITHIN delay
     THEN ações-1
     ELSE ações-2
WAITEND
```

- Troca de Informações ("*Rendezvous*")

```
WAIT FOR RECEIVE evento WITHIN delay
     THEN ações-1
WAITEND
```

- "Delay" Incondicional

```
DELAY(tempo);
```

- Término de Ações

```
STOP(ação-1, ação-2, ...);
```

- Habilitação de Eventos

```
SET(evento-1, evento-2, ...);
```

- Desabilitação de Eventos

```
RESET(evento-1, evento-2, ...);
```

Se uma ação é decomposta em subações disparadas exclusivamente por processos e eventos no tempo, isto é, sem relações de fluxo de controle entre as subações, a estrutura hierarquica é dada por uma decomposição independente. Esta decomposição é delimitada por "(\" e "/)".

- <CODE> : Neste ítem permite-se especificar a codificação de

uma ação em uma linguagem de programação arbitrária, determinada pelo projetista. O código do programa é delimitado por aspas (") e é especificado como texto. Note que esta codificação corresponde a implementação real em software de uma ação e geralmente se aplica as ações terminais, isto é, ações que não possuem decomposição.

● <TRIGGERED> : Ações independentes podem ser disparadas por processos e eventos no tempo. Tais eventos devem ser declarados aqui. O evento tem que ser um objeto de projeto do tipo EVENTO e deve estar habilitado através da operação SET. O evento <SYSTEMSTART>, que ocorre uma única vez no sistema e indica a ação principal, não necessita ser especificado explicitamente e nem ser setado.

● <INPUT/OUTPUT> : Os dados utilizados por uma ação são descritos formalmente através da especificação dos dados de entrada e dados de saída. Também é possível especificar as interfaces utilizadas na recepção e/ou transferência de dados.

● <ALT> : Este ítem é utilizado para descrever o mecanismo de transferência de dados para as ações e/ou procedimentos que são utilizados várias vezes durante a especificação do sistema. Entende-se como mecanismo de transferência de dados a atribuição do valor atual da variável para o parâmetro formal do procedimento chamado.

● <REALIZATION> : Permite especificar informação sobre o possível tipo da realização de uma ação. Existem três alternativas de realização :

- □ SOFTWARE - indicando que a ação é desempenhada por software. Neste caso é possível especificar a linguagem de programação na qual será realizada, e o nome do arquivo fonte a ser produzido "automaticamente" pelo gerador de código apropriado;
- □ HARDWARE - indicando que a ação é realizada por um dispositivo físico;

- **MANUAL** - indicando que a ação é executada pelo usuário da mesma.

● <PROCESSED> : Permite especificar o dispositivo na qual a ação é executada. Este dispositivo pode ser, por exemplo, um sistema homem-máquina desempenhando a ação, unidades de hardware que executam a função da ação, computadores que executam a ação na forma de programas, etc..

B.2. - OBJETO DE PROJETO DO TIPO MÓDULO

São representados pela palavra chave <ACTION MODULE> e diferem dos objetos de projeto do tipo AÇÃO em sua semântica. Este objeto de projeto descreve subsistemas onde não é importante, ou ainda não é conhecido o seu ajuste no fluxo de controle, mas onde uma separação lógica se faz necessária. Sintaticamente corresponde a uma ação com o atributo <MODULE>, mas as informações adicionais possuem um significado diferente.

● <DECOMPOSITION> : O componente de mais alto nível de uma especificação é um módulo representando todo o sistema. Nos primeiros níveis de decomposição o projeto pode ser logicamente decomposto em outros módulos e/ou ações. Em geral, módulos que já possuem um grande nível de refinamento, são decompostos em ações do tipo TASK, PROCEDURE ou MACRO. Ao contrário de uma ação, um módulo é uma unidade não executável, e um módulo pode ser decomposto em módulos e/ou ações, porém uma ação não pode ser decomposta em módulos, só em ações.

● <IMPORT/EXPORT> : Permite descrever as interfaces entre módulos. Pode ser dividida em quatro partes : Exportação de módulos e ações, Exportação de dados, Importação de módulos e ações, e Importação de dados. Os objetos exportados por um módulo tornam-se disponíveis para outros módulos, e os objetos importados por um módulo são aqueles objetos de projetos providos por outros módulos. O módulo também implementa o principio do "ocultamento da

informação", pois para o projeto o ambiente de um módulo é representado por suas interfaces.

- <INPUT/OUTPUT> : Enquanto no ítem <import/export> especificam-se unicamente a disponibilidade de objetos (módulos, ações e dados) entre módulos, neste ítem é possível estabelecer conexões entre módulos e seu ambiente. Neste caso, os dados de entrada e saída podem ser relacionados com as correspondentes interfaces de entrada e saída.

De maneira análoga ao objeto de projeto do tipo ação também é possível incluir informações relativas a : <DESCRIPTION>, <CODE> e <REALIZATION>.

B.3. - Objeto de Projeto do Tipo Dado

São representados pela palavra chave <DATA> e são utilizados para descrever todas as informações processadas durante a fase de projeto. Existem três possibilidades de definição de dados :

- <TYPE> - Permite a definição de um novo tipo abstrato de dado, a partir dos dados já definidos na especificação.
- <FORMAL> - Indica que o dado é um parâmetro formal de um procedimento definido no sistema de software.
- DADOS SIMPLES - Estes dados não possuem atributos especiais e são de algum tipo primitivo definido na linguagem EPOS-S.

A linguagem EPOS-S possui vários tipos primitivos de dados, como mostra a Tabela B.0. Além destes tipos primitivos é possivel definir novos tipos abstratos de dados, permitindo a criação de estruturas de dados que modelam de forma adequada os dados dos sistema. Para a definição de novos tipos uma série de atributos são definidos pelo EPOS-S. São eles :

- ARRAY DE DADOS - É caracterizado pelo atributo <ARRAY> e permite a definição de vários elementos de um mesmo tipo

logicamente conectados em um array n-dimensional.

● **CLASSE** - É caracterizado pelo atributo <CLASS> e permite a especificação de um grupo de dados que possuem o mesmo tipo primitivo ou abstrato de dados.

● **ARQUIVOS** - É caracterizado pelo atributo <FILE>. Um arquivo é uma coleção de dados, denominados registros lógicos, e que podem ser de vários tipos ou tamanhos. A estrutura de dados em um arquivo é dado por uma sequência de dados no arquivo. A menor unidade acessada é um registro lógico e assim o acesso direto a decomposição de um dado de um registro lógico não é permitido.

● **FILAS** - É caracterizado pelo atributo <QUEUE>. Uma fila é um campo de dados que consiste de vários elementos idênticos, referidos como entidades lógicas. Estas entidades lógicas podem ser dados elementares ou podem ser decompostos no próximo nível de refinamento do projeto. O acesso é na ordem FIFO ("*First-In-First-Out*"), isto é, a primeira entidade a entrar é também a primeira a sair.

● **PILHAS** - É caracterizado pelo atributo <STACK>. Pilhas são idênticas as filas, exceto em sua estratégia de acesso, que no caso, é a LIFO ("*Last-In-First-Out*"), onde somente é possível acessar o elemento armazenado por último.

● **SINAIS** - É caracterizado pelo atributo <SIGNAL> e são utilizados para identificar dados que são transmitidos como sinais para conectores de dispositivos, ou para identificar descrições de sequências de estados mecânicos, elétricos, óticos, etc.

Além destes atributos é possivel especificar :

- O escopo ("SCOPE-Part") de um determinado dado dentro do sistema, como por exemplo um dado global ao sistema ou local a um procedimento;
- As dimensões ("BOUNDS-Part") para dados do tipo array

n-dimensional, bem como os limites de cada dimensão;

- Atribuição de valores ("IDENTICAL-Part") para um dado definido na especificação, permitindo a definição de constantes;
- Inicializar ("INITIAL-Part") uma variável do sistema;
- Especificar o range ("RANGE-Part") de valores que as variáveis podem assumir;

Tipo de dado	Explanação (valores permitidos)
FIXED	Número inteiro
FLOAT	Número em ponto flutuante
CHAR	Caracter
BIT	Bit
BOOLEAN	Valores Lógicos TRUE ou FALSE
POINTER	Ponteiro para algum tipo de dado
DUR	Duração
CLOCK	Tempo

Tabela B.0. - Tipos de Dados Primitivos do EPOS-S

B.3. - OBJETO DE PROJETO DO TIPO INTERFACE

É identificado pela palavra chave <INTERFACE> e descrevem a troca de dados entre :

- Os sistemas de processamento e o ambiente, isto é, entre o processo físico e o operador do sistema;
- Componentes individuais do sistema de processamento, isto é, entre os computadores individuais de uma rede; e
- Ações, representando interfaces lógicas.

Estas interfaces não necessariamente necessitam ser interfaces de hardware, mas também interfaces lógicas entre programas. Existem basicamente dois tipos ("TYPE-Part") de interface definidos :

Interfaces externas, identificadas pela palavra chave <EXTERNAL>, e interfaces internas, identificadas pela palavra chave <INTERNAL>. Devido à estreita relação entre dados e interfaces, pois dados também são trocados via interfaces, a descrição do objeto de projeto do tipo INTERFACE é similar à descrição do objeto de projeto do tipo DADO, e assim dispensam-se maiores detalhes.

B.4. - Objeto de Projeto do Tipo Evento

É identificado pela palavra chave <EVENT> e descreve eventos que ocorrem em processos físicos ou em sistemas em processamento na forma de sinais binários, e que afetam a operação das ações. Um evento pode ser introduzido como um termo genérico para vários eventos, e neste caso pode ser decomposto ("DECOMPOSITION-Part") em um arbitrário número de outros eventos, no próximo nível de refinamento do sistema. Em termos de ocorrência no tempo, existem cinco categorias de eventos que podem ser especificados na "TYPE-Part" :

- Eventos estocásticos, identificados pela palavra chave <INTERRUPT>, que ocorrem de maneira irregular no tempo.
- Eventos que ocorrem ciclicamente em intervalos de tempo regular, e são identificados pela palavra chave <CYCLIC>.
- Eventos que ocorrem somente em um único e bem definido intervalo de tempo. Eventos deste tipo são identificados pela palavra chave <CLOCK>.
- Eventos que são utilizados na sincronização de tarefas, e são identificados pela palavra chave <RENDEZVOUS>.
- Eventos que servem para disparar uma exceção, e são identificados pela palavra chave <EXCEPTION>.

Em projetos orientados a eventos, os eventos formam o ponto inicial do desenvolvimento. Neste caso as ações que devem ser disparadas por eventos são determinadas na "TRIGGERS-Part". Deve-se observar que o evento <SYSTEMSTART> já é prédefinido na

linguagem EPOS-S e deve ser utilizado para indicar a inicialização do sistema.

B.5. - Objeto de Projeto do Tipo Condição

São identificados pela palavra chave <CONDITION> e descrevem condições dependentes de dados que influenciam o fluxo de controle e as ações. Condições são entidades lógicas e podem assumir os valores lógicos "TRUE" (condição satisfeita) ou "FALSE" (condição não satisfeita).

De maneira semelhante aos objetos de projeto do tipo AÇÃO pode ser decomposta em outras subcondições em sua "DECOMPOSITION-Part" ou especificado diretamente em uma determinada linguagem de programação através de sua "CODE-Part". A decomposição de uma condição pode seguir um processo "*top-down*", onde é decomposta em um arbitrário número de subcondições que são logicamente combinadas, indicando como o valor booleano da condição é calculado. A decomposição pode incluir operadores Relacionais (Tabela B.1) e/ou operadores Lógicos (Tabela B.2).

Operadores Relacionais	
>	(maior que);
>=	(maior ou igual);
<	(menor que);
<=	(menor ou igual);
=	(igual) e
/=	(diferente).

Tabela B.1. - Operadores Relacionais do EPOS-S

Operadores Lógicos	
NOT	(negação lógica);
AND	(E lógico);
OR	(OU lógico) e
EXOR	(OU excluivo).

Tabela B.2. - Operadores Lógicos do EPOS-S

B.6. - Objeto de Projeto do Tipo Dispositivo

Uma característica importante do EPOS é que além do suporte ao sistema de software, também oferece suporte para o projeto e a descrição dos componentes de hardware. Este suporte é proporcionado pelo objeto de projeto DISPOSITIVO, identificado pela palavra chave <DEVICE>, e são úteis para a descrição de unidades que desempenham ações ou funções de interface. Tais unidades podem ser :

- Sistemas homem-máquina ou unidades organizacionais, que desempenham funções descritas por objetos de projeto do tipo AÇÃO e INTERFACE;
- Componentes físicos através dos quais as ações ou interfaces são realizadas;
- Hardware de computadores tais como microcomputadores e seus componentes, nos quais ações são executadas na forma de programas.

Os elementos formais do objeto de projeto dispositivo são apropriados para a descrição de hardware que possuam uma estrutura hierárquica. Por exemplo, iniciando com o sistema completo (máquinas ou dispositivos), microcomputadores podem ser descritos por objetos de projeto do tipo dispositivo no nível de seus principais componentes ("COMPONENTS-Part"), tais como processadores, memórias, periféricos, etc. Cada componente do dispositivo pode então ser descrito incrementando-se o nível de detalhes das partes individuais. Além disso, permite especificar as conexões ("INOUT-Part") do dispositivo para o hardware como plugs, conectores, soldas, ou pinos de um componente, e também todas as conexões para o ambiente externo.

ANEXO C : O GERADOR DE CÓDIGO C-COMPOSER

O C-Composer é uma ferramenta de software do sistema EPOS que gera automaticamente um programa fonte em linguagem C a partir de uma especificação EPOS-S. As informações especificadas através dos objetos de projeto da base de dados EPOS são traduzidas para uma estrutura de programa C. São criados um programa principal ("*main*") e subprogramas ("*procedures*") com os respectivos fluxos de controle e dados associados. A seguir apresentam-se algumas características e regras de conversão deste gerador de código.

- **Seleção de um objeto inicial**

O C-Composer permite uma tradução de porções da base de dados através de seleção de um objeto de projeto inicial. A partir deste objeto inicial todas as ações pertencentes ao seu escopo serão traduzidos para um programa fonte em linguagem C. Qualquer ação contida na base de dados pode ser selecionada como objeto inicial, exceto aquelas que possuem o atributo <TASK>.

- **Criando a estrutura de um programa**

O C-Composer cria todos os subprogramas, tarefas e dados pertencentes ao escopo do objeto inicial selecionado. Para obter um programa C executável, o objeto inicial deve ser do tipo <ACTION MODULE> e conter em sua decomposição o programa principal. Por sua vez, o programa principal deve ser uma <ACTION TASK> e com o evento SYSTEMSTART declarado no ítem TRIGGERED. Se a declaração do evento SYSTEMSTART é omitida ou o programa principal é uma ação de tipo <ACTION PROCEDURE>, um procedimento em C é gerado. Para ações que não são do tipo <ACTION MODULE> é gerado, ou um procedimento em C se a ação possui o atributo <PROCEDURE>, ou um fluxo de controle se a ação possui o atributo <MACRO> ou é uma simples ação sem atributo.

- **Análise de Transformabilidade**

Antes de realizar a transformação de uma especificação em EPOS-S para C, uma análise de transformabilidade é executada. Esta análise verifica os seguintes ítens da especificação, a partir do objeto inicial selecionado :

- **Completeza** ("*Completion*") : Verificação se todas as informações requeridas para a transformação estão especificadas, isto é, se todos os objetos de projeto referenciados na especificação estão definidos.
- **Consistência** ("*Consistency*") : Se a especificação EPOS-S não contém construções de fluxos de controle que não são transformáveis para C. Tais construções são : PARALEL, SET, RESET, DELAY e WAIT.
- **Conformidade** ("*Compliance*") : Conformidade com as restrições e requisitos de cada objeto de projeto especificado. Por exemplo, uma ação com o atributo <MACRO> não pode possuir parâmetros; dados indexados ("*arrays*") sem a declaração de suas dimensões; etc.

A geração de código somente é executada se a análise referente a estes três ítens é correta.

C.1. - TRANSFORMAÇÃO DE AÇÕES

A transformação de uma ação depende do atributo associado à mesma. Os atributos <MODULE> e <TASK> são permitidos somente para a indicação de um programa principal ("*main*") em C. Ações que possuem o atributo <PROCEDURE> são transformadas em procedimentos da linguagem C, sendo gerados o identificador, a lista de parâmetros, se existirem, e o respectivo corpo do procedimento com suas variáveis locais. Ações com o atributo <MACRO> e aquelas sem atributo (ações simples) geram apenas partes de programas que compõem o programa principal e/ou procedimentos.

- **Transformação de uma "DECOMPOSITION-Part"**

Ações que possuem uma decomposição (DECOMPOSITION-Part) em sua especificação são compostas por subações e/ou construções de fluxo de controle e operações elementares que expressam o comportamento da ação em questão. A seguir, apresentamos a transformação das estruturas de controle pertencentes a linguagem EPOS-S para C :

- **Ramificação Condicional**

```
IF (condição)
     THEN { ações-1 ; }
     ELSE { ações-2 ; }
```

- **Múltiplas Ramificações**

```
SWITCH (dado)
{    CASE valor-1 : ações-1 ; break ;
     ... ... ... ... ... ... ... ...
     CASE valor-n : ações-n ; break ;
     DEFAULT      : ações-default ; break;
}
```

- **"Loop" Contador**

```
FOR (counter = 1; counter <= nro-vezes; counter++)
{ ações ; }
```

- **"Loop" Condicional**

```
WHILE (condição)
{ ações ; }
```

- **Término de Ações**

```
EXIT (1);
```

As subações são transformadas de forma recursiva até chegarem a ações simples, que não possuem decomposição, ou ações terminais onde o próprio código está especificado. Os fluxos de controle PARALEL, SET, RESET, DELAY e WAIT não são transformáveis para

código C.

- Transformação de uma "CODE-Part"

Ações que possuem uma CODE-Part em sua especificação já estão expressas em uma apropriada codificação em linguagem C, e neste caso, o código é copiado diretamente para o corpo do programa fonte gerado.

C.2. - TRANSFORMAÇÃO DE DADOS

Um dado pode ser decomposto em vários subdados permitindo a especificação de diversos tipos de dados abstratos. A definição de dados com o atributo TYPE e compostos de vários registros geram definições de estruturas de dados do tipo "struct" da linguagem C. A Tabela C.0 ilustra a transformação de alguns tipos de dados pré-definidos da linguagem EPOS-S para C.

EPOS	C	Explanação
FIXED	int	Número inteiro
FLOAT	float	Número real
CHAR	char	Caracter
CHAR n	char[n]	String de Caracteres
POINTER	*	Ponteiro
BOOLEAN	BOOLEAN	Lógica
<DATA-ID>	<data-id>	Tipo abstrato

Tabela C.0. - Transformação de Dados para C

Um dado é caracterizado como uma constante se possui em sua especificação uma "IDENTICAL-Part". Estas constantes são geradas no inicio do programa através de um comando "#define" do pré-processador C e são válidas para todo o programa principal. Um range de valores pode ser especificado para qualquer tipo de dado

que não seja especificado como constante. Porém, em C, dados não podem ser definidos com um range de valores, a única exceção são os tipos de dados enumerados ("enum" da linguagem C), onde um grupo de valores ordenados é definido. Assim, se um dado possui o atributo TYPE e um grupo de valores especificados em sua "RANGE-Part" uma definição do tipo "typedef enum" é gerada.

Se o dado possui o atributo TYPE ARRAY e uma "BOUNDS-Part", um tipo de dado indexado com as dimensões espcificadas na BOUNDS-Part é gerado. Devido ao fato de na linguagem C arrays serem endereçados a partir de 0, o tamanho do array é incrementado de 1 visando manter a analogia com as demais linguagens de programação. Além disso, apesar de utilizar um espaço de armazenamento adicional é possível gerar a especificação em outra linguagem de programação sem a necessidade de trocar o endereçamento dos elementos do array.

Se a especificação de um dado possui uma "CODE-Part" este é transferido para o programa fonte C inalterado. Os Tipos de dados CLOCK, BIT n e DUR e dados com os atributos QUEUE, STACK, FILE e SIGNAL não são transformáveis para C.

C.3. - Transformação de Condições

O objeto de projeto do tipo CONDITION é utilizado para a descrição de dados dependentes de condição e que guiam o fluxo de controle de uma ação. Uma condição pode ter uma "DECOMPOSITION-Part" ou uma "CODE-Part". No primeiro caso uma condição pode ser decomposta em várias sub-condições envolvendo operadores relacionais e/ou lógicos. As Tabelas c.1 e c.2 mostram respectivamente as transformações dos operadores relacionais e lógicos para código C.

No segundo caso, referente a CODE-Part, a condição é especificada diretamente em código C. Nesta opção a estrutura da condição já é expressa com os operadores relacionais e/ou lógicos próprios da linguagem C.

EPOS-S	C
= EQ	==
/= NE	!=
< LT	<
<= LE	<=
> GT	>
>= GE	>=

Tabela C.1. - Transformação de Operadores Relacionais

EPOS-S	C
NOT	!
AND	&&
OR	||
EXOR	~

Tabela C.2. - Transformação de Operadores Lógicos

C.4. - LIMITAÇÕES

Algumas construções da linguagem de especificação EPOS-S não são transformáveis para C. São elas :

- Objetos de projeto do tipo DEVICE
- Objetos de projeto do tipo INTERFACE
- Objetos de projeto do tipo EVENT
- Objetos de projeto do tipo DATA com os atributos QUEUE, STACK, FILE e SIGNAL

Se a especificação contém algumas destas construções, no momento da geração de código tais objetos são ignorados, não produzindo nenhuma estrutura ou fluxo de dados e de controle. Porém, se tais construções são necessárias, pode-se gerar código via a opção "CODE-Part" de cada objeto de projeto.