UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA

DEPARTAMENTO DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

# "ASPECTOS DE IMPLEMENTAÇÃO DO PROTOCOLO DE SESSÃO UTILIZANDO FERRAMENTAS CASE"

AUTOR: CARLOS RAUL ARIAS MENDEZ
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.

Dezembro de 1990

Este exemplar corresponde à redação final da defendida por Carlos Raul Arias Mendez e aprovada pela Comissão Julgadora em 17/12/1990

Orientador

UNICAMP
BIBLIOTECA CENTRAL

Dedico este trabajo a mi esposa Rosa y a mis hijos Alejandra y Rodrigo.

## A G R A D E C I M E N T O S

Ao Prof.Dr. Manuel de Jesus Mendes pela orientação e apoio, a meus amigos de estudo e a todos aqueles que, de uma ou de outra contribuiram na realização deste trabalho, desejo expressar meus agradecimentos.

# S U M Á R I O

O presente trabalho mostra como utilizar a ferramenta CASE, nomeada EPOS, para especificar a implementação do protocolo da camada de Sessão do modelo OSI/ISO (unidades funcionais Kernel e Full-duplex), considerando-se parte do projeto SISDI_MAP atualmente em andamento no Departamento de Computação e Automação Industrial-FEE-UNICAMP.
Como produto da especificação obteve-se o código final, na linguagem "C", que em parte foi gerada automaticamente pelo EPOS.

# A B S T R A C T

The present work shows how to use a CASE tool, called EPOS, to specify an implementation of OSI/ISO Session layer protocol (Kernel and Full-duplex functional units), which is part of the SISDI_MAP project, presently under development in the Departamento de Computação e Automação Industrial-FEE-UNICAMP.

A final code was obtained as a product of the specification, in "C" language, generated in part automatically by EPOS tool.

# I N D I C E


CAPÍTULO 1 : INTRODUÇÃO.....1-1

1.1. Objetivos.....1-3

1.2. O projeto SISDI_MAP.....1-4

1.2.1. Modo de operação do SISDI_MAP.....1-4

1.3. EPOS - Uma breve descrição.....1-6

1.4. A Camada de Sessão.....1-8

CAPÍTULO 2 : A CAMADA DE SESSÃO.....2-1

2.1. Funções da camada de sessão.....2-1

2.2. Definição de conceitos utilizados.....2-1

2.2.1. Token.....2-1

2.2.2. Sincronização e unidades de diálogo.....2-2

2.2.3. Atividade.....2-3

2.2.4. Resincronização.....2-3

2.2.5. Qualidade de Serviço (QOS).....2-4

2.3. Negociação.....2-5

2.4. Fases e serviços da camada de sessão.....2-5

2.4.1. Fase de estabelecimento da conexão.....2-5

2.4.2. Fase de transferência de dados.....2-5

2.4.3. Fase de liberação da conexão de sessão.....2-8

2.5. Unidades Funcionais e Subconjuntos.....2-14

2.5.1. Unidades Funcionais.....2-14

2.5.2. Subconjuntos.....2-17

2.6. Protocolo da camada de sessão.....2-20

2.6.1. Serviços utilizados pela SPM.....2-20

2.6.2. Exemplos explicativos do protocolo de sessão.....2-23

2.7. Escopo da implementação.....2-28

CAPÍTULO 3 : DESCRIÇÃO DO PROJETO DESENVOLVIDO.....3-1
3.1. Especificação dos requisitos.....3-1
3.1.1. Roteiro.....3-1
3.1.2. Descrição da especificação dos requisitos feita com EPOS-R.....3-5
3.2. Especificação da implementação.....3-8
3.2.1. Roteiro.....3-8
3.2.2 Descrição da especificação da implementação.....3-13
3.2.2.1. Descrição.....3-14
3.2.3. Geração do código a partir da especificação.....3-28

CAPÍTULO 4 : EXEMPLO DE EXECUÇÃO.....4-1
4.1. Seqüência A1.....4-3
4.2. Seqüência A2.....4-5
4.3. Seqüência A3.....4-8

CAPÍTULO 5 : CONCLUSÕES.....5-1

REFERÊNCIAS BIBLIOGRÁFICAS

ANEXOS
A: Tabelas de Estado da SPM que foram implementadas.
B: Estrutura de Dados dos parâmetros das primitivas e SPDUs na camada de sessão
C: Estrutura de Dados dos parâmetros das primitivas na camada de transporte
D: Requisitos da implementação. (Ver anexo em texto separado).
E: Especificação da implementação. (Ver anexo em texto separado).
F: Listagem gerado automaticamente pelo Composer-C. (Ver anexo em texto separado).

# CAPÍTULO 1: INTRODUÇÃO.

O rápido desenvolvimento de tecnologias utilizadas na fabricação de computadores, o qual fez reduzir tamanho e custo com aumento do desempenho, está permitindo, entre outras coisas, distribuir geograficamente sistemas computacionais, provocando uma maior necessidade de comunicação (dados) entre estes sistemas sem a intervenção direta do homem, para reduzir as possibilidades de erro.

Ante tal situação emergiram as soluções particulares, especialmente dos fabricantes de computadores, mas que do ponto de vista econômico do usuário, a melhor solução seria contar com os equipamentos heterogêneos que possam comunicar-se entre si para poder preservar seus investimentos em software, inclusive quando mudem as tecnologias do hardware, conseguindo-se uma maior eficiência que no caso da dependencia de um único fornecedor.

Tal solução implicava projetar uma arquitetura de comunicação aberta, no sentido de considerar um vasto campo de aplicações (manufatura, processos, aeronáutica, comunicações, etc.) que possam utilizá-la como comunicação padronizada a nível de sistemas computacionais. A resposta chegou nos fins dos anos 70, nascendo o padrão internacional de comunicação liderada pela ISO (Internatinal Standard Organization) que promove o modelo de referência OSI (Open System Interconnection) das sete camadas, baseado na experiência recolhida das soluções particulares já existentes nas redes proprietárias (SNA da IBM, DECnet da Digital Equipment e outras).

Atualmente, a padronização das quatro camadas inferiores (física, enlace, rede e transporte) está definida. As camadas de sessão (5) e apresentação (6) também estão definidas mas podem ainda sofrer modificações (por exemplo, para a camada de sessão está-se estudando a possibilidade de oferecer serviços de difusão -"broadcasting"). A camada de aplicação (7) ainda não está fechada. Por estes motivos o modelo padronizado da OSI/ISO ainda não pode ser aplicado na sua totalidade: isto levou à General Motors em 1979 a buscar uma solução mais rápida a seu problema (interligar uma diversidade de equipamentos alocados na sua indústria automotriz até chegar ao nível do chão de fábrica), criando-se em 1980 a Força-Tarefa MAP (Manufacturing Automation Protocol)[1] para

preparar a especificação de um sistema aberto de comunicação de dispositivos inteligentes fornecidos por diversos fabricantes; para isso, considerou-se um subconjunto de protocolos do padrão OSI/ISO, juntamente com o protocolo MMS (Manufacturing Message Service) alocado na camada de aplicação e a especificação da interface API (Application Programming Interface) entre o modelo aberto de comunicação e o processo de aplicação. A primeira versão MAP foi publicada em Outubro de 1982. Atualmente a última versão (3.0) está congelada (a partir de 1988 e por 6 anos) para permitir sua implementação no mercado industrial.

No respeitante à camada de sessão, ela forma parte do MAP versão 3.0 através das unidades funcionais Kernel e Full-duplex.

Este trabalho, que visa implementar as unidades funcionais Kernel e Full-duplex do protocolo de sessão, formará parte do projeto de implementação de protocolos MAP na UNICAMP, conhecido como SISDI_MAP, e do qual se dão maiores detalhes no item 1.2. deste capítulo introdutório.

A seguir, apresenta-se uma breve descrição sobre o conteúdo dos capítulos desta tese:

Capítulo1. -Introdução: Além do parágrafo introdutório, apresentam-se os objetivos do presente trabalho (item 1.1.); uma descrição do projeto SISDI_MAP (item 1.2.); uma breve descrição da ferramenta de engenharia de software EPOS (item 1.3.) utilizada para desenvolver o trabalho; e uma descrição qualitativa, sem entrar em detalhes técnicos, sobre a camada de sessão (item 1.4.).

Capítulo2. -A camada de Sessão: Neste capítulo se apresenta um resumo sobre as funções e serviços que a camada de sessão fornece dentro do modelo de referência do Sistema Aberto de Interconexão da ISO.

Capítulo 3. -Descrição do projeto desenvolvido: Este capítulo forma o corpo principal da tese. Nele se descreve como foi desenvolvido o trabalho de especificação dos requisitos e a implementação das unidades funcionais Kernel e Full-duplex da Camada de Sessão, o qual deu origem aos relatórios de: Requisitos da Implementação (Anexo D) e Especificação da Implementação (Anexo E) desenvolvidos utilizando a ferramenta EPOS. Finalmente explica-se como foi gerado o código na Linguagem "C", cuja listagem completa está no Anexo F.

Capítulo 4. -Exemplo de Execução: Para dar uma maior claridade sobre a especificação da implementação apresenta-se, neste capítulo, um exemplo que descreve os passos que o protocolo de sessão percorreria ante a ocorrência de certos eventos, até chegar às ações finais que o protocolo estipula. A descrição é realizada fazendo referência aos objetos de projeto especificados no Anexo E.
Capítulo 5. -Conclusões: Neste último capítulo apresentam-se as conclusões que foram tiradas da experiência do trabalho, durante a implementação do protocolo de sessão, fazendo-se uso da ferramenta de engenharia de software nomeada EPOS.
Anexos: O Anexo A apresenta as tabelas de estado do protocolo de sessão que foram implementadas, o Anexo B a listagem do Arquivo de Mensagens do Protocolo de Sessão e o Anexo C a listagem do Arquivo de Mensagens do Protocolo de Transporte. O Anexo D apresenta o relatório gerado no EPOS-R para a especificação dos requisitos que deve cumprir a implementação, e o Anexo E o relatório gerado no EPOS-S para a especificação da implementação. Finalmente o Anexo F apresenta a listagem do código gerado automaticamente pelo Composer-C a partir da especificação da implementação.

Por serem muito extensos os relatórios dos Anexos D e E, como também a listagem no Anexo F, estes foram anexados à tese num texto separado.

## 1.1 Objetivos.

O presente trabalho visa dois objetivos fundamentais , os quais foram atingidos em etapas sucessivas.

O primeiro objetivo foi elaborar um documento que especifique uma implementação das unidades funcionais Kernel e Full-duplex do protocolo da camada de sessão do modelo OSI/ISO, considerando como ambiente de operação o Sistema Didático MAP (SISDI_MAP), do qual finalmente formará parte. Para alcançar este objetivo fez-se uso da ferramenta de engenharia de software nomeada EPOS.

O segundo objetivo atingido, a partir do projeto especificado na primeira etapa, foi a geração do programa computacional, codificado na linguagem "C", e para instalação num microcomputador IBM-PC ou compatível. Para isso, utilizou-se o gerador de código automático para a linguagem "C" (Composer-C) que o EPOS fornece.

## 1.2. O projeto SISDI MAP.

O SISDI_MAP (SIStema Didático MAP) é um projeto que está em desenvolvimento no Departamento de Computação e Automação Industrial da FEE - UNICAMP, mas que já tem sua estrutura básica implementada [2] utilizando a linguagem C.

Este projeto tem como objetivo principal, simular a comunicação entre dois ou mais usuários de uma rede MAP (Manufacturing Automation Protocol), utilizando um computador IBM-PC ou compatível e um núcleo multi-tarefa e tempo real (desenvolvido na linguagem C., como se detalha na referência [3]).

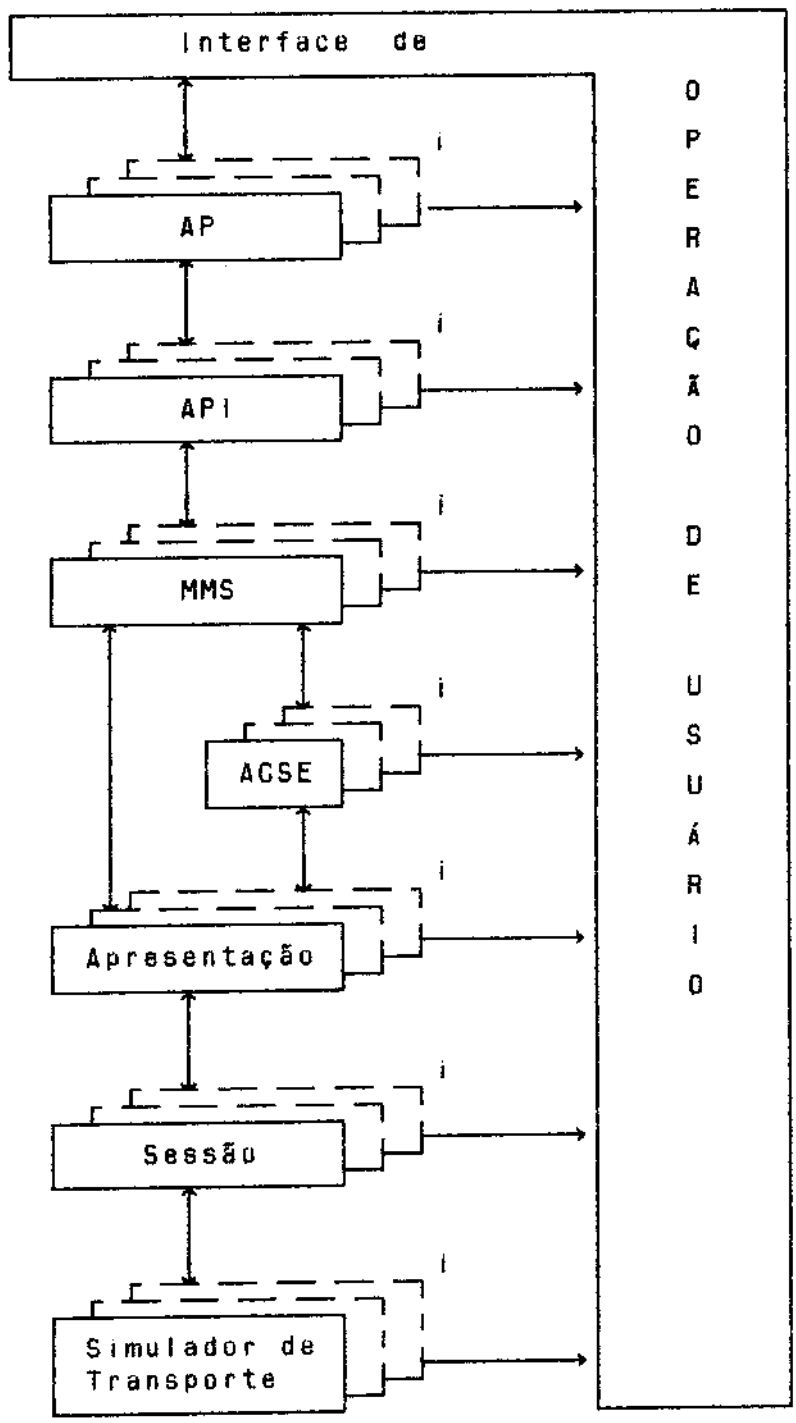
A atual estrutura funcional do SISDI_MAP é mostrada na figura 1.1. Este esquema permitirá incorporar sucessivamente outros protocolos da camada de aplicação.

Além de simular a comunicação, o projeto SISDI_MAP permite monitorar as seqüências de dados geradas e, também, inserir erros nas PDU's que são manipuladas no simulador da camada de transporte, através da Interface de Operação de Usuário.

A implementação da camada de sessão (Kernel e Full-duplex) deverá finalmente inserir-se no ambiente de simulação do SISDI_MAP. Portanto, a implementação final deverá considerar este ambiente através das interfaces entre as camadas de apresentação [17] e transporte (simulada).

### 1.2.1. Modo de operação do SISDI MAP.

De forma breve, pode-se dizer que ele opera em um modo de chaveamento de contexto. Isto quer dizer, que por exemplo a camada de sessão, é representada por uma tarefa para o núcleo de tempo real, sempre escalonada com um índice de contexto que identificará um usuário local ou remoto. Tal chaveamento de contexto é controlado pelo simulador da camada de transporte.




NOTA: As caixas com linhas cheias representam tarefas, e as caixas de linhas segmentadas representam os distintos contextos.

i: índice de contexto.

Figura 1.1: Arquitetura funcional do SISDI_MAP.

Finalmente, a troca de mensagens entre as camadas é feita através de portas unidirecionais, (portas de entrada), nas quais só será armazenado um ponteiro de um bloco de estrutura de dados previamente definido e padronizado para todas as portas.

## 1.3. EPOS - Uma breve descrição

O EPOS (Engineering and Project- management Oriented support System) [4] é uma ferramenta de engenharia de software (CASE) destinada a ajudar no desenvolvimento de projetos de software em aplicações de tempo real com processos concorrentes.

O EPOS foi concebido no Instituto de Engenharia de Controle e Automação de Processos da Universidade de Stuttgart da Rep. Fed. da Alemanha em 1970, e está em constante desenvolvimento. Posteriormente, em 1980, começou a ser comercializado.

Para gerar o projeto, o EPOS fornece várias ferramentas que permitem especificar e criar uma Base de Dados, a qual é gerenciada por este pacote durante todo o ciclo de vida do projeto. Para isso, dispõe-se das seguintes linguagens de especificação:

- EPOS-Requirements (EPOS-R)[5]: Em uma primeira etapa permite estabelecer os requisitos e restrições tanto na proposta do problema como na solução conceitual do projeto de software, incluindo a especificação de tabelas de estado ou de processos de decisão. Além disso, permite a criação de um dicionário de palavras chaves chamado "Lexicon".
- EPOS-Specifications (EPOS-S)[6]: Como segunda etapa, esta ferramenta permite especificar a solução do problema de acordo com os requisitos e restrições já estabelecidas utilizando o que o EPOS define como objetos de projeto (ACTION MODULE, ACTION, DATA, CONDITION, INTERFACE, EVENT e DEVICE).
- EPOS-Project (EPOS-P)[7]: Permite estabelecer responsabilidades dentro do grupo de pessoas que participam no projeto, no sentido de distribuir a execução do projeto entre os membros do grupo, estabelecendo também prazos e datas de término, com o objetivo de permitir o gerenciamento de software.

Durante o desenvolvimento do projeto, o EPOS provê outras ferramentas que atuam sobre a Base de Dados que foi criada para facilitar e orientar o trabalho do projetista. Tais ferramentas são as seguintes:

- EPOS-Communication (EPOS-C)[8]: Permite a interação homem-máquina através do uso de janelas de menu.

. EPOS-Analysis (EPOS-A)[9]: É uma ferramenta de análises para cada etapa da especificação (EPOS-R, EPOS-S e EPOS-P). Em geral, ela faz análises de sintaxe e, em especial, análises específicas para cada etapa: por exemplo para:

- EPOS-R: . Análises de compatibilidade com a informação na Base de Dados.
  . Análise de redundância, consistência e completeza das tabelas de decisão.
- EPOS-S: . Teste de completeza, conflitos de concordância nos tipos dos dados e descrições não referenciadas.
  . Teste de chamadas incorretas de procedimentos, consistências nas sentenças do fluxo de controle de execução das ações (as ações são representadas pelo objeto de projeto ACTION).
  . Análises na consistência da decomposição dos dados (nos objetos de projeto DATA e CONDITION) e nos eventos especificados (no objeto de projeto EVENT).
  . Análise de sincronização.
  . Análise da presença de "deadlock's".
  . Análises para determinar se os requisitos e restrições, definidos em EPOS-R, estão sendo reconhecidos na especificação da solução em EPOS-S.

. EPOS-Management (EPOS-M)[10]: É a ferramenta que, a partir da especificação feita em EPOS-P, ajuda no planejamento, supervisão e controle do projeto, através da geração de gráficos PERT, carta Gantt e distribuição de responsabilidades, incluindo custos.

. EPOS-Documentation (EPOS-D)[11]: Ferramenta utilizada para a geração de documentos (texto e gráfico) de acordo com a informação descrita em EPOS-R, EPOS-S e EPOS-P. Entre eles, pode-se indicar:

- Tabelas de decisão a partir dos processos de decisão estabelecidos em EPOS-R.
- Texto completo ou parcial dos requisitos em EPOS-R e na especificação da implementação em EPOS-S (isso inclui capa,

índice, lista de referências cruzadas, cabeçalhos, pé de linha, etc).

- Extrato do glossário de termos (Lexicon) utilizados em EPOS-R.
- Diagramas hierárquicos, fluxograma, diagramas N-S, redes de Petri, diagramas do fluxo de dados, diagramas sobre hardware (circuitos) de um dispositivo especificado.

Finalmente EPOS fornece ferramentas chamadas "Composer"[12], para a geração automática de código a partir da especificação feita em EPOS-S, em linguagens como ADA, PEARL, PASCAL, FORTRAN e C. Nesta etapa, também é possível fazer certo tipo de modificações sobre o programa fonte gerado automaticamente pelos "Composer" e retroalimentar a Base de Dados para atualizá-la de acordo com as alterações feitas.

Nos roteiros apresentados no Capítulo 3 desta tese serão dados maiores detalhes sobre o EPOS-R e EPOS-S, os quais foram as ferramentas mais utilizadas no desenvolvimento do projeto de software para a implementação do protocolo de sessão.

## 1.4 A Camada de Sessão.

A Camada de sessão, em conjunto com as camadas de apresentação e aplicação formam as "camadas superiores" do modelo de referência OSI, (como se observa na figura 1.2), as quais provêm serviços orientados ao usuário; para tal efeito, a camada de sessão usa um canal livre de erro, fornecido pela camada de transporte, e acrescenta elementos que são úteis para uma variedade de aplicações. Desta forma provê ferramentas (serviços) que podem ser negociadas entre os usuários-SS (usuários do Serviço de Sessão) comunicantes, além de fazer-se responsável pelo uso desses serviços acessados por primitivas.

A camada de sessão não controla a forma de uso das primitivas, essa tarefa é responsabilidade do usuário, que determina, baseado na aplicação, quais primitivas a usar, quando utilizá-las e como tirar proveito delas. Por exemplo, a camada de sessão fornece primitivas para inserir pontos de sincronismo na transmissão de um fluxo de dados, mas o usuário determinará como fará uso deles.

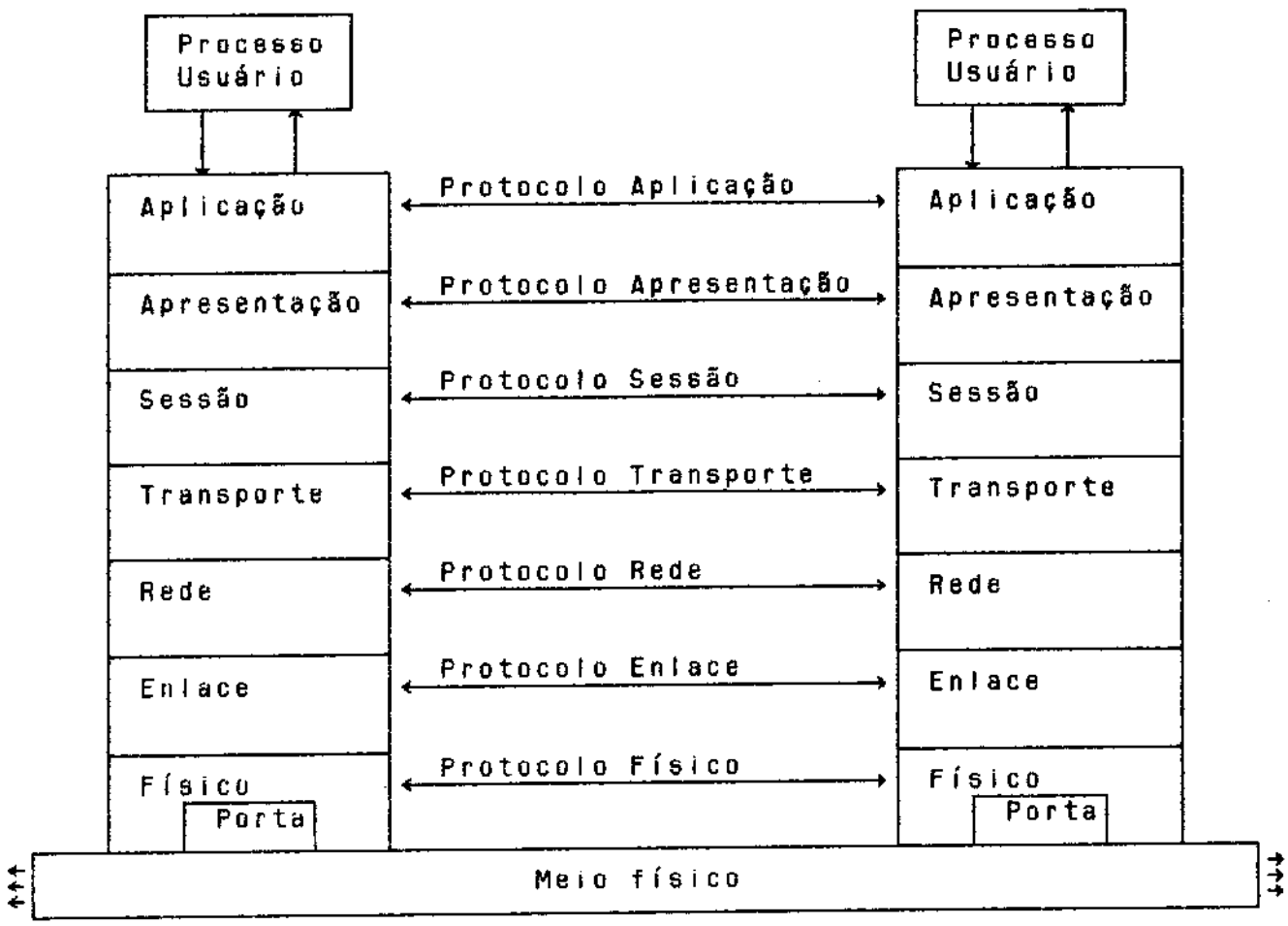



Figura 1.2: O Modelo de Referência OSI/ISO.

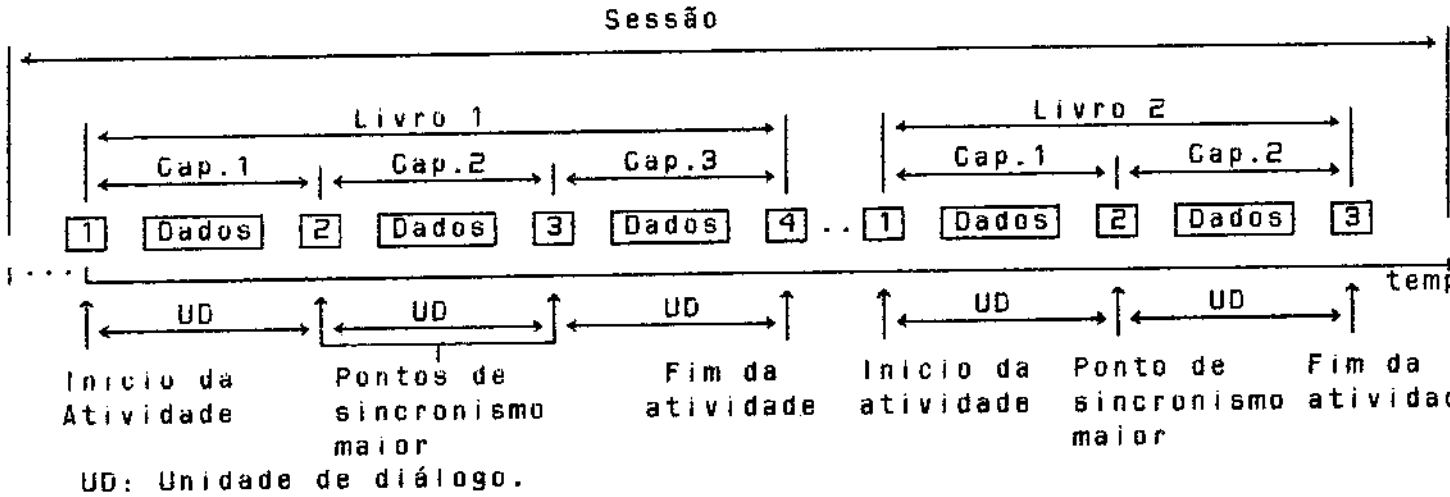
Do ponto de vista qualitativo [13], pode-se dizer que os serviços podem agrupar-se da seguinte forma:

(a) Intercâmbio de dados
(b) Sincronização.
(c) Relatório de exceção.
(d) Gerenciamento de direitos.

(a) Intercâmbio de dados: Este serviço básico da camada de sessão considera três etapas bem definidas: estabelecimento da conexão (lógica) entre dois usuários-SS comunicantes, troca de dados e desligamento da conexão. Ela não suporta a difusão de dados ("broadcasting"), mas atualmente está-se estudando na ISO tal possibilidade. A troca de dados pode ser "half-duplex" ou "full- duplex", por negociação durante a fase de estabelecimento da conexão.

(b) Sincronização: Este serviço permite ao usuário inserir pontos de sincronismo no fluxo de dados que se está transmitindo com o objetivo de possibilitar a recuperação, face a falhas de transmissão, a partir de um ponto anterior, evitando-se assim recomeçar a transmissão desde o seu início. Dois tipos de pontos de sincronismo podem ser usados: ponto de sincronismo maior e ponto de sincronismo menor. Sua utilização implica na definição dos conceitos de "atividade"/"unidade de diálogo", marcados por pontos de sincronismo maior/menor através da primitiva correspondente.

Considere-se o seguinte exemplo (Figura 1.3): para a transmissão do conteúdo de dois livros para um mesmo usuário, estabelece-se a conexão de sessão com o envio do conteúdo de cada livro em "atividades" distintas. O usuário receptor é avisado mediante a inserção de um ponto de sincronismo maior no início da transmissão (com a primitiva S-ACTIVITY-STARTreq/ind) e outro no final da transmissão (com a primitiva S-ACTIVITY-ENDreq/ind) de cada livro. Além disso, como o conteúdo dos livros está estruturado em capítulos, inseriram-se pontos de sincronismo maior (com a primitiva S-SYNC-MAJOR), avisando-se o usuário receptor o início e fim de um capítulo, (uma unidade de diálogo).



UD: Unidade de diálogo.

Figura 1.3: Atividades e unidades de diálogo na transmissão de dados sobre uma conexão de sessão.

Portanto, quando acontecer uma falha na transmissão terá que retransmitir-se só o capítulo que estava sendo enviado quando aconteceu a falha. Também podem inserir-se pontos de

sincronismo menor para envio página a página, normalmente tal mecanismo não é utilizado porque aumenta a "sobrecarga" (overhead) de dados transmitidos. Este tipo de sincronismo é interesante, por exemplo, na transmissão segura de mensagens ou comandos não estruturados de forma complexa.

O fato de utilizar o conceito de atividade permite também a possibilidade de interrompê-la para dar lugar na mesma conexão, a outra atividade normalmente curta no tempo comparada com a interrompida, continuando-se posteriormente a atividade original a partir do ponto em que ocorreu a interrupção. Por exemplo, durante a transmissão do conteúdo de um livro, precisa-se enviar uma mensagem ao mesmo usuário receptor através de correio eletrônico usando a mesma conexão (Figura 1.4).

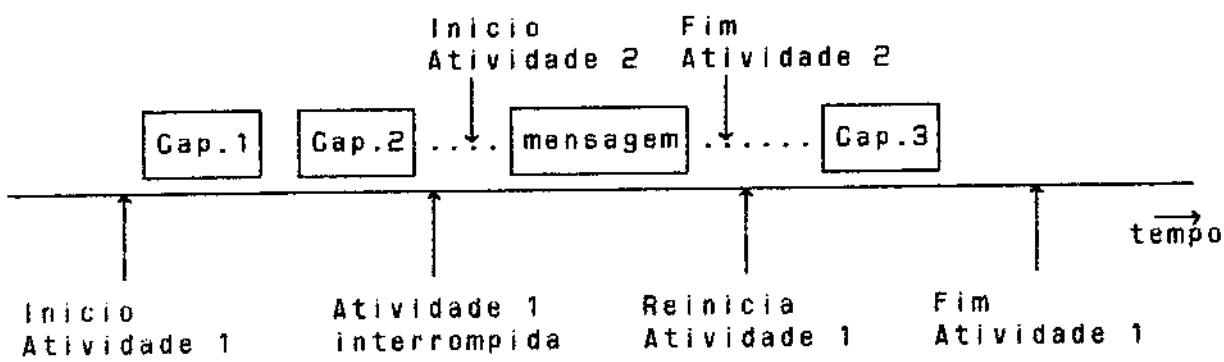



Figura 1.4 : Interrupção de uma atividade por outra.

(c) Relatório de exceção: Quando ocorre num dos usuários algum problema e é necessário a comunicação ao outro usuário, poderá usar-se o serviço (se estiver disponível) do "relatório de exceção" através da primitiva S-U-EXCEPTION-REPORT, onde o campo "user data" explica os fatos acontecidos. Será responsabilidade do outro usuário determinar o que fazer em tal situação. Também, a camada de sessão poderá utilizar este serviço, através da primitiva S-P-EXCEPTION-REPORT, quando acontecer algum problema dentro da camada de sesão ou nas camadas inferiores.

(d) Gerenciamento dos direitos (tokens): Normalmente neste protocolo é essencial que ambos os usuários comunicantes não tentem a mesma operação ao mesmo tempo (por exemplo, tentar iniciar uma atividade para a transmissão de um arquivo). Para evitar esses

tipos de problemas, a camada de sessão fornece "tokens" que podem ser intercambiados entre os usuários. Quem possue o "token" tem o "direito de" iniciar uma ação específica (no exemplo anterior, o usuário que tenha o "token" de atividade/maior poderá iniciar a transmissão do arquivo e se não tinha tal "token", deverá solicitá-lo ao outro usuário).

Cada "token" está relacionado com algum tipo de atividade. Para isso, a camada de sessão fornece as primitivas necessárias para gerenciar quatro tipos de "tokens":

(1) Token de dados.

(2) Token de liberação.

(3) Token de sincronização-menor.

(4) Token de sincronização maior e de atividade.

# CAPÍTULO 2: A CAMADA DE SESSÃO.

A camada de sessão corresponde à camada 5 no modelo de referência da OSI/ISO. Seu comportamento é estabelecido pelos documentos "Basic Connection Oriented Session Service Definition" (ISO 8326)[14] e "Basic Connection Oriented Session Protocol Specification" (ISO 8327)[15].

## 2.1. Funções da camada de sessão.

A camada de sessão tem, como função básica, estabelecer uma conexão lógica com outro usuário do Serviço de Sessão (Usuário-SS), permitir a troca de dados de forma sincronizada entre tais Usuários-SS, e liberar a conexão de uma forma ordenada.

Tem as seguintes funções adicionais, as quais podem ser aplicadas segundo negociação:

- Negociar o uso de "tokens" para a troca de dados, sincronização e liberação da conexão, e determinar se a troca de dados será half-duplex ou duplex;
- Estabelecer pontos de sincronização no diálogo e, no caso de ocorrência de erros, reiniciar o diálogo a partir de um ponto de sincronização pré-definido;
- Interromper um diálogo e reiniciá-lo posteriormente num ponto pré-estabelecido.

O protocolo considerado na camada opera no modo com conexão.

## 2.2. Definição de conceitos utilizados.

No contexto da camada faz-se uso dos conceitos de "Token", unidades de diálogo, atividades, sincronização, resincronização e qualidade de serviço (QOS), explicados a seguir.

### 2.2.1. Token.

Numa conexão de sessão, o "token" é um atributo ou direito que é entregue só a um usuário-SS no tempo, permitindo-lhe invocar e fazer uso exclusivo de certos serviços.

Definem-se quatro "tokens":

a) Token de dados:

b) Token de liberação;

c) Token de sincronização-menor;

d) Token de sincronização maior e de atividade.

Um "token" pode estar em um de dois estados:

-Não-disponível para nenhum usuário-SS, isto é, nenhum usuário-SS têm o direito exclusivo do serviço: em tal caso o serviço, ou está disponível para ambos os usuários-SS (transferência de dados e liberação) ou não está disponível para nenhum dos usuários-SS (sincronização e atividades).

-Disponível, caso em que sempre estará:

a) Alocado a um usuário-SS, que possui o direito de usar o serviço associado;

b) Não-alocado ao outro usuário-SS, que não tem o direito de usar o serviço, mas pode adquiri-lo posteriormente.

### 2.2.2. Sincronização e unidades de diálogo.

A transferência de dados pode ser compactada pelo usuário-SS em unidades de diálogo, abrangendo cada unidade de diálogo um conjunto de dados lógicamente relacionados (por exemplo, os capítulos de um livro podem formar unidades de diálogo). Para isso, deve-se inserir pontos de sincronização feitos pelo usuário-SS que transmite os dados. Estes pontos de sincronização podem ser de dois tipos:

-Pontos de sincronização maior, os quais definem o início e fim de uma unidade de diálogo, e que devem ser confirmados explícitamente pelo usuário-SS receptor;

-Pontos de sincronização menor, os quais podem ou não ser usados dentro de uma unidade de diálogo para estruturar de forma mais fina a transferência de dados, e que podem ou não ser explícitamente confirmados.

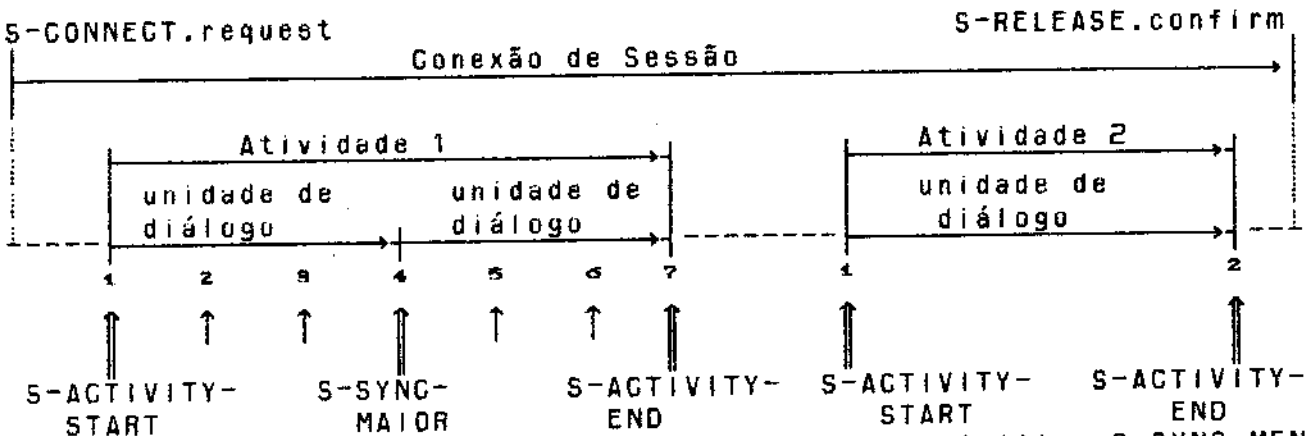
O objetivo dos pontos de sincronização é permitir uma recuperação da transmissão dos dados, quando aconteça uma falha, retransmitindo-se somente a partir do ponto de sincronismo anterior ao momento de ocorrência da falha.

2.2.3. Atividade.

Este conceito permite aos usuários-SS distinguir,dentro de uma ou várias conexões de sessão, um pacote de dados (formado por uma ou mais unidades de diálogo) que é lógicamente distinto de outro pacote de dados (em analogia com um documento ou arquivo específico distinto de outro).

Poderão existir várias atividades consecutivas (só uma no tempo) dentro de uma conexão de sessão: uma atividade poderá ser interrompida e logo reiniciada na mesma ou em subseqüentes conexões de sessão, mecanismo que pode ser considerado como uma forma de resincronização.

Na figura 2.1 definem-se duas atividades: a atividade 1 é estruturada em duas unidades de diálogo, as quais tem inseridos pontos de sincronismo menor. A atividade 2 é estruturada por uma unidade de diálogo e não tem pontos de sincronismo menor. Os números indicam os valores dos pontos de sincronismo.




↑ = Ponto de sincronização menor, utilizando a primitiva S-SYNC-MENOR.

⇑ = Ponto de sincronização maior.

Figura 2.1: Relação entre Unidade de Diálogo, Atividade e Pontos de Sincronização.

2.2.4. Resincronização.

A resincronização pode ser iniciada por qualquer dos dois usuários-SS comunicantes, no caso de algum erro na transmissão de dados. Define um novo estado para a fase de conexão e implica na realocação de "tokens" e estabelecimento de novos valores numéricos para os pontos de sincronização. Ao acontecer a resincronização, os dados ainda não liberados pela camada são eliminados.

2.2.5. Qualidade de Serviço - QOS(Quality of Service).

Refere-se as características desejáveis de uma conexão de sessão e que, em parte, são negociadas entre os usuários-SS. Na verdade, considera parâmetros de qualidade que a conexão de transporte deverá satisfazer. Estes parâmetros são transportados pelas TPDUs (e não pelas SPDUs), mas são negociados entre os provedores-SS comunicantes durante a fase de estabelecimento da conexão de transporte (a atividade inicial da fase de estabelecimento da conexão de sessão).

Existem duas classes de parâmetros QOS:

a) os negociados durante a fase de estabelecimento da conexão de sessão:

1) Proteção da conexão de sessão.
2) Prioridade da conexão de sessão.
3) Taxa de erro residual.
4) Vazão, para cada sentido de transferência.
5) Atraso de trânsito para cada sentido de transferência.
6) Transferência de diálogo otimizada.
7) Controle estendido.

b) os não negociados durante a fase de estabelecimento da conexão de sessão, mas com valores selecionados e/ou conhecidos por outros métodos (por exemplo, conhecimento e acordo prévio, ou por meio de funções de gerenciamento):

1) Atraso no estabelecimento de uma conexão de sessão.
2) Probabilidade de falha no estabelecimento de uma conexão de sessão.
3) Probabilidade de falha de uma transferência.
4) Atraso na liberação de uma conexão de sessão.
5) Probabilidade de falha na liberação de uma conexão de sessão.
6) Resilência ou rejeição da conexão de sessão.

Em geral, o método de negociação dos parâmetros envolve os seguintes passos:

Quando o usuário-SS chamante propõe os valores dos parâmetros QOS negociáveis, estes são passados através de TPDUs ao lado chamado onde, em primeira instância, o provedor-SS chamado aceita/modifica ou rejeita os valores (e com isso, aceita ou

rejeita a conexão de transporte associada à fase de estabelecimento da conexão de sessão). Se aceita, então o usuário-SS chamado deve decidir se aceita/modifica ou rejeita os valores (e com isso, aceita ou rejeita o estabelecimento da conexão de sessão) dos parâmetros QOS propostos.

2.3. Negociação.

Durante a fase de estabelecimento da conexão de sessão é realizada, entre ambos os usuários-SS, a negociação de diversos elementos e características da futura conexão, entre outros:

1) Unidades Funcionais.

2) Alocação inicial de "tokens", sempre que se tenha selecionado uma unidade funcional que opere com "token".

3) Inicialização do número de série do ponto de sincronização, se as unidades funcionais relacionadas com sincronização forem escolhidas.

4) Parâmetros QOS.

2.4. Fases e Serviços da Camada de Sessão.

A camada de sessão oferece 21 serviços diferentes, os quais podem ser executados utilizando até 62 primitivas, e fazendo uso de 34 SPDUs, (Tabela 2.1).

O serviço de sessão abrange as seguintes três fases sequenciais:

-Fase de estabelecimento da conexão de sessão.

-Fase de transferência de dados.

-Fase de liberação da conexão de sessão.

Em cada fase têm-se os seguintes serviços:

2.4.1. Fase de estabelecimento da conexão de sessão.

Serviço de conexão: É um serviço do tipo confirmado que tem como função estabelecer a conexão de sessão entre dois usuários-SS e negociar "tokens" e parâmetros que serão usados na conexão.

2.4.2.Fase de transferência de dados.

Durante esta fase, são trocados dados entre os usuários-SS sobre a conexão estabelecida. As SSDUs (Session Service Data Unit) são transportadas entre os usuários-SS através das SPDUs (Session Protocol Data Unit).

-Transferência de Dados:

Tem-se os seguintes quatro serviços associados com a transferência de dados propriamente dita:

a) Serviços de transferência normal de dados que permitem a troca de SSDUs de dados normais, podendo operar de dois modos:

1) half-duplex, quando o direito de enviar dados pertence a quem possue o "token" dado.

2) duplex, quando não há restrições sobre o direito de enviar dados.

Este serviço permite tambem a segmentação e concatenação de dados.

b) Transferência de dados urgentes (expedited data), para transferência de uma quantidade limitada de dados (até 14 octetos), independentemente do estado dos "tokens", através da opção urgente de transporte.

c) Transferência de dados tipificados (typed data), que permite transferir dados independentemente do estado do "token" dado, usando a mesma rota de transferência dos dados normais.

d) Transferência de dados fora de atividade (capability data), que permite a transferência limitada de dados (até 512 octetos) quando não há nenhuma atividade em progresso. O direito de enviar tais dados pertence a quem possue o direito de iniciar a atividade seguinte. Utiliza a mesma rota de dados normais.

-Transferência de Tokens:

Existem três serviços relacionados com o gerenciamento dos "tokens":

e) Entrega de tokens, que permite a um usuário-SS entregar um ou mais "tokens" ao outro usuário-SS.

f) Solicitação de tokens, que permite a um usuário-SS requisitar um ou mais "tokens" do outro usuário-SS.

g) Entrega de controle, que permite a um usuário-SS entregar o controle de todos os "tokens" disponíveis ao outro usuário-SS.

-Sincronização e resincronização:

h) Ponto sincronização menor, o qual permite ao usuário-SS separar o fluxo de dados (normais e tipificados), ou seja, separar os dados que foram transmitidos antes da invocação do serviço, dos dados que serão transmitidos posteriormente. Seu uso é controlado

pelo "token" de sincronização menor.

i) Ponto sincronização maior, o qual permite ao usuário-SS definir secções de um fluxo de dados (normais, tipificados ou urgentes) a serem transmitidas em cada direção como unidades de diálogo. Seu uso é controlado pelo "token"- maior/atividade.

j) Servico de resincronização, através do qual o usuário-SS coloca a conexão de sessão num ponto de sincronização (prévio ou novo) e realoca os "tokens" disponíveis. Este serviço pode causar a perda de SSDUs (do tipo normal, tipificado ou urgente).

-Informação de erros e situações de excepção:

k) Informação de excecões feita pelo provedor-SS, que permite informar ao usuário-SS as condições de exceção ou de erros no protocolo do provedor-SS. Este serviço pode causar a perda de SSDUs (normais, tipificados e urgentes), e só pode ser usado enquanto uma atividade estiver em progresso.

l) Informação de excecões feita pelo usuário-SS, que permite ao usuário-SS informar uma condição de exceção quando o "token" dado está disponível, mas não alocado ao usuário-SS. O uso deste serviço pode causar a perda das SSDUs (normais, tipificados e urgentes) e só poderá ser usado enquanto uma atividade estiver em progresso.

-Gerenciamento de atividades:

m) Servico de início de atividades, o qual permite ao usuário-SS indicar que uma nova atividade vai começar. Seu uso é controlado pelo "token" de maior/atividade e só pode ser iniciado se não existir nenhuma atividade em progresso.

n) Servico de reinício de atividade, que permite ao usuário-SS reiniciar uma atividade que foi previamente suspensa. Seu uso é controlado pelo "token" de maior/atividade. Este serviço só pode ser executado se não houver atividade em progresso, e pode reiniciar uma atividade numa conexão de sessão suspensa ou noutra conexão qualquer.

o) Servico de interrupção de atividade, o qual permite ao usuário-SS, interromper uma atividade sem descartar os dados anteriormente transferidos, e a ser posteriormente reiniciada. Seu uso é controlado pelo "token" de maior/atividade; pode causar a perda de SSDUs (normais, tipificadas e urgentes) que estejam no

"buffer" do provedor-SS, prontos a ser transferidos, antes da interrupção. Este serviço só pode ser executado se uma atividade estiver em progresso.

p) Serviço descartador de atividade, com o qual o usuário-SS termina, de forma abruta, uma atividade em progresso, descartando todos os dados transferidos até o momento de acontecer o término anormal da atividade, que não será reiniciada. Seu uso é controlado pelo "token" maior/atividade, e pode causar a perda de SSDUs .

q) Serviço de finalização de atividade, que permite ao usuário-SS finalizar uma atividade (e iniclalizar o ponto de sincronização maior para uma próxima atividade). Este serviço só pode ser invocado se uma atividade está em progresso e seu uso é controlado pelo "token" maior/atividade.

2.4.3. Fase de liberação da conexão de sessão.

Esta fase tem três serviços a ela associados:

a) Serviço de liberação ordenada, o qual permite a qualquer dos usuários-SS comunicantes executar uma liberação ordenada da conexão de sessão sem perda de dados. Pode ser de dois tipos:

-Negociada, em que o "token" de liberação deve estar disponível, e o lado que receber o requerimento tem o direito de aceitar ou rejeitar (neste último caso, a conexão de sessão continua). Em ambos os casos não há perda de dados.

-Não negociada, o "token" liberação não está disponível e o usuário-SS requesitado deverá aceitar o pedido de liberação.

b) Serviço de aborto iniciado pelo usuário, que permite a qualquer dos usuários-SS a liberação da conexão de sessão, em qualquer instante, informando o outro usuário-SS. O uso deste serviço, poderá causar a perda de dados ainda não transferidos.

c) Serviço de aborto iniciado pelo provedor, em que o provedor-SS indica a ambos os usuários-SS que, por razões internas, aborta a conexão de sessão. Este serviço pode causar a perda de dados ainda não transferidos ao outro usuário-SS.

Os serviços e primitivas de serviço associadas são apresentados na Tabela 2.1. Na Tabela 2.2 apresentam-se os parâmetros associados a cada primitiva de serviço.

| Serviço | Primitivas | SPDUs Associadas |
|---|---|---|
| Conexão de Sessão | S-CONNECT.request<br>S-CONNECT.indication<br>S-CONNECT(accept).response<br>S-CONNECT(accept).confirm<br>S-CONNECT(reject).response<br>S-CONNECT(reject).confirm | CONNECT (CN)<br>CONNECT (CN)<br>ACCEPT (AC)<br>ACCEPT (AC)<br>REFUSE (RF)<br>REFUSE (RF) |
| Transferência Normal Dados | S-DATA.request<br>S-DATA.indication | DATA TRANSFER (DT)<br>DATA TRANSFER (DT) |
| Transf. Dados Urgentes | S-EXPEDITED-DATA.request<br>S-EXPEDITED-DATA.indication | EXPEDITED DATA (DT)<br>EXPEDITED DATA (DT) |
| Transf. Dados Typed | S-TYPED-DATA.request<br>S-TYPED-DATA.indication | TYPED DATA (TD)<br>TYPED DATA (TD) |
| Transf. Dados Fora de Atividade | S-CAPABILITY-DATA.request<br>S-CAPABILITY-DATA.indicat.<br>S-CAPABILITY-DATA.response<br>S-CAPABILITY-DATA.confirm | CAPABILITY DATA (CD)<br>CAPABILITY DATA (CD)<br>CAPABILITY DATA ACK(CDA)<br>CAPABILITY DATA ACK(CDA) |
| Entrega de Tokens | S-TOKEN-GIVE.request<br>S-TOKEN-GIVE.indication | GIVE TOKENS (GT)<br>GIVE TOKENS (GT) |
| Solicitude de Tokens | S-TOKEN-PLEASE.request<br>S-TOKEN-PLEASE.indication | PLEASE TOKENS (PT)<br>PLEASE TOKENS (PT) |
| Entrega Controle | S-CONTROL-GIVE.request<br>S-CONTROL-GIVE.indication | GIVE TOKENS CONFIRM(GTC)<br>GIVE TOKENS CONFIRM(GTC)<br>GIVE TOKENS ACK (ACK) |
| Ponto de Sincronizaç. Menor | S-SYNC-MINOR.request<br>S-SYNC-MINOR.indication<br>S-SYNC-MINOR.response<br>S-SYNC-MINOR.confirm | MINOR SYNC POINT (MIP)<br>MINOR SYNC POINT (MIP)<br>MINOR SYNC ACK (MIA)<br>MINOR SYNC ACK (MIA) |
| Ponto de Sincronizaç. Maior | S-SYNC-MAJOR.request<br>S-SYNC-MAJOR.indication<br>S-SYNC-MAJOR.response<br>S-SYNC-MAJOR.confirm | MAJOR SYNC POINT (MAP)<br>MAJOR SYNC POINT (MAP)<br>MAJOR SYNC ACK (MAA)<br>MAJOR SYNC ACK (MAA) |
| Resincroni-zação | S-RESYNCHRONIZE.request<br>S-RESYNCHRONIZE.indication<br>S-RESYNCHRONIZE.response<br>S-RESYNCHRONIZE.confirm | RESYNCHRONIZE (RS)<br>RESYNCHRONIZE (RS)<br>RESYNCHRONIZE ACK (RA)<br>RESYNCHRONIZE ACK (RA) |
| Info.Exceç.-P | S-P-EXCEPTION-REPORT.ind. | EXCEPTION REPORT (ER) |

Tabela 2.1: Serviços, primitivas de serviço e SPDUs associados.

| Serviço | Primitivas | SPDUs Associadas |
|---|---|---|
| Info.Exceç.-U | S-U-EXCEPTION-REPORT.req.<br>S-U-EXCEPTION-REPORT.ind. | EXCEPTION DATA (ED)<br>EXCEPTION DATA (ED) |
| Inicio de Atividade | S-ACTIVITY-START.request<br>S-ACTIVITY-START.indicat. | ACTIVITY START (AS)<br>ACTIVITY START (AS) |
| Reinicio de Atividade | S-ACTIVITY-RESUME.request<br>S-ACTIVITY-RESUME.indicat. | ACTIVITY RESUME (AR)<br>ACTIVITY RESUME (AR) |
| Interrupção de Atividade | S-ACTIVITY-INTERRUPT.req.<br>S-ACTIVITY-INTERRUPT.ind.<br>S-ACTIVITY-INTERRUPT.resp.<br>S-ACTIVITY-INTERRUPT.conf. | ACTIVITY INTERRUPT (AI)<br>ACTIVITY INTERRUPT (AI)<br>ACTIVITY INTERRUPT ACK(AIA)<br>ACTIVITY INTERRUPT ACK(AIA) |
| Descartador de Atividade | S-ACTIVITY-DISCARD.request<br>S-ACTIVITY-DISCARD.indic.<br>S-ACTIVITY-DISCARD.respon.<br>S-ACTIVITY-DISCARD.confirm | ACTIVITY DISCARD (AD)<br>ACTIVITY DISCARD (AD)<br>ACTIVITY DISCARD ACK(ADA)<br>ACTIVITY DISCARD ACK(ADA) |
| Finalização de Atividade | S-ACTIVITY-END.request<br>S-ACTIVITY-END.indication<br>S-ACTIVITY-END.response<br>S-ACTIVITY-END.confirm | ACTIVITY END (AE)<br>ACTIVITY END (AE)<br>ACTIVITY END ACK(AEA)<br>ACTIVITY END ACK(AEA) |
| Liberação Ordenada | S-RELEASE.request<br>S-RELEASE.indication<br>S-RELEASE(accept).response<br>S-RELEASE(accept).confirm<br>S-RELEASE(reject).response<br>S-RELEASE(reject).confirm | FINISH (FN)<br>FINISH (FN)<br>DISCONNECT (DN)<br>DISCONNECT (DN)<br>NOT FINISHED (NF)<br>NOT FINISHED (NF) |
| Aborto-U | S-U-ABORT.request<br>S-U-ABORT.indication | ABORT (AB)<br>ABORT (AB)<br>ABORT ACCEPT (AA) (1) |
| Aborto-P | S-P-ABORT.indication | ABORT (AB) |

(1) AA: A utilização desta SPDU é mostrada na figura 2.4. Ela não está relacionada diretamente com nenhuma primitiva.

Tabela 2.1:(continuação).

| Serviço | Parâmetros | Tipo de Primitiva | | | |
|---|---|---|---|---|---|
| | | req. | ind. | resp. | conf. |
| S-CONNECT | Identificador de conexão | M | M(=) | M | M(=) |
| | Endereço do SSAP chamante | M | M(=) | | |
| | Endereço do SSAP chamado | M | M(=) | M(=) | M(=) |
| | Resultado (1) | | | M | M(=) |
| | QOS(Qualidade de Serviço) | M | M | M | M |
| | Requeriment. de sessão(2) | M | M(=) | M | M(=) |
| | No de serie inicial do ponto de sincronização | C | C(=) | C | C(=) |
| | Alocação inicial de tokens | C | C(=) | C | C(=) |
| | Dados do usuario-SS | U | C(=) | U | C(=) |
| S-DATA | Dados do usuario-SS | M | M(=) | X | X |
| S-EXPEDITED-DATA | Dados do usuario-SS | M | M(=) | X | X |
| S-TYPED-DATA | Dados do usuario-SS | M | M(=) | X | X |
| S-CAPABILITY-DATA | Dados do usuario-SS | U | C(=) | U | C(=) |
| S-TOKEN-GIVE | Tokens | M | M(=) | X | X |
| S-TOKEN-PLEASE | Tokens | M | M(=) | X | X |
| | Dados do usuario-SS | U | C(=) | X | X |
| S-CONTROL-GIVE | Não tem parâmetros | | | X | X |
| S-SYNC-MINOR | Tipo (3) | M | M(=) | | |
| | No serie do ponto sincro. | M | M(=) | M | M(=) |
| | Dados do usuario-SS | U | C(=) | U | C(=) |
| S-SYNC-MAJOR | No serie do ponto sincro. | M | M(=) | | |
| | Dados do usuario-SS | U | C(=) | U | C(=) |
| S-RESYNCHRONIZE | Tipo de resincronizaç.(4) | M | M(=) | | |
| | No de serie do ponto de sincronização | C | M(=) | C | M(=) |
| | Tokens | C | M(=) | C | M(=) |
| | Dados do usuario-SS | U | C(=) | U | C(=) |
| S-P-EXCEPTION-REPORT | Razões (5) | X | M | X | X |
| S-U-EXCEPTION-REPORT | Razões (6) | M | M(=) | X | X |
| | Dados do usuario-SS | U | C(=) | X | X |

Tabela 2.2: Os serviços e seus parâmetros (continua na seguinte página).

| Serviço | Parâmetros | Tipo de Primitiva | | | |
|---|---|---|---|---|---|
| | | req. | ind. | resp. | conf. |
| S-ACTIVITY-START | Identif. de atividade (7)<br>Dados do usuario-SS | M<br>U | M(=)<br>C(=) | X<br>X | X<br>X |
| S-ACTIVITY-RESUME | Identif. de atividade (8)<br>Identif. de atividade original (9)<br>No serie do ponto sincro.<br>Identificador da conexão de sessão original (10)<br>Dados do usuario-SS | M<br>M<br>M<br>U<br>U | M(=)<br>M(=)<br>M(=)<br>C(=)<br>C(=) | X<br>X<br>X<br>X<br>X | X<br>X<br>X<br>X<br>X |
| S-ACTIVITY-INTERRUPT | Razões (11) | M | M(=) | X | X |
| S-ACTIVITY-DISCARD | Razões (11) | M | M(=) | X | X |
| S-ACTIVITY-END | No serie do ponto sincro.<br>Dados do usuario-SS | M<br>U | M(=)<br>C(=) | <br>U | <br>C(=) |
| S-RELEASE | Resultado (12)<br>Dados do usuario-SS | <br>U | <br>C(=) | M<br>U | M(=)<br>C(=) |
| S-U-ABORT | Dados do usuario-SS (13) | U | C(=) | X | X |
| S-P-ABORT | Razões (14) | X | M | X | X |

Legenda:

X: Indica que o tipo de primitiva não existe para tal serviço.

M: A presença do parâmetro é mandatória.

C: A presença do parâmetro é condicional.

U: A presença do parâmetro é opção do usuário.

espaço: Parâmetro ausente.

(=): O valor do parâmetro é idéntico ao valor do parâmetro correspondente na primitiva-SS precedente.-

Tabela 2.2: Os serviços e seus parâmetros.

Notas explicativas sobre a Tabela 2.2:

(1) Resultado: Indica o sucesso ou falha da requesição de estabelecimento de conexão. Seu valor pode ser:

a) aceite;

b) rejeitado pelo usuário-SS chamado;

c) rejeitado pelo provedor-SS.

(2) Requisitos de sessão: Lista de unidades funcionais negociadas.

(3) Tipo: Indica se o usuário-SS requisita ou não confirmação explícita, transparente ao provedor-SS, com os valores:

a) explícito;

b) opcional.

(4) Tipo de ressincronização: Especifica uma das opções de ressincronização, com os valores:

a) abandonar;

b) reiniciar;

c) setar.

(5) Razões: Especifica a razão para o relatório de exceção:

a) erro de protocolo;

b) erro não especificado.

(6) Razões: Especifica a razão para o relatório de exceção, e é transparênte ao provedor-SS, com os valores:

a) o usuário-SS recebe dados que não pode manusear de forma correta;

b) erro do usuário-SS local;

c) erro de seqüência;

d) demanda do "token" dado;

e) erro irrecuperável no procedimento;

f) erro não especificado.

(7) Identificador de atividade: Permite ao usuário-SS identificar a nova atividade e é transparente ao provedor-SS. Tem um máximo de 6 octetos.

(8) Identificador de atividade: Permite ao usuário-SS dar um novo identificador à atividade que está sendo reiniciada. é transparente ao provedor-SS.

(9) Identificador de atividade original: Especifica a identificação original da atividade que está sendo reiniciada. é transparente ao provedor-SS.

(10) Identificador da conexão de sessão original: Especifica o identificador da conexão de sessão, na qual a atividade que está sendo reiniciada foi originalmente estabelecida. é transparente ao

provedor-SS.

(11) Razões: Especifica a razão para a interrupção/descarte da atividade: é transparente ao provedor-SS e as razões podem ser iguais às indicadas na nota (6).

(12) Resultado: Especifica se é ou não aceite a liberação. Pode ser:

a) afirmativa;

b) Negativa( só quando o "token" de liberação estiver disponível).

(13) Dados do usuario-SS: Contem 1 a 9 octetos de informação do usuário.

(14) Razões: Indica as razões do aborto:

a) desconexão de transporte;

b) erro de protocolo;

c) indefinido.

## 2.5. Unidades Funcionais e Subconjuntos.

### 2.5.1. Unidades funcionais.

Com o objetivo de facilitar a negociação dos serviços, o padrão internacional da ISO estabelece agrupamentos lógicos de serviços relacionados. Tais agrupamentos, denominados unidades funcionais, são negociados durante a fase de estabelecimento da conexão de sessão.

Na Tabela 2.3 apresenta-se a relação entre as unidades funcionais e os serviços, e na Tabela 2.4 os "tokens" usados pelas unidades funcionais.

Como se pode deduzir da Tabela 2.3, cada unidade funcional fica definida em seu comportamento pelos serviços que a compõem. Além disso:

a) A unidade funcional Kernel não é negociável,isto é, sempre deve estar presente em qualquer implementação, porque suporta os serviços básicos para estabelecimento da conexão, transferência de dados normais e liberação não negociada da conexão de sessão.

b) Não poderão selecionar-se as unidades funcionais Half-duplex e Duplex simultaneamente. Além disso, o "token"-dado deve estar disponível quando se escolhe Half-

duplex.

c) A unidade funcional Duplex não necessita de serviços adicionais, porque a transferência de dados pode ser feita em qualquer sentido da conexão de sessão, ou seja, ambos os usuários-SS têm o mesmo direito de acessar a conexão de sessão para transferir dados . Logo, não se requer o uso do "token"-dado.

d) A unidade funcional "Intercâmbio de Dados Fora de Atividade", só poderá ser selecionada se a unidade funcional Gerenciamento de atividades também tenha sido escolhida.

e) A unidade funcional de "Informação de Exceções" só poderá ser negociada em conjunto com a unidade funcional "Half-duplex".

| Unidade Funcional | Serviços |
|---|---|
| Kernel (não negociável) | Conexão de Sessão<br>Transferência de Dados Normais<br>Liberação ordenada<br>Aborto Iniciado pelo Usuario<br>Aborto Iniciado pelo Provedor |
| Liberação Ordenada | Liberação Ordenada<br>Entrega de Tokens<br>Solicitude de Tokens |
| Half-duplex | Entrega de Tokens<br>Solicitude de Tokens |
| Duplex | Nenhum serviço adicional |
| Dados Urgentes | Transferência de Dados Urgentes |
| Dados Tipos | Transferência de Dados Tipos |
| Transferência de Dados Fora de Atividade | Transferência de Dados Fora de Atividade |
| Sincronização Menor | Ponto Sincronização Menor<br>Entrega de Tokens<br>Solicitude de Tokens |
| Sincronização Maior | Ponto Sincronização Maior<br>Entrega de Tokens<br>Solicitude de Tokens |
| Resincronização | Resincronização |
| Exceções | Informação de Exceções do Provedor-SS<br>Informação de Exceções do Usuario-SS |
| Gerênciamento de Atividade | Inicio de Atividade<br>Reinicio de Atividade<br>Interrupção de Atividade<br>Descartador de Atividade<br>Finalização de Atividade<br>Entrega de Tokens<br>Solicitude de Tokens<br>Entrega Controle |

Tabela 2.3 : Serviços associados com cada Unidade Funcional.

| Unidade Funcional | Token |
|---|---|
| Liberação Ordenada | Token de liberação |
| Half-Duplex | Token dado |
| Sincronização Menor | Token Sincronização-menor |
| Sincronização Maior | Token maior/atividade |
| Gerênciamento de Atividade | Token maior/atividade |

Tabela 2.4: Unidades Funcionais que usam tokens.

2.5.2. Subconjuntos.

O padrão internacional da ISO define e recomenda as seguintes combinações de unidades:

a) Subconjunto Combinado Básico (BCS), o qual permite transferência normal de dados, liberação ordenada e opção entre half-duplex ou duplex.

O BCS comprende as seguintes unidades funcionais:

1) Kernel;

2) Half-duplex;

3) Duplex.

b) Subconjunto Sincronizado Básico (BSS), o qual permite transferência normal e tipificada de dados, liberação ordenada negociada, sincronização maior e menor, e resincronização.

O BSS abrange as seguintes unidades funcionais:

1) Kernel;

2) Liberação Negociada;

3) Half-duplex;

4) Dados tipificados;

5) Sincronização Menor;

6) Sincronização Maior;

7) Resincronização.

c) Subconjunto de Atividade Básica (BAS), o qual permite transferência de dados normais, tipificados e fora de atividades, liberação ordenada não negociada, sincronização menor, gerenciamento de atividades e exceções.

O BAS abrange as seguintes unidades funcionais:

1) Kernel;

2) Half-duplex;

3) Dados Tipificados;

4) Intercambio de Dados Fora de Atividade;

5) Sincronização Menor;

6) Execeções;

7) Gerenciamento de Atividades.

(Ver a Tabela 2.5 que relaciona as unidades funcionais com os subconjuntos e SPDUs).

| Unidade Funcional | Código SPDU | Nome da SPDU | BCS | BSS | BAS |
|---|---|---|---|---|---|
| Kernel | CN | CONNECT | M | M | M |
| | AC | ACCEPT | M | M | M |
| | RF | REFUSE | M | M | M |
| | FN | FINISH | M | M | M |
| | DN | DISCONNECT | M | M | M |
| | AB | ABORT | M | M | M |
| | AA | ABORT ACCEPT | R | R | R |
| | DT | DATA TRANSFER | M | M | M |
| Half-duplex | PT | PLEASE TOKENS | O | M | M |
| | GT | GIVE TOKENS | O | M | M |
| Exceções | ER | EXCEPTION REPORT | O | - | M |
| | ED | EXCEPTION DATA | O | - | M |
| Dados typed | TD | TYPED DATA | O | M | O |
| Liberação Negociada | FN | FINISH | O | M | - |
| | DN | DISCONNECT | O | M | - |
| | NF | NOT FINISHED | O | M | - |
| | PT | PLEASE TOKENS | O | M | - |
| | GT | GIVE TOKENS | O | M | - |
| Sincronização Menor | MIP | MINOR SYNC POINT | - | M | M |
| | MIA | MINOR SYNC ACK | - | M | M |
| | PT | PLEASE TOKENS | - | M | M |
| | GT | GIVE TOKENS | - | M | M |
| Sincronização Maior | MAP | MAJOR SYNC POINT | - | M | - |
| | MAA | MAJOR SYNC ACK | - | M | - |
| | PR | PREPARE | - | M | - |
| | PT | PLEASE TOKENS | - | M | - |
| | GT | GIVE TOKENS | - | M | - |
| Resincronização | RS | RESYNCHRONIZE | - | M | O |
| | RA | RESYNCHRONIZE ACK | - | M | O |
| | PR | PREPARE | - | M | - |
| Urgênte | EX | EXPEDITED DATA | O | - | - |

Notas:

M: Mandatorio

R: Recepção mandatoria; transmissão opcional

O: Opcional, pode ser usado se é implementado e negociado

-: Não é requisitado em esta unidade funcional

BCS: Basic Combinado Subset | BSS: Basic Synchronized Subset

BAS: Basic Activity Subset

Tabela 2.5: Relação entre Unidades Funcionais, SPDUs e Subconjuntos.

(continuação da Tabela 2.5)

| Unidade Funcional | Código SPDU | Nome da SPDU | BCS | BSS | BAS |
|---|---|---|---|---|---|
| Gerenciamento de Atividade | AS | ACTIVITY START | - | - | M |
| | AR | ACTIVITY RESUME | - | - | M |
| | AI | ACTIVITY INTERRUPT | - | - | M |
| | AIA | ACTIVITY INTERRUPT ACK | - | - | M |
| | AD | ACTIVITY DISCARD | - | - | M |
| | ADA | ACTIVITY DISCARD ACK | - | - | M |
| | AE | ACTIVITY END | - | - | M |
| | AEA | ACTIVITY END ACK | - | - | M |
| | PT | PLEASE TOKENS | - | - | M |
| | GT | GIVE TOKENS | - | - | M |
| | GTA | GIVE TOKENS ACK | - | - | M |
| | GTC | GIVE TOKENS CONFIRM | - | - | M |
| | PR | PREPARE | - | - | M |
| Transf. de Dados Fora de Atividade | CD | CAPABILITY DATA | - | - | M |
| | CDA | CAPABILITY DATA ACK | - | - | M |
| Uso de Transporte Urgênte | | | O | M | O |
| Uso de Duplex | | | O | O | O |

## 2.6. Protocolo da Camada de Sessão.

A Máquina de Protocolo de Sessão (SPM) comunica-se com o usuário-SS através dos SSAPs, utilizando as primitivas de serviço definidas pela ISO8326 e com a SPM-par através do intercâmbio de SPDUs, (utilizando os serviços de transporte).

### 2.6.1. Serviços utilizados pela SPM.

a) Para a fase de conexão de transporte, utilizam-se as seguintes primitivas de transporte:

T-CONNECT.request;

T-CONNECT.indication;

T-CONNECT.response;

T-CONNECT.indication;

T-DISCONNECT.request;

T-DISCONNECT.indication.

Durante esta fase, nenhuma SPDU é transportada.

Uma conexão de sessão pode ser alocada a uma conexão de transporte já existente ou criar-se uma nova. Além disso, poderá incluir-se a reutilização ou a criação de uma conexão de transporte

urgente, o que dependerá da seleção da unidade funcional "Urgente". Só o iniciador da conexão de transporte tem o direito de gerar e enviar a SPDU CONNECT, iniciando o processo de estabelecimento da conexão de sessão.

b) Para a fase de transporte de dados normais e SPDUs, a SPM faz uso das seguintes primitivas de transporte:

T-DATA.request;

T-DATA.indication.

As SPDUs que podem ser transportadas são as seguintes:

CONNECT;

ACCEPT;

REFUSE;

FINISH;

DISCONNECT;

NOT FINISHED;

DATA TRANSFER;

TYPED DATA;

CAPABILITY DATA;

CAPABILITY DATA ACK;

GIVE TOKENS;

PLEASE TOKENS;

GIVE TOKENS CONFIRM;

GIVE TOKENS ACK;

MINOR SYNC POINT;

MINOR SYNG ACK;

MAJOR SYNC POINT;

MAJOR SYNC ACK;

RESYNCHRONIZE;

RESYNCHRONIZE ACK;

EXCEPTION REPORT;

EXCEPTION DATA;

ACTIVITY START;

ACTIVITY RESUME;

ACTIVITY INTERRUPT;

ACTIVITY INTERRUPT ACK;

ACTIVITY DISCARD;

ACTIVITY DISCARD ACK;

ACTIVITY END;

ACTIVITY END ACK.

Se a rota de transporte urgente não estiver disponível, as seguintes SPDUs podem ser enviadas através da rota de transporte de dados normais:

ABORT e ABORT ACCEPT.

c) Para a fase de liberação é possível a reutilização da conexão de transporte. A SPM utiliza as seguintes primitivas do serviço de transporte:

T-DATA.request;

T-DATA.indication.

As SPDUs usadas para indicar a liberação e possível reutilização da conexão de transporte são as seguintes:

REFUSE;

FINISH;

DISCONNECT;

ABORT;

ABORT ACCEPT.

No caso de reuso da conexão de transporte existem duas possibilidades:

a) a SPM iniciante da conexão de transporte necessita reutilizá-la, o que será anunciado através das SPDUs: ABORT ou FINISH.

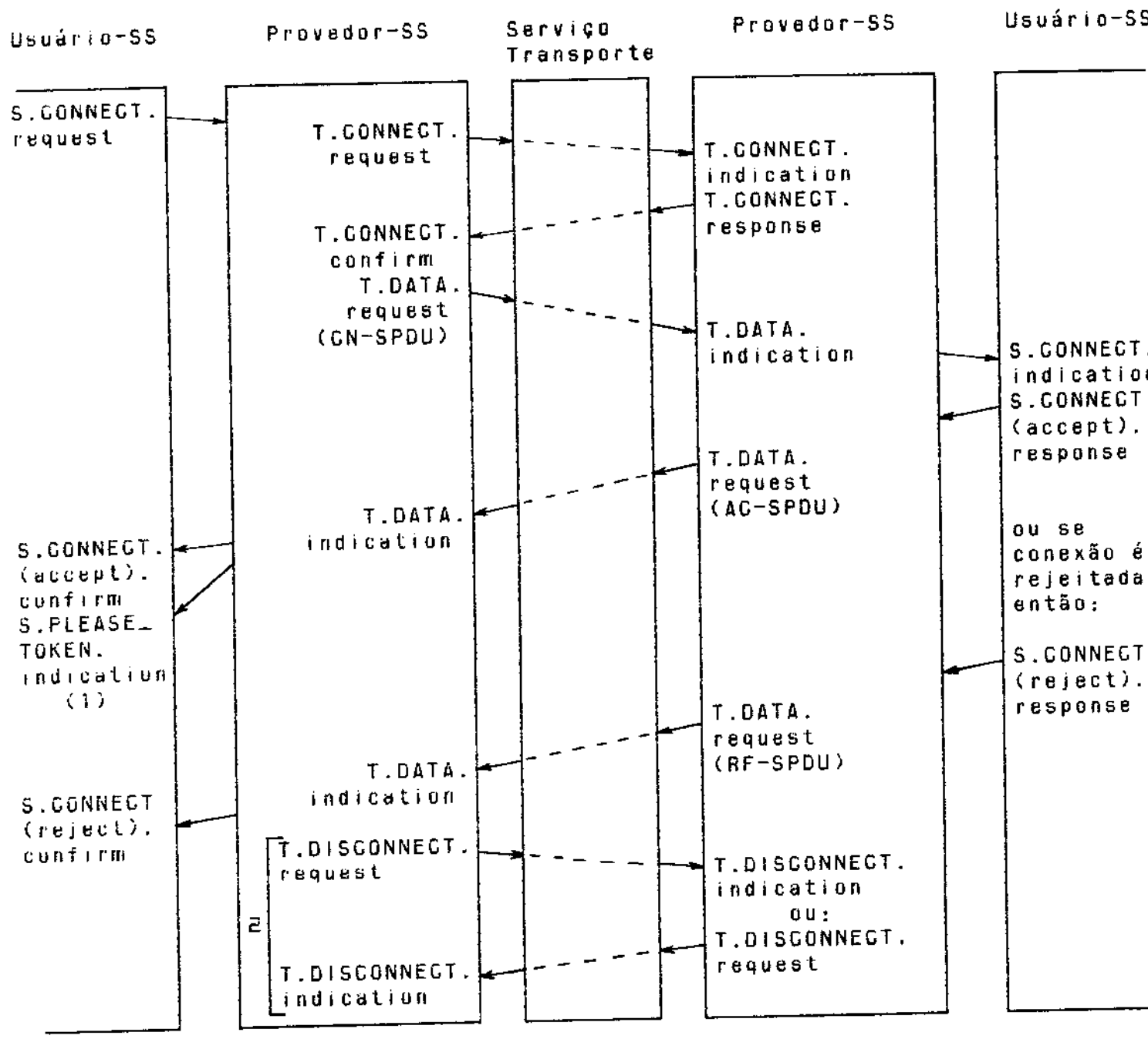
b) a SPM iniciante da conexão de transporte receba uma SPDU REFUSE ou ABORT, a qual indicará por parâmetro, que a conexão de transporte deve ser mantida.

Para evitar problemas de contenção de uma conexão de transporte retida, só o iniciador da conexão de transporte pode reutilizá-la, enviando uma SPDU CONNECT para estabelecer uma nova conexão de sessão.

2.6.2. Exemplos explicativos do Protocolo de Sessão.

Usando diagramas temporais apresentam-se, a seguir, casos típicos que podem acontecer no protocolo de sessão:

- Fase de estabelecimento da conexão com e sem sucesso (Figura 2.2).
- Liberação ordenada (aceite) sem reutilização da conexão de transporte (Figura 2.3).
- Aborto da conexão de sessão iniciado pela SPM na ocorrência de um erro de protocolo ou iniciado pelo usuário-SS, sem transporte urgente (Figura 2.4).
- Diferentes alternativas de transmissão de dados (Figura 2.5):
  a) Fluxo de dados normais.
  b) Transmissão de dados urgentes.
  c) Transmissão de dados "typed".



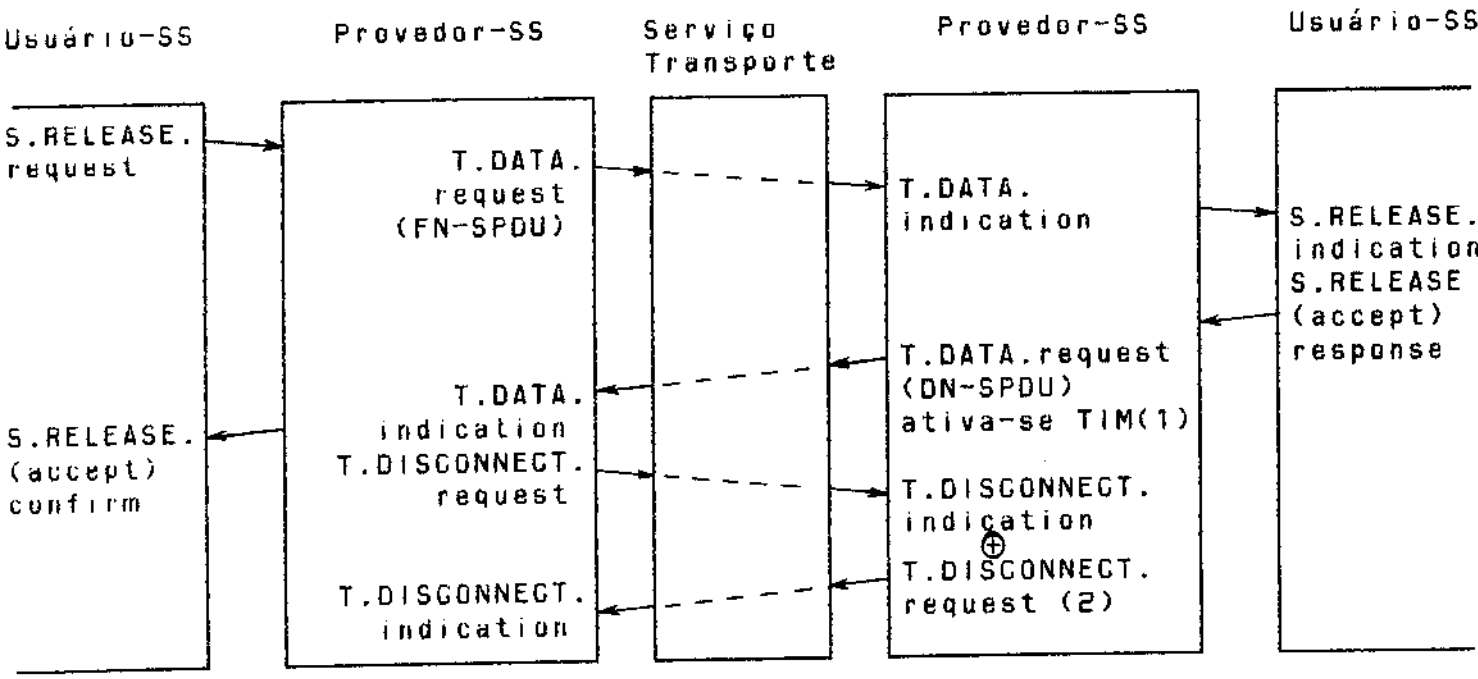

(1).Esta primitiva é gerada se o usuário do serviço de sessão chamad necessita de algum token indicado na SPDU ACCEPT.

(2).TIM: Disconnection and abort timer. Se a conexão de transporte não va ser utilizada, o timer ativa-se esperando a chegada de um T.DISCONNECT indication: a não ocorrência do evento no intervalo de tempo esperado provoca a execução de um T.DISCONNECT.request.

Figura 2.2: Fase de estabelecimento da conexão de sessão com e sem sucesso .



BIBLIOTECA
FEE/FEM/FEQ

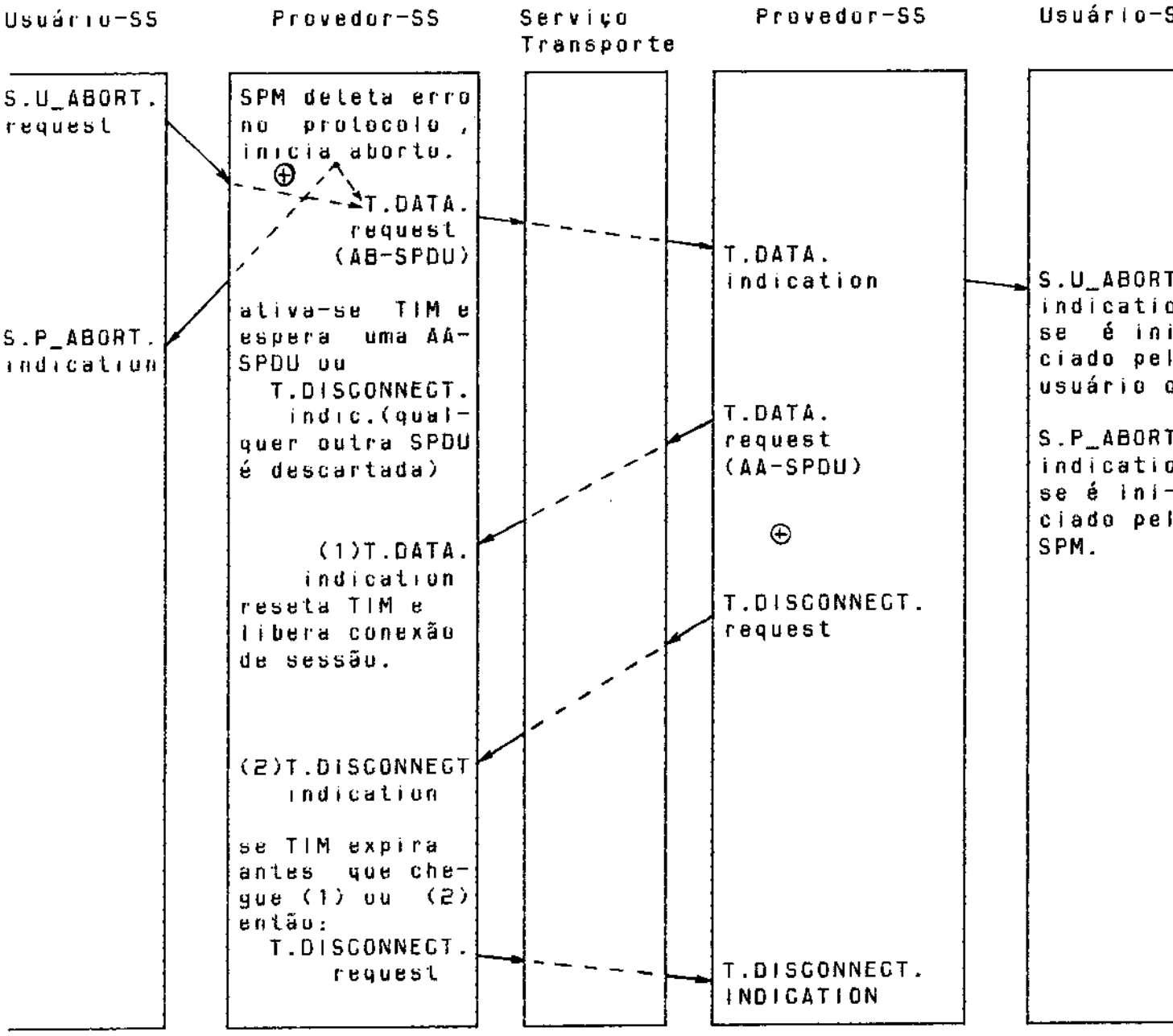



(1).TIM: Disconnection and abort timer

(2).A contagem do timer é cancelada se T.DISCONNECT.indication chegar antes do seu final. Caso contrário, executa-se T.DISCONNECT.request.

NOTA: A SPDU FINISH(FN) indica que a conexão de transporte não será reutilizada.

Figura 2.3: Liberação ordenada (aceita) sem reutilização da conexão de transporte.




(1).Acontece se usuário-SS iniciante solicitou manter a conexão d transporte e é aceita pelo provedor-SS respondedor.

(2).Acontece se usuário-SS iniciante solicita liberar a conexão d transporte ou se provedor-SS respondedor rejeita manter a conexão d transporte.

Figura 2.4: Aborto da conexão de sessão iniciada pela SPM se ocorre um erro no protocolo ou iniciado pelo usuário-SS sem transporte urgente. A SPDU-AB gerada tem informação sobre quem solicitou aborto.

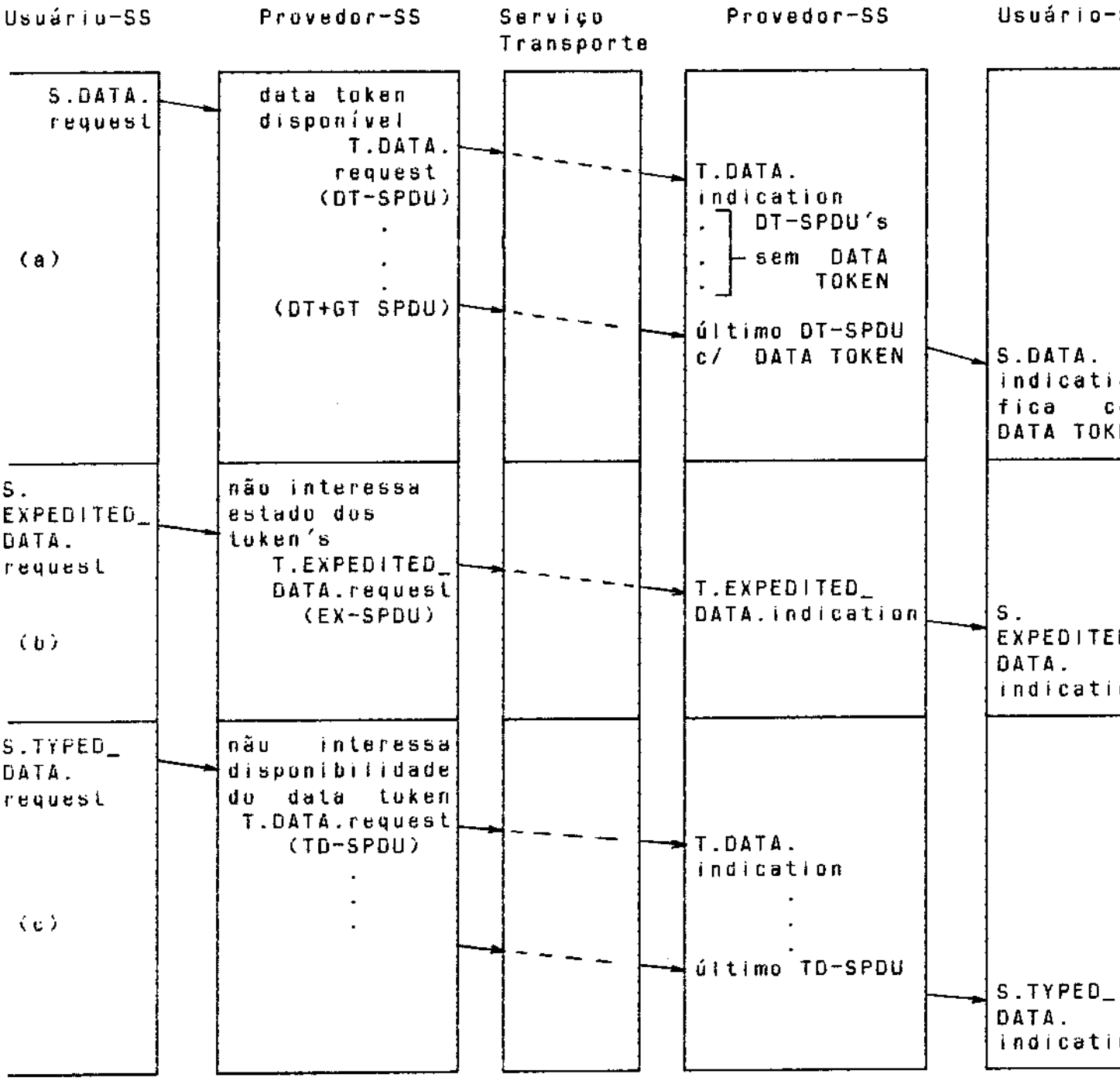



Figura 2.5: Diferentes alternativas de transmissão de dados:

(a)Fluxo de dados normais.

(b)Transmissão de dados urgentes. ("expedited").

(c)Transmissão de dados "typed".

Na Tabela 2.6 apresenta-se um quadro resumido sobre algumas caracteristicas importantes dos tipos de dados, que podem ser transmitidos numa conexão de sessão:

| Tipos de Dados SSDUs | Pode ser mapeada em mais de uma SPDU ? | Quantidade de dados a transmitir | Rota de transporte usada | Utiliza token ? |
|---|---|---|---|---|
| Normais | Sim | Ilimitada | Fluxo normal | Sim, data token |
| Typed | Sim | Ilimitada | Fluxo normal | Não |
| Fora de Atividade (capability) | Não | Limitada (512 octetos máximo) | Fluxo normal | Sim,major/ activity token |
| Urgentes (expedited) | Não | Limitada (14 octetos máximo) | Fluxo urgente | Não |

Tabela 2.6: Comparação entre os diversos tipos de dados numa conexão de sessão.

## 2.7. Escopo da implementação.

De acordo com um dos objetivos desta tese, o escopo de implementação da camada de sessão abrange as unidades funcionais Kernel e Full-duplex, incluindo os seguintes serviços:

No Kernel: -Serviço de conexão.
-Serviço de transferência de dados.
-Serviço de liberação não negociada da conexão (ou seja, se um dos usuários comunicantes decide liberar a conexão, o outro deverá aceitar obrigatóriamente).

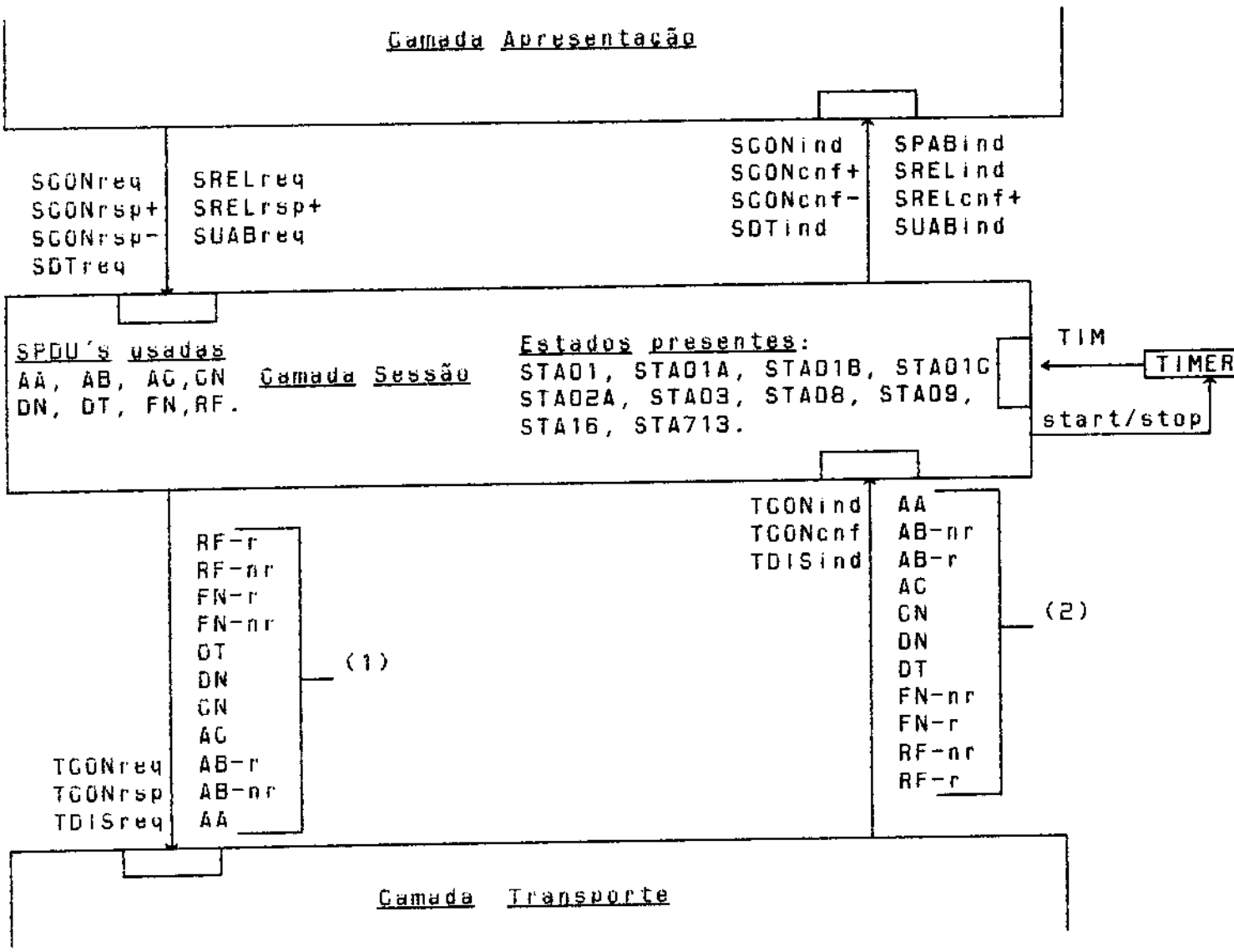
No Full-duplex: -o "data token" não estará disponível e enquanto estiver estabelecida a conexão de sessão, ambos os usuários comunicantes terão o direito de enviar dados em qualquer momento.

Para dar uma idéia geral da presente implementação, apresenta-se na tabela 2.7 uma comparação entre o protocolo completo e o protocolo a implementar, baseada nos parâmetros mais significativos da camada.

| | Protocolo completo | Protocolo a implementar |
|---|---|---|
| Número de estados | 29 | 10 |
| Número de eventos que chegam | 77 | 22 |
| Número de eventos de saída | 76 | 22 |
| Número de transições | 645 | 100 |
| Número de predicados | 75 | 14 |
| Número de ações específicas | 32 | 12 |
| Número de SPDU's | 34 | 8 |

Tabela 2.7

No gráfico da figura 2.6 apresentam-se, utilizando a estrutura do modelo SISDI_MAP, todos os tipos de eventos que poderão chegar às portas de entrada das camadas de Apresentação, Sessão e Transporte, todas as SPDU's e todos os estados da máquina de protocolo que esta implementação vai considerar.




(1) Estes eventos de saída representam as diversas SPDU's que são transportadas pela primitiva T_DATA.request (que na presente implementação será representada pelo evento TDTreq. que não é um evento padrão).

(2) Estes eventos de entrada representam as diversas SPDU's que são transportadas pela primitiva T_DATA.indication (que na presente implementação será representada pelo evento TDTind, que não é um evento padrão).

Figura 2.6: Modelo que relaciona a camada de sessão com as outras, mostrando tambem sua relação com SPDUs e eventos envolvidos.

# CAPÍTULO 3: DESCRIÇÃO DO PROJETO DESENVOLVIDO.

Neste capítulo descreve-se como foi realizado o projeto de especificação da implementação do protocolo de sessão, unidades funcionais Kernel e Full-duplex, enfatizando aqueles aspectos considerados mais importantes.

Primeiramente, explica-se a especificação dos requisitos de projeto utilizando o EPOS-R e depois, a especificação de implementação utilizando o EPOS-S. Em ambos os casos apresenta-se, em primeiro lugar, um roteiro que descreve os elementos formais do EPOS utilizados no desenvolvimento do projeto, para a etapa em consideração, com o objetivo de poder entender melhor a especificação feita. Posteriormente, explica-se como foi realizada a especificação da etapa em consideração, fazendo constantemente referência ao Anexo D (Especificação dos requisitos) ou ao Anexo E (Especificação de implementação). Tais anexos representam, em seu conjunto, o trabalho realizado para alcançar os objetivos propostos.

Nesta especificação entendeu-se por requisitos as características que qualquer implementação do protocolo de sessão deverá cumprir, por exemplo, as tabelas de estado do protocolo de sessão; e por restrições, as características que esta implementação deverá obedecer, sendo estas dependentes do ambiente de operação e da solução proposta.

## 3.1. Especificação dos requisitos.

### 3.1.1. Roteiro.

Em primeiro lugar, o EPOS-R permite utilizar a linguagem informal para descrever objetivos, propostas do problema, soluções, etc.

Em segundo lugar, o EPOS-R fornece elementos (palavras-chave) que têm como função chamar a atenção do leitor para os pontos importantes na proposta de uma idéia e também incorporar tais idéias na especificação formal da solução em EPOS-S. Tais elementos ou palavras-chave são os REQUIREMENT e CONSTRAINT, os quais têm um número definido pelo projetista para sua identificação numa lista de requisitos e restrições definidas durante a especificação formal do problema.

Baseado nessa identificação, o projetista poderá indicar na especificação da solução, em EPOS-S, que um requisito ou restrição está sendo considerado na especificação de um objeto de projeto(ACTION, DATA, CONDITION, DEVICE, EVENT, INTERFACE). Esta consideração é expressa, formalmente, no <fullfils-part> do objeto de projeto especificado. Posteriormente, fazendo uso das ferramentas de análise, é possível determinar se todos os requisitos e restrições estabelecidos foram considerados.

Por último, o EPOS-R permite formalmente descrever um processo caracterizado por ter variáveis de entrada e saída, além das regras que definem seu comportamento. Por exemplo, as tabelas de estado da máquina de protocolo da Camada de Sessão (Kernel e Full-duplex) foram tratadas como processos de decisão e definidas como requisitos.

Para entender melhor a sintaxe formal utilizada para definir um processo de decisão apresenta-se, como exemplo, a especificação de uma tabela de estado fictícia como processo de decisão (Tabela 3.1).

| Estado<br><br>Evento entrada | STA01 | STA01C |
|---|---|---|
| RF-r | // | TDISreq<br><br>STA01 |
| SCONreq | TCONreq<br>[2]<br>STA01B | p01:<br>CN<br>STA02A |

Tabela 3.1

A tabela de estado (Tabela 3.1) pode ser representada pelo processo da figura 3.1.

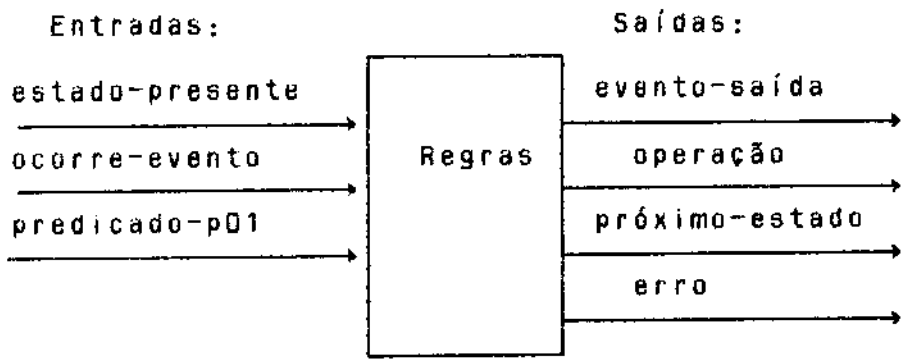



Figura 3.1.: Esquema típico para um processo qualquer.

Para um processo de decisão, as entradas definem o CONDITIONLIST, as saídas o OPERATIONLIST, e as regras as RULES.

No exemplo proposto, os valores, para as variáveis de entrada e saída, são os seguintes:

estado-presente: STA01, STA01C.

ocorre-evento: RF-r, SCONreq.

predicado-p01: v, f, (v = Verdadeiro, f = Falso).

evento-saída: TDISreq, TCONreq, CN.

operação: [2].

próximo-estado: STA01, STA01B, STA02A.

erro: ação a tomar.

A tabela de estado vai ser representada finalmente por três processos de decisão, nomeados por:

MAQUINA-PRINCIPAL.

OUT-EVENT.

NEXT-STATE.

Para ligar OUT-EVENT e NEXT-STATE à MAQUINA-PRINCIPAL utilizar o termo formal LOOKAT.

Portanto, a descrição formal como processo de decisão da tabela de estado proposta, fica da seguinte forma:

```
REQUIREMENT 100 DECISION-PROCESS MAQUINA-PRINCIPAL
  CONDITIONLIST:
          estado-presente( STA01, STA01C ),
          ocorre-evento( RF-r, SCONreq ),
          predicado-p01( v, f ).
  OPERATIONLIST: LOOKAT OUT-EVENT,
                 LOOKAT NEXT-STATE,
                 operação,
                 erro.
  RULES:
  (STA01, SCONreq, - : X, X, X, - ),
  (STA01C, RF-r, - : X, X , -, - ),
  (STA01C, SCONreq, v : X, X, - , - ),
  ( , , : - , - , - , - ).  (condição ELSE)

 DECISION-PROCESS OUT-EVENT
  CONDITIONLIST:
          estado-presente( STA01, STA01C ),
          ocorre-evento( RF-r, SCONreq ),
          predicado-p01( v, f ).
  OPERATIONLIST:
          TDISreq, TCONreq, CN.
  RULES:
  (STA01, SCONreq, - : -, X, - ),
  (STA01C, RF-r, - : X, - , - ),
  (STA01C, SCONreq, v : -, -, X ).

  DECISION-PROCESS NEXT-STATE
  CONDITIONLIST:
                estado-presente( STA01, STA01C ),
                ocorre-evento( RF-r, SCONreq ),
                predicado-p01( v, f ).
  OPERATIONLIST:
                STA01, STA01B, STA02A.
  RULES:
  (STA01, SCONreq, - : -, X, - ),
  (STA01C, RF-r, - : X, - , - ),
  (STA01C, SCONreq, v : -, -, X ).
```

3.1.2. Descrição da especificação dos requisitos feita com EPOS-R.

O relatório completo, com a especificação dos requisitos, encontra-se no Anexo D desta tese. Tal relatório foi gerado por EPOS-D e será constantemente referenciado neste capítulo.

Na especificação dos requisitos e restrições podem distinguir-se duas partes:

Na primeira, item 1.1, "Objetivos e requisitos que deve cumprir a implementação", estabelecem-se, na linguagem natural, os objetivos que deve atingir a implementação através dos requisitos: REQUIREMENT's 210, 220, 230 e 240, e o comportamento da máquina de estado através dos requisitos:
REQUIREMENT's 1000, 1010, 1020, 1030 e 1040, que representam a tabela de estado de estabelecimento da conexão de sessão,
REQUIREMENT's 2000, 2010, 2020, que representam a tabela de estado da transferência de dados,
REQUIREMENT's 3000, 3100, 3110, 3120, 3130, 3200. 3210, 3220 e 3230, que representam a tabela de estado para a liberação da conexão de sessão, e
REQUIREMENT's 4000, 4010, 4020, 4030 e 4040, que representam a tabela de estado para o aborto da conexão de sessão.

As tabelas de estado foram representadas como processos de decisão e a numeração dos requisitos foi escolhida de forma arbitrária, mas, dando uma idéia de agrupamento de requisitos e/ou restrições afins: por exemplo, os objetivos propostos abrangem os requisitos 2xx. A numeração permite intercalar posteriormente outros requisitos e restrições.

As tabelas de estado mencionadas anteriormente, especificadas como processos de decisão, foram submetidas à análise de consistência, completeza e redundância pelo EPOS-A. O resultado da análise foi correto para todas as tabelas; somente no processo de decisão que representa a liberação da conexão detectou-se uma inconsistência quando o estado presente é STAD9 e chega uma primitiva S-RELEASE.response(positive), mas as combinações lógicas que provocam a inconsistência nunca ocorrem, como se pode verificar na tabela 3.2. Por exemplo, Vdnr será colocado em "Verdadeiro" só pelo provedor-SS chamante e isso significa que Vtca é "Verdadeiro", portanto, as combinações lógicas (a) e (c) nunca acontecem. Para a combinação lógica (b), toda vez que Vcoll é colocado em

"Verdadeiro", (executar ação específica [18]), também Vtrr é colocado em "Falso", (executar ação específica [8]), logo esta combinação lógica não ocorre.

| | Vtca | Vtrr | Vcoll | Vdnr | p66 | ^p66&p75 | p69&p01 |
|---|---|---|---|---|---|---|---|
| (a) | F | F | V | V | F | V | V |
| (b) | F | V | V | F | V | F | V |
| (c) | F | V | V | V | V | F | V |

onde:

p66 = Vtrr;

^p66&p75 = ^Vtrr&[(Vcoll&Vdnr)or^Vtca];

p69&p01 = Vcoll&^Vtca

Tabela 3.2: Combinações lógicas que provocam inconsistência, mas que nunca ocorrem na execução da SPM.

Na segunda parte do Anexo D, ítem 2. "Solução proposta", estabelece-se o modelo da solução (item 2.1) que abrange as restrições CONSTRAINT's 300, 310, 320, 330, 340, 350, 360 e 370, que correspondem aos gráficos das figuras 1 e 2 e à tabela 1 do anexo D.

O gráfico da figura 1 Anexo D estabelece o modelo SISDI_MAP para a camada de sessão, ficando formalmente especificado através da restrição 300.

O gráfico da figura 2 Anexo D estabelece qual é a forma em que é abordada a implementação a nível de diagrama de blocos. A explicação de cada um dos blocos é feita chamando a atenção nos pontos importantes através das restrições CONSTRAINT's 310, 320, 330, 340, 350, 360 e 370.

É importante enfatizar que se procurou uma solução em que seja possível, sem maior esforço, mudar a implementação para outro ambiente de operação isolando, funcionalmente, a máquina de protocolo de sessão das interfaces de Entrada/Saída.

Outro fato importante de enfatizar, e que é particular desta implementação, é a forma em que se traduz a informação de cabeçalho da mensagem recebida, explicada na CONSTRAINT 310. No caso em que se precise implementar outras unidades funcionais, aumentarão na tabela 1 (Anexo D) os espaços livres, o que implica no maior consumo de memória RAM. Neste caso, é recomendável a

implementação do bloco TRATA-MENSAGEM, por exemplo, através de um prugrama baseado em "switch-case" que pegue os valores de cabeçalho da mensagem recebida, para as variáveis de seleção nome-serviço e nome-primitiva, descendo-se sucessivamente até ao nome do evento de entrada ISO.

No item 2.2 "Estrutura de dados do SISDI_MAP" (Anexo D) estabelece-se a restrição 410, que estipula a estrutura de dados do SISDI_MAP já definida, e que se encontra no anexo F.

No item 2.3 "Predicados e variáveis ISO que esta implementação abrange" (Anexo D) apresenta como são consideradas as variáveis que a ISO define para a camada de sessão, e que vão ser utilizadas na presente implementação. A tabela 2 (Anexo D) apresenta as variáveis ISO, como elas vão ser especificadas no EPOS-S e o valor que vão ter (fixo ou variável) durante a execução do protocolo, incluindo a determinação dos valores fixos (ver Notas da tabela 2 do Anexo D). As restrições 510 e 520 (Anexo D) estabelecem que as variáveis ISO serão especificadas como objetos de projeto DATA e que os predicados serão especificados como objetos de projeto CONDITION.

Já no item 2.4 "Variáveis próprias desta implementação" (Anexo D), definem-se três variáveis internas que são particulares desta implementação, as quais são especificadas como restrições CONSTRAINT's 610, 620 e 630.

Finalmente, no item 2.5 "Definição das questões locais ("Local choice") na SPM" (Anexo D) aborda-se a tomada de decisões sobre o comportamento da máquina de protocolo, deixado em aberto pela ISO na ocorrência de certas situações. Essencialmente duas são as situações consideradas nesta implementação: possibilidade de reutilizar a conexão de transporte, no caso de desligar-se a conexão de sessão, o que significa deixar ligada a conexão de transporte e, definição de ações nas interseções não válidas da tabela de estado. O primeiro caso define-se na restrição 710 e o segundo caso apresenta duas alternativas através das restrições 720 e 730.

## 3.2. Especificação da implementação.

### 3.2.1. Roteiro.

A especificação da implementação das unidades funcionais Kernel e Full-duplex da camada de Sessão, implicou gerar 366 objetos de projeto, divididos em: 190 ACTION's, 136 DATA's e 40 CONDITION's, relacionados através de uma estrutura hierárquica de 12 níveis.

O presente roteiro tem como propósito explicar o formato de cada um dos objetos utilizados na especificação desenvolvida, para permitir desta forma, um melhor entendimento da documentação apresentada neste capítulo.

Dos seis tipos de objetos de projeto que o EPOS-S define, (ACTION, DATA, CONDITION, EVENT, INTERFACE e DEVICE), somente três foram utilizados na especificação completa da implementação : ACTION, DATA e CONDITION. As razões pela sua escolha foram as seguintes:

(1) O SISDI_MAP e o Núcleo de Tempo Real, já existentes, não foram especificados formalmente. Portanto, foram consideradas as interfaces de E/S utilizando-se as primitivas espera_mens(...) e envia_mens(...) do Núcleo de Tempo Real, o que tornou desnecessário a sua especificação com o objeto INTERFACE.

(2) Um dos objetivos da especificação visou gerar automaticamente a maior porcentagem possível de código. O Composer C, o qual faz a geração automatica para código C, não considera ou ignora os objetos de projeto EVENT, INTERFACE e DEVICE na sua atual versão.

A seguir, descrevem-se os elementos que compõem os objetos de projeto utilizados no desenvolvimento desta especificação.

Uma parte comum para todos os tipos de objetos de projeto é a descritiva, a qual é delimitada pelas palavras chave DESCRIPTION e DESCRIPTIONEND, permitindo descrever, opcionalmente, as características PURPOSE, NOTE, TEST, PERFORMANCE, REFERS-TO, DATE, FULFILS, CATEGORY e IDENTIFICATION. Desta só as seguintes foram usadas:

PURPOSE: Para explicar a função que o objeto cumpre dentro do escopo da especificação.

NOTE: Para explicar algum fato particular relacionado com algum detalhe importante na especificação.

FULFILS: Para indicar que o presente objeto está satisfazendo um ou vários requisitos/restrições que foram especificados no EPOS-R.

CATEGORY: Para definir palavras chave que permitam posteriormente selecionar objetos na geração da documentação.

Em relação ao formato dos objetos de projeto utilizados na especificação, deve-se dizer que não foi necessario usar todos os elementos de cada tipo, só sendo explicados aqueles que realmente foram utilizados nesta especificação.

O objeto de projeto ACTION.

Este objeto apresenta as seguintes alternativas:

ACTION MODULE: Permite encapsular o projeto em blocos e relacioná-los entre si. Distingue-se do resto das alternativas na sua semântica, não descrevendo uma seqüência de execução de ações, senão, a estrutura geral do projeto e sua relação com outros módulos, e a relação dos dados utilizados tanto pelo módulo como por outros módulos.

ACTION TASK: Permite especificar uma tarefa.

ACTION PROCEDURE: Permite especificar um procedimento que vai ser invocado em várias partes dentro da especificação.

ACTION MACRO: Permite especificar uma seqüência de ações que ocorre frequentemente, e a ser inserida no local invocado na especificação (não é uma chamada de um procedimento ou rotina).

ACTION: Especifica uma ação qualquer.

Somente a ACTION MODULE não é uma unidade executável. O formato utilizado para ela é o seguinte:

**ACTION MODULE** <identificador do módulo>.
**DESCRIPTION**:

.

.

**DESCRIPTIONEND**
**DECOMPOSITION**:
(/ lista de identificadores de módulos e/ou de ações, separados por vírgula /).
**ACTIONEND**

(Por exemplo, ver as ACTION MODULE's definidas no extrato da especificação).

O formato geral utilizado pelos outros tipos de ACTION's é o

seguinte:

**ACTION** x <identificador da ação>.

**DESCRIPTION**:

.

.

**DESCRIPTIONEND**

**DECOMPOSITION**: (Explicação por separado).

**CODE**: (Aqui se coloca o código, entre aspas, na linguagem escolhida).

**TRIGGERED**: **SYSTEMSTART**. (utilizado na ACTION TASK, que indica ao Composer que esta ação representa o programa principal)

**INPUT**:(lista de dados, separados por vírgula, utilizados pela ação).

**OUTPUT**:(lista de dados, separados por vírgula, que a ação gera).

**ALT**:(Utilizado só na ACTION PROCEDURE para definir os parâmetros de entrada/saída do procedimento.

**ACTIONEND**

"x" pode ser TASK ou PROCEDURE ou nenhum dos dois.

DECOMPOSITION permite especificar o fluxo do controle da execução das ações que a ACTION em consideração abrange. Para isso, faz uso das seguintes sentenças de controle: if-then-else, switch-case, while, repeat, wait-event, wait-condition, wait-rendezvous, parallel-action, stop, delay, set e reset.

Na especificação usam-se if-then-else, switch-case e repeat. Como exemplos, citam-se ACTION MACRO DETERMINA-USAR-PROTOCOLO, ACTION PROCEDURE PROTOCOLO-DE-SESSAO e ACTION TASK CONTROLADOR-DO-PROVEDOR-SS, respectivamente.

No caso de ALT, pode-se usar como exemplo a ACTION PROCEDURE PROTOCOLO-DE-SESSAO, onde se definem os seguintes parâmetros de entrada para este procedimento: EVENTO_ENTRADA e ESTADO_ATUAL. Estes parâmetros pegam os valores das variáveis INCOMING_EVENT e VAR_INTERNA!ESTADO_PRESENTE(INDEX_CONTEXT) respectivamente quando o procedimento for invocado, como acontece na ACTION MACRO DETERMINA-USAR-PROTOCOLO. As partes INPUT e OUTPUT na ACTION PROCEDURE, definem se o parâmetro é de entrada ou de saída. No exemplo, os parâmetros estão declarados no INPUT, portanto, ambos são parâmetros de entrada.



UNICAMP
BIBLIOTECA CENTRAL

A parte ALT é opcional para uma ACTION PROCEDURE. Sua omissão torna mais fácil o manuseio das invocações, em relação à declaração dos parâmetros, quando o procedimento é frequentemente invocado na especificação; como exemplo, cita-se a ACTION PROCEDURE TRATAR-MENS que é invocada na ACTION PROCESSA-PORTA-APRESENTACAO e em outras ações. Neste caso não se utilizou ALT, mas na invocação do procedimento declararam-se os parâmetros (à esquerda do símbolo `=') e as variáveis (à direita do símbolo `='), que entregam ou recebem o valor. Na invocação deve-se usar a palavra chave USING que é acrescentada ao nome do procedimento invocado, como se mostra no exemplo mencionado.

O objeto de projeto DATA.

Nesta especificação podem-se encontrar os tipos DATA TYPE, DATA FORMAL, DATA ARRAY e DATA, os quais apresentam o seguinte formato:

**DATA** x <identificador do dado>. (x pode ser TYPE, FORMAL, ARRAY ou nada)

**DESCRIPTION:**

.

.

**DESCRIPTIONEND**

**DECOMPOSITION:**

(Válido para todos os tipos de DATA. é utilizada para decompor o objeto em outros dados de acordo com uma estrutura de acordo com a aplicação. Como exemplo, ver DATA TYPE ESTRUT_VAR_INTERNA que faz uma decomposição do tipo 'structure'; neste caso, o objeto especifica um tipo de variável nomeada ESTRUT_VAR_INTERNA, cujos membros são indicados na decomposição. Esta estrutura é utilizada posteriormente pelo objeto DATA VAR_INTERNA).

**CODE:** Pode-se inserir o código, entre aspas, para definir uma variável na linguagem escolhida.

**BOUNDS:** Só tem validade nos DATA ARRAY's; ver DATA ARRAY VTCA, no qual define-se um arranjo unidimensional cujo comprimento máximo é N.

**SCOPE:** Só tem significado no DATA FORMAL. Aqui deve-se colocar o nome da ACTION PROCEDURE à qual pertence o objeto DATA declarado.

**TYPE**: Define o tipo do dado: nesta especificação usaram-se somente FIXED, para inteiros e BOOLEAN para booleanos. Tambem pode-se indicar a outro objeto dado (DATA TYPE) que define o tipo, como por exemplo em VAR_INTERNA.

**IDENTICAL**: Define um valor para o objeto dado, ou seja, define o objeto dado como uma constante (como exemplo ver DATA STA01).

**RANGE**: Define um intervalo ou uma lista de valores para o objeto dado especificado. Pode ser utilizado para definir dados do tipo enumerado.

**DATAEND**

O objeto de projeto CONDITION.

O formato para este objeto de projeto é o seguinte:

**CONDITION** <identificador da condição>.

**DESCRIPTION**:

.

.

**DESCRIPTIONEND**

**DECOMPOSITION**: (Ver exemplos mencionados mais para frente).

**CODE**: (Ver exemplos mencionados mais para frente).

**CONDITIONEND**

Este objeto é utilizado nas declarações if-then-else: como, por exemplo:

(a) A ACTION PROCEDURE TRATAR-MENS, a qual utiliza a condição COND_ENTRADAS, que por sua vez é decomposta na expressão de comparação TIPO_ERROR = 100 e onde TIPO_ERROR é um objeto DATA.

(b) A ACTION TASK CONTROLADOR-DO-PROVEDOR-SS, que utiliza a condição IN-PORT-PRES mas que não é decomposta, mas sim sustituída pela parte CODE que representa o resultado da invocação de uma primitiva do Núcleo de Tempo Real.

(c) A ACTION ACAO-23 que faz uso da condição COND_AC23, que por sua vez é decomposta em uma relação lógica dos objetos de projeto CONDITION P01, P16 e P65.

Observações gerais:

(1) O EPOS não distingue letras maiúsculas de minúsculas.

(2) Os textos entre aspas não são modificados.

(3) Se as partes DECOMPOSITION e CODE são declaradas

simultaneamente, só a parte CODE é considerada quando o código for gerado.

3.2.2. Descrição da especificação da implementação.

O relatório completo da especificação, gerada por EPOS-D, encontra-se no Anexo E da tese; nele, os objetos de projeto estão ordenados por nível hierárquico, apresentando-se primeiro um índice de todos os objetos de projeto, depois o corpo principal com a especificação e por último um índice de referências cruzadas.

Na especificação definiram-se três palavras-chave no <category-part>:

(a) "extrato-resumo", indica que o objeto de projeto forma parte de um extrato da especificação com o objetivo de dar uma idéia geral da implementação sem entrar em detalhes.

(b) "exemplo-exec", indica que o objeto de projeto é utilizado para descrever o exemplo de execução apresentado no capítulo 4 desta tese.

(c) "terminal", indica que o objeto de projeto é uma ação terminal, ou seja, a ação não é decomposta e só possue o <code-part>, neste caso em código C.

Estas palavras-chave podem ser utilizadas para gerar extratos da documentação, por exemplo geração de um relatório que dê uma idéia geral da implementação (utilizando a palavra-chave "extrato-resumo").

Na descrição far-se-á referência aos objetos do relatório do Anexo E, a menos que se indique o contrário. Para uma mais clara explicação se usarão gráficos do tipo hierárquico, fluxograma e "block". Este último combina o fluxograma com a entrada e saída de dados. Todos os gráficos foram gerados em "plotter" por EPOS-D automaticamente a partir da base de dados da especificação.

3.2.2.1.Descrição.

A camada de sessão, representada pelo módulo PROVEDOR-SS, foi estruturada em duas grandes entidades especificadas pelos módulos TEMPORIZADOR e CONTROLADOR, (como se observa no diagrama hierárquico da figura 3.2), desenvolvidas independentemente e implementadas como tarefas, (por exemplo a ACTION TASK CONTROLADOR-DO-PROVEDOR-SS pertencente à entidade CONTROLADOR, e da ACTION TASK

GERENCIADOR-TIMERS pertencente à entidade TEMPORIZADOR). A implementação final da tarefa GERENCIADOR-TIMERS ficou pendente porque atualmente o núcleo de tempo-real não possue temporizadores e a solução de criar tarefas temporizadoras, proposta como alternativa na especificação, pode provocar um excesso de tarefas no ambiente limitado de simulação em PC. A tarefa CONTROLADOR-DO-PROVEDOR-SS é o objeto finalmente implementado, e a forma de trabalho quando escalonada, pode ser observada no fluxograma da figura 3.3.

Quando a tarefa CONTROLADOR-DO-PROVEDOR-SS é escalonada pela primeira vez, inicializa as variáveis do protocolo, como se especifica na ação INICIALIZA-PROTOCOLO, para entrar posteriormente num laço infinito.

Dentro do laço infinito, cada uma das portas de entrada da tarefa é consultada, de forma seqüencial, para determinar se chegou uma mensagem através dos objetos de condição IN-PORT-PRES, IN-PORT-TRANSP e IN-PORT-TIMER. Se uma mensagem chegou, ela é consumida e processada de acordo com a porta correspondente, como se especifica através das ações PROCESSA-PORTA-APRESENTACAO, PROCESSA-PORTA-TRANSPORTE e PROCESSA-PORTA-TIMER. Essas ações estão estruturadas da mesma forma, executando as ações TRATAR-MENS e DETERMINA-USAR-PROTOCOLO seqüencialmente; mas definindo-se o tipo de porta, quando é preenchido o parâmetro CHAVE, no momento de invocar a ação TRATAR-MENS.

Se não acontecem erros durante a execução de TRATAR-MENS o procedimento PROTOCOLO-DE-SESSAO é executado, como se especifica na ação DETERMINA-USAR-PROTOCOLO.

As ações procedimento TRATAR-MENS, PROTOCOLO-DE-SESSAO e ERROR, são objetos fundamentais desta especificação porque a partir delas constrói-se o restante do sistema. Estas três ações ficam relacionadas no corpo principal da tarefa quando o código na linguagem C é gerado.

A seguir explicam-se estes procedimentos:

(A). A ação procedimento TRATAR-MENS encarrega-se de pegar as mensagens e traduzi-las em eventos padrões ISO, ou seja, conforme a interface de entrada entre o protocolo e o ambiente de operação (cumpre as restrições 310, 320 e 330 estabelecidas em EPOS-R anexo D).

No diagrama "block" da figura 3.4 apresenta-se o fluxograma da ação procedimento TRATAR-MENS em conjunto com a entrada e saída de dados. Na ação LER-ENTRADAS pega-se do cabeçalho da mensagem os valores do nome da primitiva e do nome do serviço, ficando armazenados temporariamente nas variáveis internas NOME_PRIMITIVA e NOME_SERVICO respectivamente.

Depois, executa-se a ação ENTREGA-CONTEXT-INDEX, o qual implica em pegar do cabeçalho da mensagem, o valor do identificador da conexão para determinar o índice de contexto ao qual pertence a mensagem recebida, (este valor fica armazenado na variável de saída INDICE_CONTEXTO).

| nome_serviço | nome_primitiva | | | | | |
|---|---|---|---|---|---|---|
| | request | indica-tion | respon-se_pos | respon-se_neg | confirm _pos | confirm _neg |
| s_connection | SCONreq | | SCONrsp _pos | SCONrsp _neg | | |
| s_release | SRELreq | | SRELrsp _pos | | | |
| s_u_abort | SUAreq | | | | | |
| s_p_abort | | | | | | |
| s_data | SDTreq | | | | | |
| t_connection | | TCONind | | | TCONcnf | |
| t_disconnect | | TDISind | | | | |
| t_data | | TDTind | | | | |

Tabela 3.3: Tabela tradutora de mensagem para eventos de entrada.

Finalmente executa-se a ação ENTREGA-INCOMING-EVENT, a qual pega os valores das variáveis NOME_PRIMITIVA e NOME_SERVICO, e a partir da Tabela 3.3 (ver Tabela 1 do Anexo D) determina-se o

evento padrão ISO ao qual pertence a mensagem recebida. Se o evento resultante for um TDTIND(TDATA.indication) a mensagem transporta uma SPDU e precisa de um processamento adicional (ver ação TRATA-EVENTO-TDTIND). O valor do evento correspondente à mensagem recebida fica armazenado na variável de saída IN_COMING_EVENT.

O objeto condição COND_ENTRADAS verifica se houve algum erro no processamento, definindo o valor da variável TIPO_ERROR. Posteriormente também é verificado na ação DETERMINA-USAR-PROTOCOLO se se executa o procedimento PROTOCOLO-DE-SESSAO ou o procedimento ERROR no caso de erros em TRATAR-MENS.

(B). A ação procedimento PROTOCOLO-DE-SESSAO é o objeto que especifica o protocolo da camada de sessão (por enquanto unidades funcionais Kernel e Full-duplex), e que nas ações terminais, no <code-part>, contém as primitivas do núcleo de tempo real necessárias para as saídas dos eventos, ou seja, para colocar as mensagens geradas pelo protocolo, nas portas de entrada das outras camadas ou tarefas. Logo, esta ação deve satisfazer os requisitos relacionados com as tabelas de estado do protocolo e com a interface de saída (ver restrição 350 anexo D) além de outras restrições relacionadas com detalhes particulares desta implementação (ver restrições 370, 610, 620 e 630 no anexo D).

No diagrama "block" da figura 3.5 mostra-se o fluxograma e os dados de entrada do procedimento. Sua utilização é feita na ação DETERMINA-USAR-PROTOCOLO onde os valores para os parâmetros de entrada são definidos.

A especificação do procedimento PROTOCOLO-DE-SESSAO está de acordo com a restrição 370, anexo D, que estabelece a metodologia de implementação da tabela de estado e que através do estado presente, determina em que coluna da tabela de estado deverá posicionar-se. Posteriormente, com o valor do evento de entrada, determina a ação, de acordo com o estabelecido pela tabela de estado para a interseção indicada.

Por exemplo se o estado atual para um certo contexto for STAD1 então, o diagrama "block" da figura 3.6 mostra as ações que deverão tomar-se dependendo do evento indicado no parâmetro de entrada EVENTO_ENTRADA.

A variável global AVISO não pertence ao protocolo mas é utilizada nesta implementação porque, em alguns casos, a SPM

estabelece que devem gerar-se duas mensagens simultaneamente, e então, as ações finais que enviam as mensagens devem saber desta situação para procurar espaço de memória e criar outro bloco de dados para uma das mensagens, (a outra utilizará o bloco de dados já existente). Quando se deva gerar duas mensagens a variável AVISO será verdadeira, do contrário, falsa. (por exemplo, quando o estado atual é STA02A e chega o evento AB-r). As ações ACAO-1 e ACAO-2 especificam as ações a tomar de acordo com a SPM.

Ao abordar a especificação da tabela de estado do protocolo fez-se uma análise sobre as ações que ela estipula, determinando-se as seguintes ações típicas:

(1). Executar ações específicas. Para isso especificou-se o procedimento EXECUTA cujo parâmetro de entrada indica qual ação específica deve ser executada.

(2). Gerar uma primitiva de serviço de transporte, tendo-se especificado o procedimento GERA-PRIMITIVA-T, cujo parâmetro de entrada indica qual primitiva de transporte, (distinta de T-DATA.request), deverá ser gerada.

(3). Gerar uma SPDU: para tal o procedimento GERA-SPDU, cujo parâmetro de entrada indica que tipo de SPDU deverá ser preparada e que posteriormente será enviada à camada de transporte através da primitiva T-DATA.request, a qual é gerada pelo procedimento ENVIA-TDATAREQ.

(4). Gerar uma primitiva do serviço de sessão, através do procedimento GERA-PRIMITIVA-S, cujo parâmetro de entrada indica qual primitiva-SS deverá ser gerada.

(5). Mudar o estado presente, do contexto ativo, para o novo estado indicado pela SPM.

(6). De (1), (2), (3), (4) e (5) deduz-se que é necessário preparar os valores dos parâmetros de entrada de cada procedimento, além de atribuir um novo valor (outro estado) para o estado presente. Para isso se especificaram uma série de ações, do tipo DEFINE-SITUACAO-X, onde X é um número associado com a ACAO-X correspondente.

Considerando o anterior e a título de exemplo, suponha-se que o estado presente seja STA01 e que ocorra o evento SCONreq. Segundo a SPM determinam-se as seguintes ações:

TCONreq
(2)
STA01B

Ou seja, deverá gerar-se a primitiva de transporte T-CONNECT.request, executar a ação específica (2) e finalmente mudar o estado presente para STA01B.

Essas atividades foram especificadas na ação ACAO-1, (o número 1 foi escolhido por ser esta a primeira ação a ser executada na tabela quando a SPM está no estado inicial STA01). As ações ACAO-X's ocorrem de forma correlada, (a menos que uma ação se repita). No <decomposition-part> da ACAO-1 estabelece-se a seguinte seqüência de execução:

DEFINE-SITUACAO-1;
GERA-PRIMITIVA-T USING(PRIM_TRANSP);
EXECUTA USING(ACAO_ESPECIFICA).

A ação DEFINE-SITUACAO-1 define o valor de PRIM_TRANSP como TCONreq, de ACAO_ESPECIFICA como A2 e define o valor de VAR_INTERNA!ESTADO_PRESENTE como STA01B.

Posteriormente, o procedimento GERA-PRIMITIVA-T gerará a primitiva T-CONNECT.request, e a seguir, o procedimento EXECUTA define Vtca como falso, completando desta forma o indicado pela SPM.

O formato geral para cada ação ACAO-X é primeiro da especificação de uma ação do tipo DEFINE-SITUACAO-X e, em segundo lugar da especificação de um ou vários dos procedimentos EXECUTA, GERA-PRIMITIVA-T, GERA-SPDU e/ou GERA-PRIMITIVA-S de acordo com a SPM.

Por último, os predicados estabelecidos no protocolo foram especificados como objetos de projetos CONDITION, os quais decompõem-se em função das variáveis ISO (no <decomposition-part> do objeto CONDITION), de acordo com a relação lógica que deve cumprir o predicado em consideração. O protocolo tomará uma decisão correspondente como acontece, por exemplo, na ação ACAO-12 que utiliza o predicado P02 como objeto condição. Todos os predicados foram especificados como condição dentro das ações do tipo ACAO-X.

(C). A ação procedimento ERROR é o objeto de projeto que especifica as ações a tomar quando chegam eventos que correspondem às interseções não válidas da tabela de estado do protocolo.

Este objeto deve cumprir as restrições 360, 720, 730 e 740 da especificação dos requisitos, onde é estabelecido verbalmente o que deve ser feito ante situações de erro.

No diagrama "block" da figura 3.7, pode-se observar a estrutura geral deste procedimento.

Quando esta ação é invocada, seu parâmetro de entrada vai conter um código que será testado na condição COND_ERROR_NAO_FATAL, (COD_ERROR distinto de 5), executando a ação ERROS-NORMAIS ou a ação ABORTA-SISTEMA. Nesta implementação existem dois tipos de código de erro representados pelas variáveis internas COD_ERROR e ORIGEM_ABORTO. A variável COD_ERROR, utilizada como parâmetro de entrada do procedimento ERROR, representa um conjunto mais variado de códigos de erro, particulares da implementação. Tais códigos são os seguintes:

0: Existe algum erro na indicação das portas de entrada.

1: Erro por falta de disponibilidade de índice de contexto, na tabela de conexão, para o evento que chegou.

2: Foi recebida uma spdu não válida.

3: Recebeu-se um evento desconhecido (não considerado na tabela tradução de eventos TAB_INC_EVENT).

4: Evento fora da seqüência do protocolo.

5: Tentou-se criar um novo bloco de dados, mas não há espaço disponível. Implica abortar o sistema de simulação.

Os erros 0, 1, 2 e 3 são gerados no procedimento TRATAR-MENS, e o erro 4 é gerado no procedimento PROTOCOLO-DE-SESSAO.

A variável ORIGEM_ABORTO pode ter só um de quatro valores possíveis, porque representa o código de erro que a ISO indica para as mensagens nos casos de aborto ou de erro na SPM. O código é o seguinte:

0: Aborto pelo usuário. é gerado nas ações DEFINE-SITUACAO-19A e DEFINE-SITUACAO-19B porque chegou o evento SUABreq. É tratado normalmente (não como erro) pela SPM.

1: Chegou uma spdu não válida ou não considerada neste protocolo. é gerado na ação ERROS-NORMAIS, portanto é considerado em casos de erro no protocolo.

2: Chegou um evento (distinto de uma spdu) fora do protocolo. É gerado na ação ERROS-NORMAIS.

3: Aconteceu uma desconexão inesperada de transporte. É gerado na ação DEFINE-SITUACAO-9 porque chegou o evento TDISind. É tratado normalmente pela SPM.

Portanto, quando a ação ERROS-NORMAIS é executada, (ver o diagrama "block" da figura 3.8, para esta ação), a primeira atividade é reduzir o código de erro interno ao código de erro indicado pela ISO, através das ações SPDU-INVAL e RAZAO-DESCON, os quais são os únicos casos tratados pelo procedimento ERROR. Os outros casos, aborto pelo usuário e desconexão inesperada de transporte, são tratados pela SPM.

Posteriormente, e antes de gerar algum evento, verifica-se a condição EXISTE_CONX_TRANSP (caso positivo) executa-se a ação ACAO-TOTAL, caso contrário a ação ACAO-PARCIAL.

A ação ACAO-TOTAL significa que se enviará uma primitiva S-P-ABORT.indication ao usuário -SS local e uma SPDU AB ao provedor -SS remoto. Além disso ativa-se o timer para esperar uma resposta.

No caso da ação ACAO-PARCIAL, ela só envia uma primitiva S-P-ABORT.indication ao usuário -SS local, porque não existe a conexão de transporte, e apaga o identificador da conexão deixando o índice de contexto disponível.

Finalmente, é executada a ação ACAO-PARTICULAR, a qual significa enviar à tela as mensagens de erro correspondentes ao código interno representado pela variável COD_ERROR, dando-se por executado o procedimento ERROR.

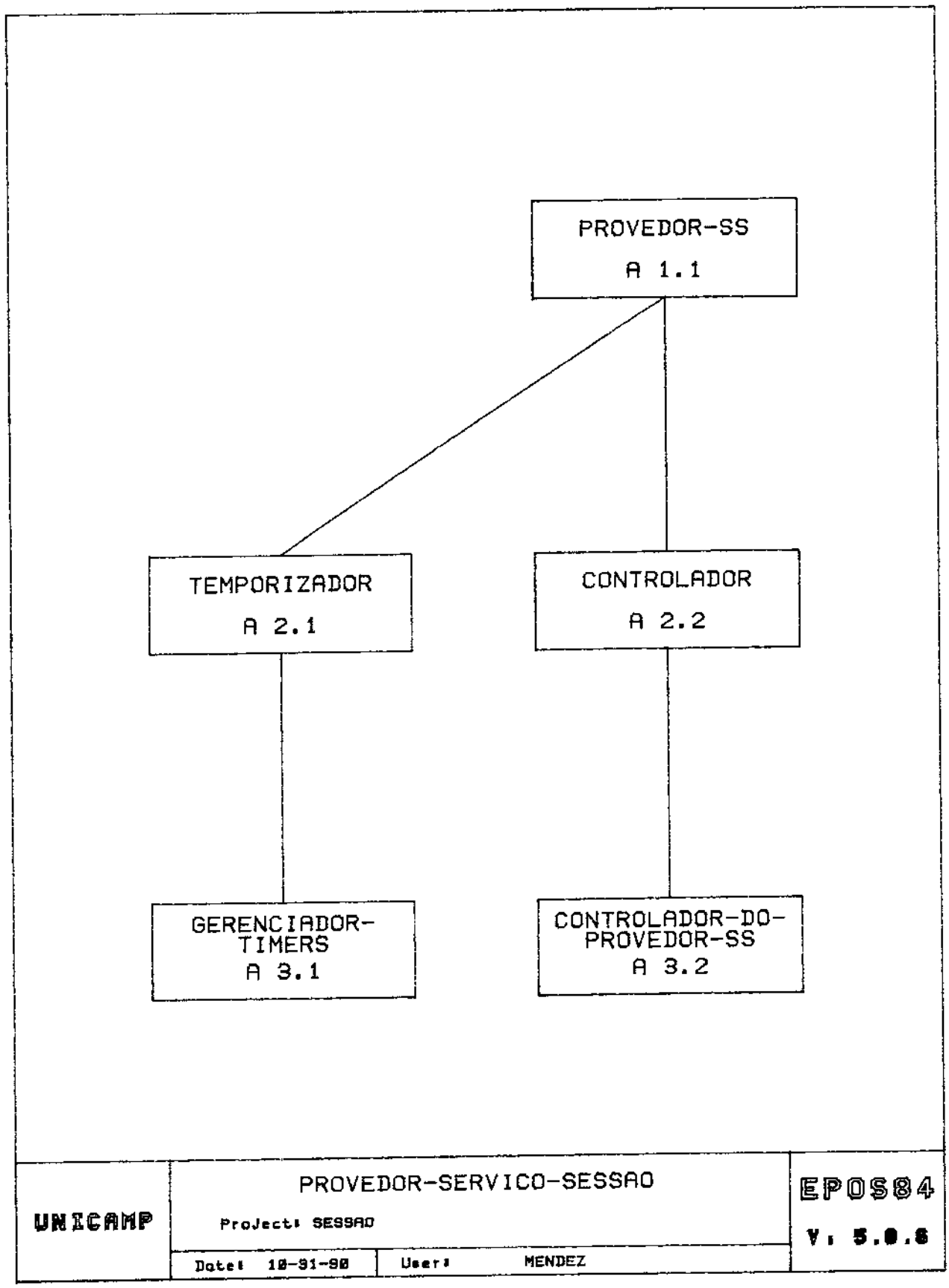



Figura 3.2: Diagrama hierárquico principal da especificação.

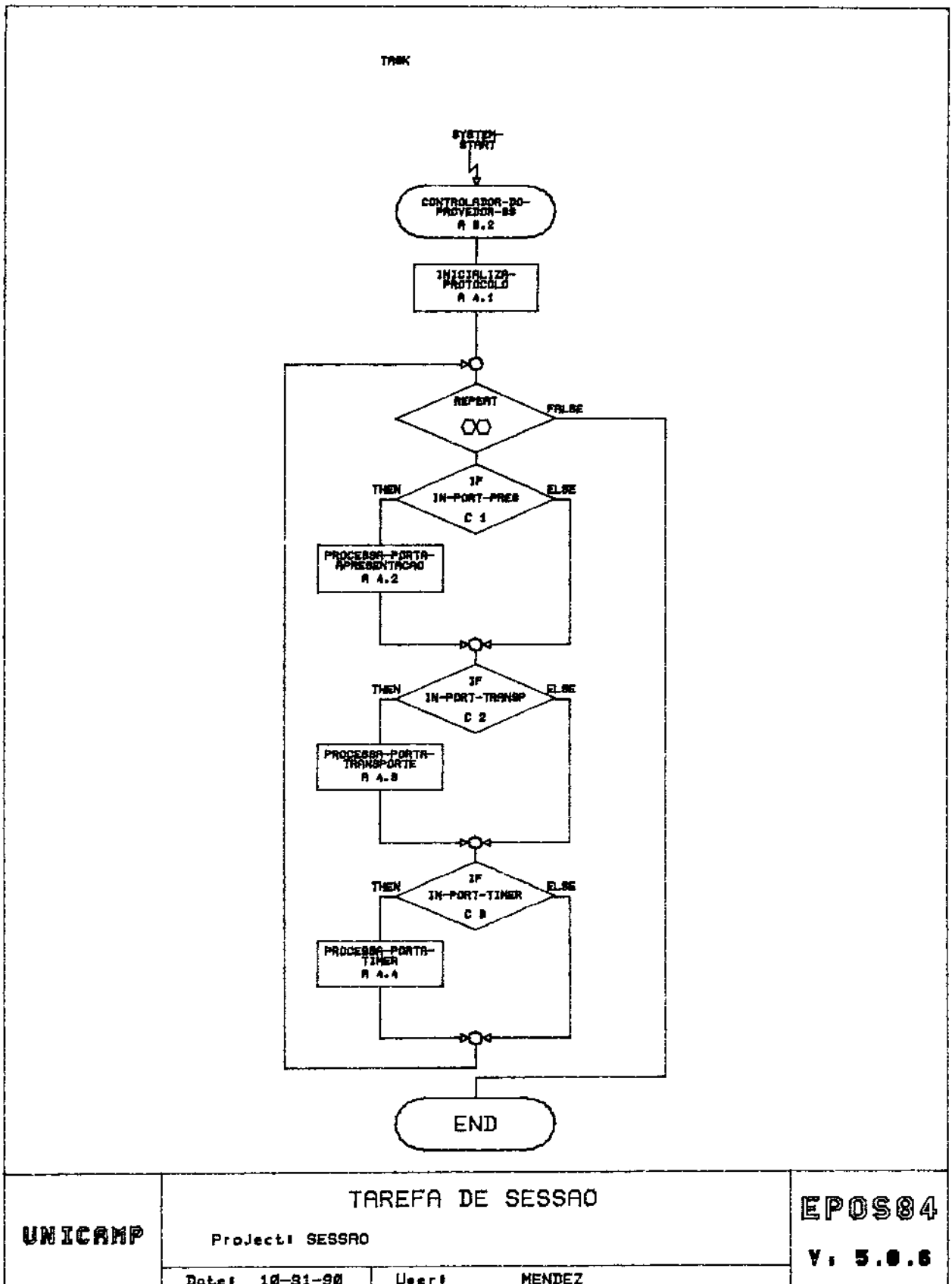



Figura 3.3: Diagrama de fluxo principal para a tarefa que executa a SPM.

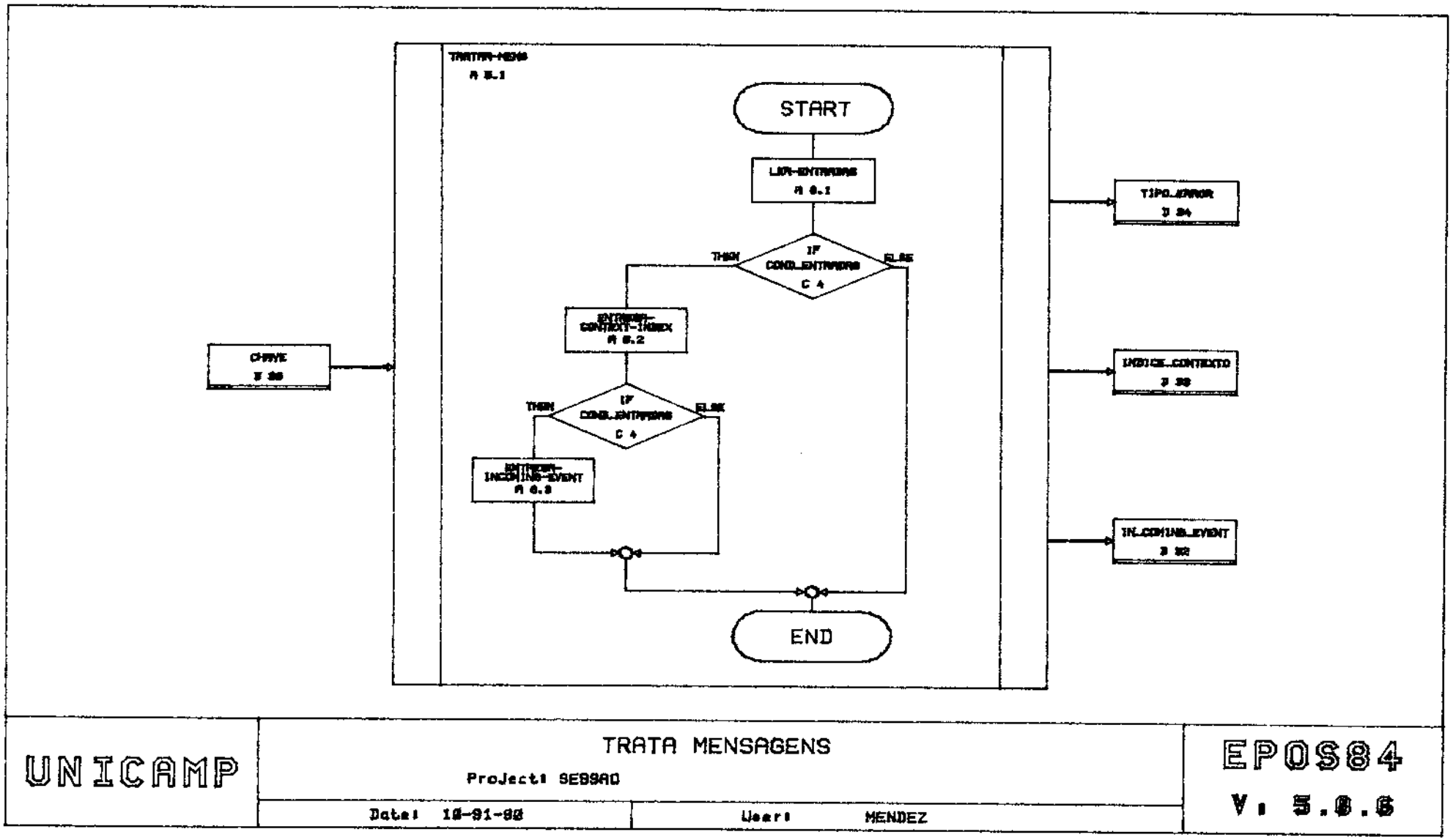



Figura 3.4: Diagrama "block" para o procedimento que trata as mensagens recebidas.

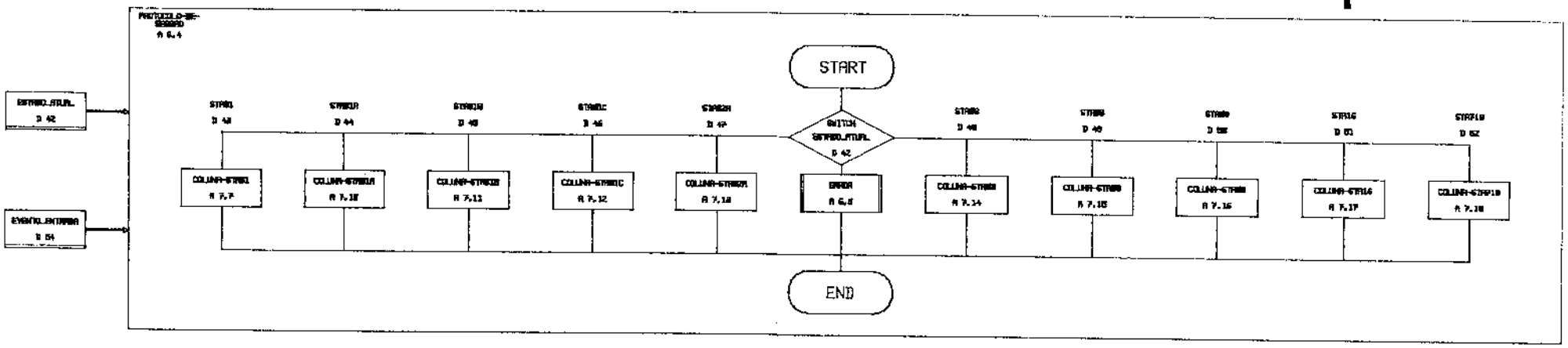



Figura 3.5: Diagrama "block" para o procedimento que implementa o protocolo de sessão.

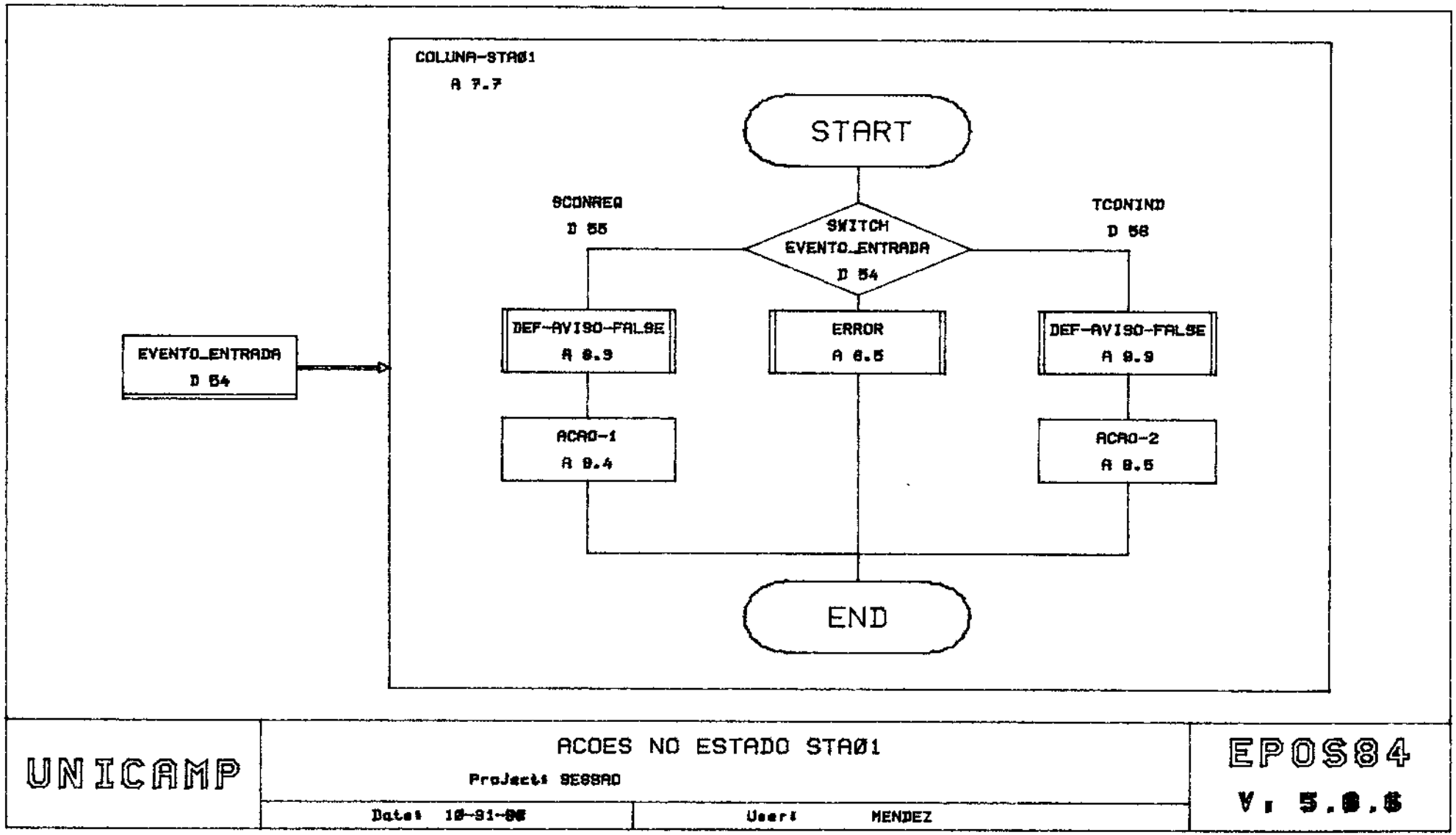



Figura 3.6: Diagrama "block", as ações a seguir quando o estado presente é STA01.

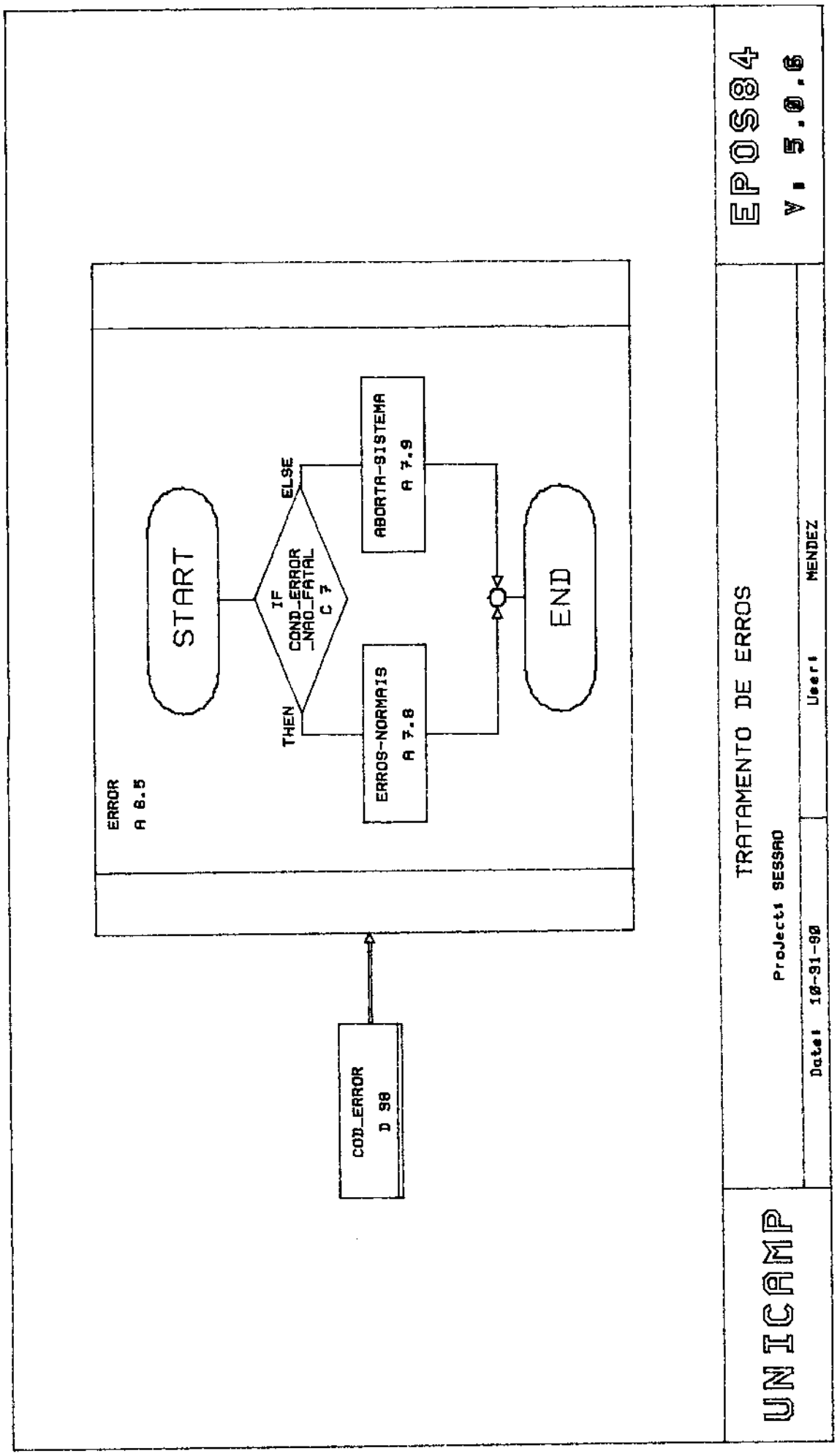



Figura 3.7: Diagrama "block" do procedimento que trata os erros na tarefa.

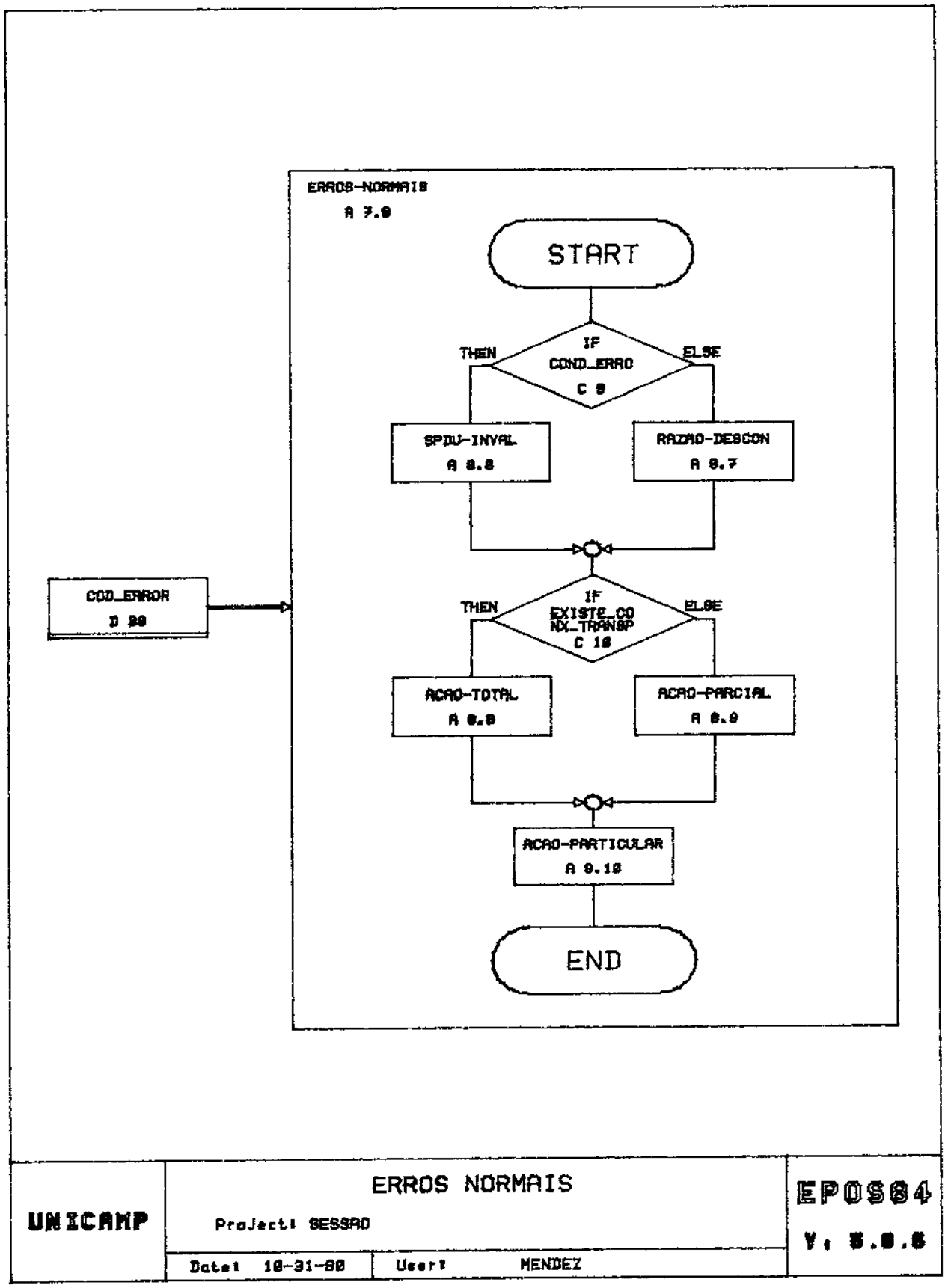



Figura 3.8: Diagrama "block" do procedimento que trata só erros na SPM.

3.2.3. Geração do código a partir da especificação.

No EPOS existem dois métodos para transformar a especificação em EPOS-S para código de uma linguagem alvo. Esses dois métodos são os seguintes:

(a). Seleção do código (Code selection): Neste método cada objeto de projeto que tem o <decomposition-part> também tem o <code-part> que implementa, num código escolhido, a especificação no <decomposition-part>. A referência a um objeto de projeto dentro do <decomposition-part> corresponderá a colocar um operador de inserção ("*>" para a linguagem "C") que apontará o objeto mencionado anteriormente e agora especificado no <code-part>. Por exemplo:

```
DECOMPOSITION:
    IF SPDU-VALIDA THEN
          TRATA-SPDU
    ELSE
          GERA-ERRO-2
    FI.
CODE:
    "if *>SPDU-VALIDA
        *>TRATA-SPDU
     else
        *>GERA-ERRO-2:".
```

Com este método só se consideram os <code-part>, onde os objetos de projeto indicados por "*>" são substituídos por seus <code-part> especificados. A substituição começa a partir do nível superior da hierarquia até chegar ao nível mais baixo. Desta forma é possível:

(1). Gerar código fonte para uma linguagem qualquer.

(2). Criar código eficiente, porque é utilizada a conversão manual de cada objeto de projeto em código fonte, o projetista deverá conferir a equivalência funcional entre o <decomposition-part> e seu <code-part> de um objeto de projeto.

(b). Transformação para um código (Code transformation): Neste método, a estrutura de controle do fluxo de execução das ações especificadas, é transformada automáticamente em uma linguagem alvo. Esta conversão considera também os objetos DATA e CONDITION, além do objeto EVENT no Composer ADA .

Pode-se dizer que o método (a) é semi-automático porque se precisa escrever o código para cada objeto de projeto e o Composer faz a substituição. Oferece a vantagem de gerar código em uma linguagem qualquer, só que ao escolher uma linguagem não suportada pelos Composer, por exemplo o Assembler, a substituição deverá ser manual.

O método (b) é automático no sentido que os Composer's geram o código na linguagem alvo escolhida (atualmente ADA, FORTRAN, PASCAL e C); mas nas ações terminais deverá inserir-se o código manualmente.

No caso deste projeto de software foi escolhido o método da "transformação para um código", porque a linguagem alvo escolhida é o "C" e o EPOS fornece o Composer correspondente.

Como exemplo de geração de código, usando o Composer "C", considere-se a ação COLUNA-STA01A, cuja especificação se mostra na figura 3.9 e o código correspondente, gerado pelo "composer", na figura 3.10.

```
ACTION COLUNA-STA01A .
DESCRIPTION :
PURPOSE :
"Esta ACTION pega o valor da variável EVENTO_ENTRADA e determina as
ações a seguir quando o estado presente é STA01A. " .
CATEGORY : 'extrato-resumo' .
DESCRIPTIONEND
DECOMPOSITION :
   SWITCH EVENTO_ENTRADA IN
      CASE CN :
         DEF-AVISO-FALSE ;
         ACAO-4
      CASE AB-NR :
         DEF-AVISO-FALSE ;
         ACAO-31
      CASE TDISIND :
         ACAO-7
      CASE TIM :
         DEF-AVISO-FALSE ;
         ACAO-8
      OUT :
         IF COND_1 THEN
            ACAO-5
         ELSE
            IF NOT COND_2 THEN
               ERROR USING ( COD_ERROR = ERR4 )
            FI
         FI
   NI .
INPUT : EVENTO_ENTRADA .
ACTIONEND
**
```

Figura 3.9: Especificação da ação COLUNA-STA01A.

Neste projeto não se especificou, em EPOS-S, a estrutura de dados, porque atualmente o Composer "C" não suporta a conversão de estruturas complexas de dados, como é o caso da estrutura de dados do SISDI_MAP (a qual é estabelecida nos anexos B e C desta tese); só convertendo estruturas simples como arranjos e "records". Nos <code-part> das ações terminais foram incluídos o manuseio dos parâmetros das primitivas, do serviço de sessão e transporte, e da informação contida nas SPDU's.

```
switch (EVENTO_ENTRADA) {
      case CN :
            {  AVISO = FALSE; }
              {
                VAR_INTERNA.ESTADO_PRESENTE[INDEX_CONTEXT] = STA01;
                printf("STA01");
                PRIM_TRANSP = TDISreq;
                ACAO_ESPECIFICA = A3;
                }
             GERA_PRIMITIVA_T ( PRIM_TRANSP );
             EXECUTA ( ACAO_ESPECIFICA );
             break;
       case AB_NR :
             {  AVISO = FALSE; }
             ACAO_31 ();
             break;
       case TDISIND :
             {
               VAR_INTERNA.ESTADO_PRESENTE[INDEX_CONTEXT]= STA01;
               printf("STA01");
               ACAO_ESPECIFICA=A3;
               VAR_INTERNA.ID_CONN[INDEX_CONTEXT] = 0;
               VAR_INTERNA.CONEXAO_TRANSP[INDEX_CONTEXT] = DESLIGADA;
               }
             EXECUTA ( ACAO_ESPECIFICA );
             break;
        case TIM :
             {  AVISO = FALSE; }
             ACAO_8 ();
             break;
        default:
             if (EVENTO_ENTRADA == AA ||EVENTO_ENTRADA == AB_R)
                 {
                  ACAO_5 ();
                  }
             else
                {
                 if ( ! (EVENTO_ENTRADA == AC || EVENTO_ENTRADA
                      == RF_NR || EVENTO_ENTRADA == RF_R ||
                      EVENTO_ENTRADA == DT || EVENTO_ENTRADA ==
                      DN || EVENTO_ENTRADA == FN_NR
                      || EVENTO_ENTRADA == FN_R) )
                   {
                    ERROR ( ERR4 );
                    }
                 }
              }
```

Figura 3.10: Código gerado automaticamente, pelo Composer-C, para a ação COLUNA-STA01A.

No anexo F desta tese apresenta-se a listagem completa do código, gerado automáticamente pelo Composer "C", para a especificação da implementação do protocolo de sessão. Foram contabilizados o número de linhas de código geradas automáticamente e aquelas que foram inseridas manualmente: além disso contabilizaram-se as sentenças de controle geradas automaticamente e feitas manualmente. Os resultados obtidos foram expressos em porcentagens e mostrados na tabela 3.4.

| Parâmetro de medição | Geração automática | Geração manual |
|---|---|---|
| linhas de código (LOC) | 55% | 45% (*) |
| if-else | 70% | 30% |
| switch-case | 93% | 7% |
| LOC de sentenças de atribuição | 0% | 100% |

(*) Decompõe-se em 41% de sentenças de atribuições e 4% de sentenças de controle.

Tabela 3.4: Resultados das medições feitas sobre o programa final obtido.

Da tabela 3.4. pode-se constatar, (apesar de a porcentagem de geração automática de LOC não ser muito grande em relação à manual), que as porcentagens de LOC's de sentenças de controle geradas automaticamente são muito maiores que as geradas manualmente: de onde resulta maior confiabilidade na especificação ao se optar por outra linguagem de programação. A especificação resultou ser praticamente independente da linguagem de implementação.

Finalmente salienta-se que, durante o desenvolvimento do programa, considerou-se o procedimento PROTOCOLO-DE-SESSAO, com as ações terminais contendo só sentenças do tipo "printf" e submetido a teste para verificação da lógica do protocolo de sessão, não se encontrou nenhum tipo de erro na sintaxe nem na lógica. Com isto pode-se afirmar que a lógica da SPM, especificada formalmente em EPOS-S, está de acordo com o estabelecido pelas tabelas de estado e portanto, a especificação pode servir como base para implementar o protocolo em outros ambientes de operação e em outras linguagens de programação.

# CAPÍTULO 4: EXEMPLO DE EXECUÇÃO.

O objetivo fundamental deste capítulo é esclarecer, através de um exemplo, a forma de abordagem da implementação até seus detalhes. Todos os objetos de projeto considerados neste exemplo foram especificados no <category-part> com a palavra chave "exemplo-exec".

Para tal efeito, pegou-se, como exemplo, a fase de estabelecimento da conexão de sessão, a qual foi representada na figura 2.2 no capítulo 2, com as seguintes hipóteses:

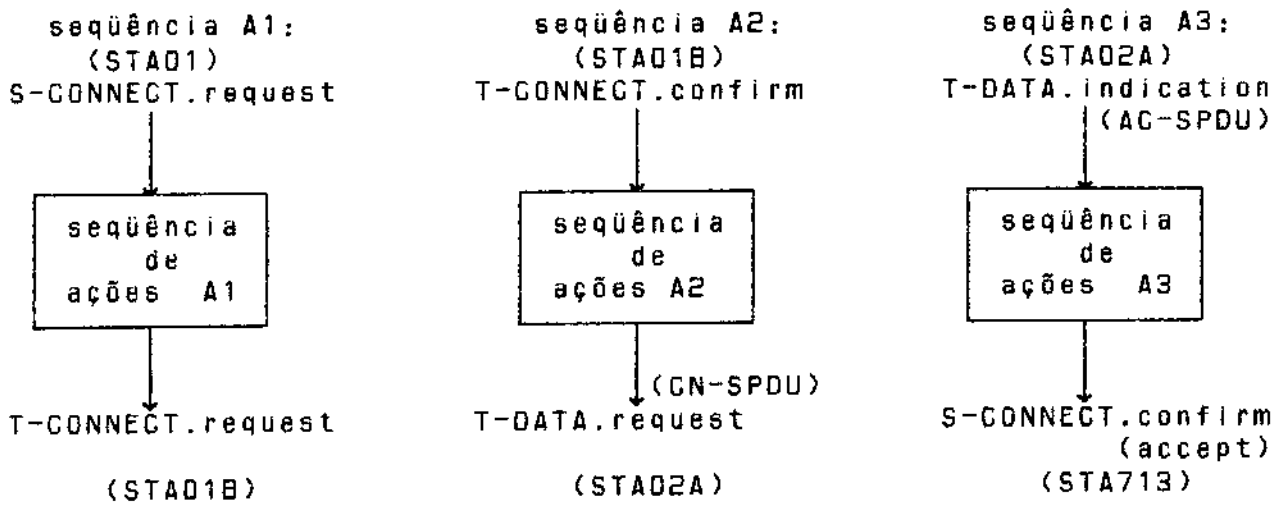
(a) A ação INICIALIZA-PROTOCOLO já foi executada, logo, a tarefa que executa o protocolo encontra-se pronta para atender as mensagens.

(b) De (a) deduz-se que no simulador não aconteceu nenhuma solicitação de conexão de sessão e que todos os contextos estão no estado inicial STA01.

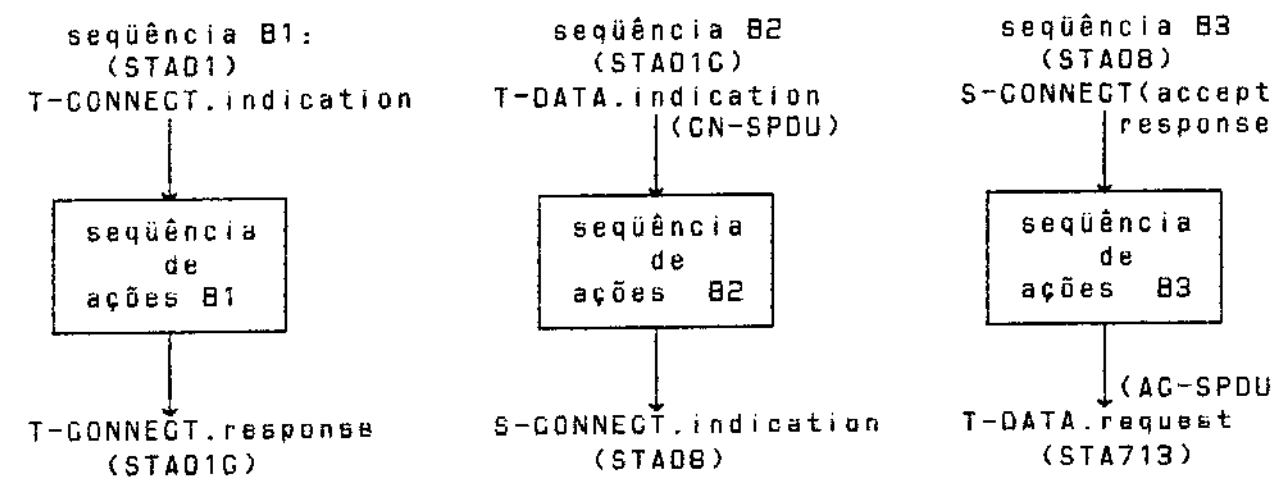
(c) O usuário-SS que solicita uma comunicação vai nomear-se "chamante A" e o usuário-SS que recebe a solicitação, "chamado B".

Para poder apresentar o exemplo em forma didática analisa-se do ponto de vista causa-efeito, ou seja, se à tarefa chega uma mensagem (causa) a tarefa tomará alguma ação (efeito). Se além disso considerarmos como referência o lado chamante e depois o lado chamado podem-se obter as sequências causa/efeito mostradas na figura 4.1 e que foram obtidas a partir do gráfico da figura 2.2:

Do lado "chamante A" tem-se:




Do lado "chamado B" tem-se:




Nota: Além das primitivas indica-se em cada gráfico, entre parênteses, o estado da SPM no momento de receber a mensagem e depois de gerar uma mensagem.
Nos T-DATA.req/ind indica-se, entre parênteses, a SPDU transportada pela primitiva-SS.

Figura 4.1: Sequência de eventos para o lado chamante A e para o lado chamado B, que acontecem numa fase de estabelecimento da conexão de sessão com sucesso.

Para o presente exemplo de execução, somente vão-se especificar as seqüências de ações do lado chamante A para evitar redundância no desenvolvimento das seqüências .

A seguir, é desenvolvida as seqüências de ações referenciando a especificação apresentada no Anexo E da tese. Será indicado o nome da ação e, onde necessário, dão-se detalhes da implementação, especialmente quando se trata de valores específicos de uma variável do exemplo. Se a ação for de tipo geral, assume-se suficiente a explicação dada no mesmo objeto de projeto especificado.

4.1. Seqüência A1: Chega uma mensagem correspondente a primitiva-SS S-CONNECT.request à porta de entrada s_pi_pres.

(1). CONTROLADOR-DO-PROVEDOR-SS: A condição IN-PORT-PRES é satisfeita. Isto significa que ao executar-se a primitiva do núcleo de tempo real "espera-mens (...)" ela devolve o valor OPERACAO_OK, e através da variável-ponteiro "block-sess", pega-se o endereço dos dados correspondentes à mensagem recebida na porta.

(2). PROCESSA-PORTA-APRESENTAÇÃO: Faz que o parâmetro de entrada CHAVE, do procedimento TRATA-MENS, seja PRES_INPUT para indicar que os dados vem desde a porta s_pi_pres.

(3). TRATAR-MENS.

(4). LER-ENTRADAS: Neste exemplo executa-se a ação LER-PORTA-PRES.

(5). LER-PORTA-PRES: Por ser a mensagem uma primitiva S-CONNECT.request implica que a informação de cabeçalho para esta mensagem deve ser:

block_sess->data.ses_serv.service_name = s_connection;
block_sess->primitive = request;
block_press->connection_id = um valor arbitrario,p.e.20.

Portanto, depois de executar-se esta ação terminal, tem-se:

NOME_SERVIGO = s_connection;
NOME_PRIMITIVA = request;
CONNECT_ID = 20;
TIPO_ERROR = 100;

(6). ENTREGA-CONTEXT-INDEX: Neste caso a ação devolverá o valor 1 porque não havia nenhum contexto ativo.

(7). ENTREGA-INCOMING-EVENT: Com os valores armazenados nas variáveis NOME_SERVICO(s_connection) e NOME_PRIMITIVA (request) e como a condição TIPO não é satisfeita, porque CHAVE é igual ao valor PRES_INPUT, dá inicio à execução da ação ABREVIA-EVENTO.

(8). ABREVIA-EVENTO: Como a condição EVENTO-DESCONH não se cumpre, porque o evento não é desconhecido, então, esta ação executa a ação TRATA-EVENTO.

(9). TRATA-EVENTO.

(10).PEGA-EVENTO: É a primeira ação de TRATA-EVENTO. Esta ação terminal faz a "tradução" da informação de cabeçalho capturada nas variáveis NOME_SERVIÇO e NOME_PRIMITIVA através do arranjo TAB_INC_EVENT, a qual foi apresentada na restrição CONSTRAINT 310 da especificação dos requisitos. O valor que deve gerar-se é SCONreq, o qual é armazenado na variável IN_COMING_EVENT. Como não se trata de um evento TCONind ou TDTind, então PEGA-EVENTO é a única ação executada dentro de TRATA-EVENTO.

Depois de terminar de executar completamente a ação TRATAR-MENS, a qual entregou os valores SCONreq para IN_COMING_EVENT, 1 para INDICE_CONTEXTO e 100 para RECEPCAO_INVALIDA, a seqüência volta à ação PROCESSA-PORTA-APRESENTACAO, a qual executa a ação DETERMINA-USAR-PROTOCOLO.

(11).DETERMINA-USAR-PROTOCOLO: Como a condição COND_RECEP_VALIDA é satisfeita, esta ação executa PROTOCOLO-DE-SESSAO cujos valores de entrada estão armazenados nas variáveis INCOMING_EVENT e VAR_INTERNA!ESTADO_PRESENTE(INDEX_CONTEXT). Esta última é equivalente a VAR_INTERNA.ESTADO.PRESENTE[INDEX_CONTEXT] na linguagem C. Tais valores são: SCONreq para INCOMING_EVENT e STAD1 para VAR_INTERNA!ESTADO_PRESENTE(1).

(12).PROTOCOLO-DE-SESSAO: Como o valor da variável de seleção ESTADO_ATUAL é STAD1 (repassado por VAR_INTERNA!ESTADO_PRESENTE(1)), esta ação executa COLUNA-STAD1.

(13).COLUNA-STAD1: Como a variável de seleção EVENTO_ENTRADA

tem o valor SCONreq, esta ação executa DEF-AVISO-FALSE e posteriomente ACAO-1.

(14).DEF-AVISO-FALSE.

(15).ACAO-1: Esta ação vai executar todas as tarefas indicadas na SPM quando o estado presente é STA01 e chega o evento SCONreq.

(16).DEFINE-SITUACAO-1: é importante indicar que nesta ação a SPM do contexto 1, que representa ao usuário-SS chamante A, muda para o estado STA01B.

(17).GERA-PRIMITIVA-T.

(18).PREPARA-PRIMITIVA-T: Como o valor da variável T-PRIMITIVA é TCONreq (repassada por PRIM_TRANSP em ACAO-1), então é executado TCON-REQ.

(19).TCON-REQ: Esta ação terminal prepara os parâmetros da mensagem correspondente a uma primitiva T-CONNECT.request. O parâmetro QOS transportado pela mensagem recebida, correspondente a S-CONNECT.request, é enviado nesta ação para ser negociado entre as camadas de transporte de ambos usuários-SS comunicantes.

Depois de executar-se a ação terminal TCON-REQ, a qual gerou uma mensagem correspondente à primitiva T-CONNNECT.request, a seqüência volta à ação ACAO-1, e executa a ação EXECUTA, cuja variável de entrada ACAO_ESPECIFICA tem o valor A2.

(20).EXECUTA: Como o valor da variável de seleção SPECIFIC_ACTION é A2, então esta ação ordena executar ACAO-ESPECIFICA-2.

(21).ACAO-ESPECIFICA-2: é executada a ação específica [2] definida pela ISO. Sendo esta a última ação para a seqüência A1.

4.2. <u>Seqüência A2</u>: A seguir, apresenta-se a seqüência A2 que acontece quando o estado da SPM é STA01B e chega uma mensagem correspondente a uma primitiva T-CONNECT.confirm (o que indica que a conexão de transporte foi estabelecida, ficando disponível para transportar SPDUs). A mensagem chega à porta de entrada s_pi_transp proveniente da camada de transporte,

gerando-se a seguinte seqüência de ações:

(1). CONTROLADOR-DO-PROVEDOR-SS: A condição IN-PORT-TRANSP é satisfeita, logo, através da primitiva do núcleo de tempo real espera-mens(...) é capturado o ponteiro que possue o endereço do conjunto de dados da mensagem recebida. Esta ação indica que deve executar-se PROCESSA-PORTA-TRANSPORTE. O valor do ponteiro fica armazenado na variável-ponteiro block_sess.

(2). PROCESSA-PORTA-TRANSPORTE: Quando executa TRATAR-MENS define-se o valor da variável CHAVE pelo valor TRANSP_INPUT, para indicar que a mensagem para processar provem da camada de transporte.

(3). TRATAR-MENS.

(4). LER-ENTRADAS: Como o valor de CHAVE é TRANS_INPUT, então, esta ação executa LER-PORTA-TRANSP.

(5). LER-PORTA-TRANSP: Como a mensagem recebida corresponde a uma primitiva T-CONNECT.confirm, então, a informação de cabeçalho da mensagem recebida deve ser:

block_sess->data.trans_serv.service_name = t_connection;
block_sess->primitive = confirm_pos;
block_sess->connection-id = 20;

Após de executada a ação, as variáveis internas ficam com os seguintes valores:

NOME_SERVICO = t_connection;
NOME_PRIMITIVA = confirm_pos;
CONNECT_ID = 20;
TIPO_ERROR=100; esta variável só será alterada em TRATAR-MENS se acontecer algum erro no processamento.

Depois de executar-se esta ação a seqüência volta a TRATAR-MENS.

(6). ENTREGA-CONTEXT-INDEX: A condição COND_ENTRADAS, em TRATAR-MENS, foi satisfeita, logo, ENTREGA-CONTEXT-INDEX é executada. Como resultado desta ação o índice de contexto deverá ser 1 porque o CONNECT_ID é 20 e esse valor já existia na tabela de contexto alocada em VAR_INTERNA!!D_CONN(1).

Depois de executar-se esta ação, a seqüência volta a TRATAR-MENS.

(7). ENTREGA-INCOMING-EVENT: Esta ação é executada porque a condição COND_ENTRADAS é satisfeita. Como os valores armazenados nas variáveis NOME_SERVICO (t_connection) e NOME_PRIMITIVA (confirm_pos) e como a condição TIPO não é satisfeita, porque CHAVE é igual a TRANSP_INPUT, esta ação executa ABREVIA-EVENTO.

(8). ABREVIA-EVENTO: Como a condição EVENTO_DESCONH não se cumpre, porque o evento não é desconhecido, a ação executa TRATA-EVENTO.

(9). TRATA-EVENTO.

(10).PEGA EVENTO: Esta ação, finalmente, faz a tradução da informação de cabeçalho da mensagem recebida, armazenada nas variáveis NOME_SERVICO e NOME_PRIMITIVA, em um valor "numerado" que corresponde ao evento TCONcnf. A tradução é feita utilizando o arranjo bidimensional TAB_ING_EVENT.

PEGA-EVENTO é a última ação controlada por TRATAR-MENS, ficando definidos os seguintes valores: TCONcnf para INCOMING_EVENT, 1 para INDICE_CONTEXTO e 100 para RECEPCAO_INVALIDA. A seqüência volta à ação PROCESSA-PORTA-TRANSPORTE, onde se executa DETERMINA-USAR-PROTOCOLO.

(11).DETERMINA-USAR-PROTOCOLO: Como a condição COND_RECEP_VALIDA é satisfeita, executa-se a ação PROTOCOLO-DE-SESSAO cujos valores para seus parâmetros de entrada são: SCONreq para INCOMING_EVENT e STA01B para VAR_INTERNA!ESTADO_PRESENTE(1).

(12).PROTOCOLO-DE-SESSAO: Como o valor da variável de seleção ESTADO_ATUAL, é STA01B, então esta ordena a execução de COLUNA-STA01B.

(13).COLUNA-STA01B: A variável de seleção EVENTO_ENTRADA tem o valor TCONcnf, que determina executar a ação DEF-AVISO-FALSE e posteriormente ACAO-38.

(14).DEF-AVISO-FALSE.

(15).ACAO-38: Esta ação val executar todas as tarefas que a

SPM estipula quando o estado presente for STA01B e chegar o evento TCONcnf.

(16).DEFINE-SITUACAO-3B: É importante indicar que nesta ação a SPM do contexto 1, que representa o usuário-SS chamante A, muda para o estado STA02A.

(17).GERA-SPDU: Neste exemplo atenderá uma SPDU CONNECT.

(18).CODIFICA-SPDU: Como o valor da variável de seleção, NOME_SPDU, é CN, executa-se GERA-SPDU-CN.

(19).GERA-SPDU-CN: Esta ação terminal encarrega-se de montar os parâmetros da SPDU CONNECT.

(20).ENVIA-TDATAREQ: Gera uma mensagem correspondente à primitiva T-DATA.request, que vai transportar a SPDU CONNECT montada previamente, com o que começa o processo de estabelecimento da conexão de sessão. A SPDU CN transporta a informação contida na primitiva S-CONNECT.request recebida anteriormente.

4.3. Seqüência A3: Nesta última seqüência do exemplo de execução, chega à tarefa, proveniente da camada de transporte, uma mensagem enviada pelo outro contexto que representa o usuário-SS chamante B. A mensagem corresponde a uma primitiva T-DATA.indication, que transporta uma SPDU ACCEPT (indicando assim, que a solicitação da conexão de sessão foi aceita pelo outro usuário). O estado atual do chamante A, representado pelo contexto 1, é STA02A. Logo, a seqüência em resposta à mensagem recebida é a seguinte:

(1). CONTROLADOR-DO-PROVEDOR-SS : A condição IN-PORT-TRANSP é satisfeita; o endereço dos dados da mensagem é capturado e armazenado na variável-ponteiro block_sess. Esta ação ordena que deve executar-se PROCESSA-PORTA-TRANSPORTE.

(2). PROCESSA-PORTA-TRANSPORTE: Idem seqüência A2.

(3). TRATAR-MENS.

(4). LER-ENTRADAS: Idem seqüência A2.

(5). LER-PORTA-TRANSP: Como a mensagem recebida corresponde a uma primitiva T-DATA.indication, a informação de cabeçalho da mensagem deve ser:
block_sess->data.transp_serv.service_name = t_data;

block_sess->primitive = indication;

block_sess->connection_id = 20;

Depois de haver-se executado esta ação as variáveis internas ficam com os seguintes valores:

NOME_SERVICO = t_data;

NOME_PRIMITIVA = indication;

CONNECT_ID = 20;

TIPO_ERROR = 100;

A seqüência volta a TRATAR-MENS.

(6). ENTREGA-CONTEXT-INDEX: é idêntica à seqüência A2.

(7). ENTREGA-INCOMING-EVENT: é similar à seqüência A2, só mudando os valores de NOME_SERVICO e NOME_PRIMITIVA, definidos no passo (5) desta seqüência.

(8). ABREVIA-EVENTO: Idem à seqüência A2.

(9). TRATA-EVENTO.

(10). PEGA-EVENTO: Utilizando a tabela TAB_INC_EVENT e os valores das variáveis NOME_SERVICO e NOME_PRIMITIVA, define temporariamente por TDTind, o valor da variável IN_COMING_EVENT porque TDTind não é um evento padrão da ISO. Neste caso quem define o evento padrão é o tipo de SPDU que chegou.

A seqüência volta a TRATA-EVENTO.

(11). TRATA-EVENTO-TDTIND: Verifica a validade da SPDU que chegou. Suponha-se, para este exemplo, que a SPDU é válida, o que leva à execução de TRATA-SPDU.

(12). TRATA-SPDU: Lê o código da SPDU transportada. Neste exemplo é uma SPDU ACCEPT, resultando a execução de EVENTO-AC.

(13). EVENTO-AC: Esta ação preenche a variável apontada por INCOMING_EVENT com o valor final AC.

Com esta ação termina-se a execução de TRATA-EVENTO-TDTIND, encerrando também a ação TRATAR-MENS e gerando como saídas os seguintes valores:
AC para INCOMING_EVENT; 1 para INDICE_CONTEXTO e 100 para RECEPCAO_INVALIDA.

A seqüência volta à PROCESSA-PORTA-TRANSPORTE, que executa DETERMINA-USAR-PROTOCOLO.

(14).DETERMINA-USAR-PROTOCOLO: Como a condição COND_RECEP_VALIDA é satisfeita, executa-se a ação PROTOCOLO-DE-SESSAO cujos valores, para seus parâmetros de entrada, são:
AC para INCOMING_EVENT e STA02A para VAR_INTERNA!ESTADO_PRESENTE(1).

(15).PROTOCOLO-DE-SESSAO: Como o valor da variável de seleção, ESTADO_ATUAL, é STA02A, executa-se COLUNA-STA02A.

(16).COLUNA-STA02A: Como a variável de seleção EVENTO_ENTRADA tem o valor AC, será executada ACAO-14.

(17).ACAO-14: Esta ação realiza todas as tarefas que a SPM indica quando o estado presente é STA02A e chega o evento AC.

(18).DEFINE-SITUACAO-14: é a primeira ação que ACAO-14 ordena. Nesta ação a SPM do contexto 1, que representa o usuário chamante A, muda para o estado STA713.

(19).GERA-PRIMITIVA-S: É implementada pela ação PREPARA-PRIMITIVA-S.

(20).PREPARA-PRIMITIVA-S: O valor da variável de seleção S_PRIMITIVA é SCONcnf_pos,o qual foi definido na ação DEFINE-SITUACAO-14, ou seja executa-se o procedimento PREP-PRIM-SCONCNF com o parâmetro de entrada CNF igual a POS, (para indicar que deverá gerar-se uma mensagem correspondente a uma primitiva S-CONNECT(accept).confirm).

(21).PREP-PRIM-SCONCNF: Como o valor do parâmetro de entrada é POS, então esta ação vai montar os parâmetros da primitiva S-CONNECT(accept).confirm e posteriormente vai enviar a mensagem à porta de entrada da camada de apresentação. A seqüência volta à ACAO-14, que ativa EXECUTA cujo parâmetro de entrada ACAO_ESPECIFICA é A5, definido na ação DEFINE-SITUACAO-14.

(22).EXECUTA: Como o valor da variável de seleção SPECIFIC_ACTION é A5, então esta ação ativará ACAO-ESPECIFICA-5.

(23).ACAO-ESPECIFICA-5: Esta ação executa a ação especifica [5] definida pela ISO.

Após esta seqüência a SPM do usuário chamante A, representado pelo contexto 1, fica no estado STA713 esperando executar a transferência de dados, já que a conexão de sessão foi estabelecida com sucesso.

É importante ressaltar que a troca de mensagens resulta, na realidade, na troca dos ponteiros que permitem posterior acesso ao bloco de estrutura de dados.

# CAPITULO 5: CONCLUSÕES.

As conclusões que neste capítulo se expõem são o resultado da experiência obtida no desenvolvimento deste projeto, utilizando a ferramenta de engenharia de software conhecida como EPOS, para a especificação e implementação do protocolo de sessão que rodará no ambiente de simulação SISDI_MAP.

Em primeiro lugar, os dois grandes objetivos propostos inicialmente (especificação e implementação) foram atingidos, dando como resultados os documentos para a especificação dos requisitos e da implementação (Anexos D e E respectivamente) e do programa gerado na linguagem "C" (Anexo F) obtido a partir da especificação desenvolvida.

As conclusões que se podem tirar sobre o trabalho realizado são as seguintes:

(1) A abordagem escolhida como modelo de implementação permitiu que a lógica da SPM esteja contida completamente em um único procedimento (PROTOCOLO-DE-SESSAO). Os únicos elementos considerados como externos ao protocolo são: o índice de contexto, a variável global AVISO e as ações terminais que estão fortemente ligadas com o ambiente em que vai rodar a implementação final. Desta forma, a especificação da lógica da execução do protocolo é completamente independente do ambiente em que rodará o programa gerado a partir dela, e independente da estrutura de dados escolhida para os parâmetros das primitivas e das SPDUs . No caso de troca de ambiente de operação, será necessário reespecificar o procedimento TRATAR-MENS que abrange a interface de entrada e tradução de eventos, revisar o uso do índice de contexto e da variável AVISO, e mudar o conteúdo das ações terminais onde se encontra a interface de saída. No caso de querer expandir para implementar outras unidades funcionais, sem mexer na lógica já especificada, será necessário remover, de seu lugar atual, o procedimento ERROR e substituí-lo pela ação que especifica a expansão da implementação.

(2) A metodologia utilizada para especificar a lógica da SPM, baseada em "switch-case", fornece mecanismos suficientes para que as possibilidades de erro na interpretação da tabela de estado sejam mínimas. Além disso, a especificação passou por um processo de análise de erros (sintaxe, objetos isolados funcionalmente, objetos não referenciados, etc) que permitiu que a fase de teste de implementação fosse rápida e eficiente; de fato, o teste da lógica do procedimento PROTOCOLO-DE-SESSAO teve sucesso na primeira execução.

(3) O EPOS-S permite especificar estruturas complexas de dados, como as do SISDI_MAP, mas na presente versão do EPOS, o Composer-C (e também os Composer's ADA, FORTRAN e PASCAL) só permitem a especificação de estruturas simples de dados (arranjos e "records"). Devido a este fato, a estrutura de dados do SISDI_MAP não foi especificada em EPOS mas diretamente escrita na linguagem "C", fazendo só uso dos <code-part> das ações terminais.
Além disso, o Composer-C também não reconhece os objetos EVENT, INTERFACE e DEVICE, nem as sentenças de processamento paralelo; por esse motivo não foram especificadas as interfaces nem foram consideradas as mensagens como eventos.

(4) As regras de sintaxe da especificação (principalmente no <decomposition-part>) são muito mais simples que as regras de sintaxe da linguagem "C", o que torna mais fácil a revisão e mudanças na estrutura da especificação (incluindo a fase de análise por EPOS-A).

(5) O tempo ocupado na especificação é recuperado na compilação do programa gerado automaticamente porque o Composer-C se preocupa em gerar um código livre de erros de sintaxe.

(6) Durante o desenvolvimento da especificação, o EPOS fornece documentação gráfica, ajudando ao projetista na revisão da lógica, e facilitando o desenvolvimento do projeto (a documentação gerada pode ser anexada diretamente ao relatório final do projeto).

(7) Em relação à comparação entre o código gerado automaticamente e o código escrito à mão para este projeto, o qual está expresso na tabela 3.4, pode-se concluir o seguinte:

(a) Apesar da quantidade de Linhas de Código geradas automaticamente não ser muito maior (55%), comparada com as Linhas de Código geradas à mão, a maior porcentagem de sentenças de controle encontra-se no código gerado automaticamente (70% para if-else e 93% para switch-case), o que implica em que a lógica do programa dependa fortemente da especificação (expresso nos <decomposition-part>) e menos do código escrito à mão.

(b) Como conseqüência de (a) e considerando que as sentenças nos <code-part>, na sua maioria, são sentenças de atribuição (que podem ser mudadas facilmente para outra linguagem), pode-se afirmar que o projeto resultou pouco dependente da linguagem de programação escolhida.

(8) O programa gerado automaticamente é pouco amigável, devido ao não aproveitamento da modularidade que a linguagem "C" oferece e à não exploração da otimização do código que a linguagem permite. A manutenção, no caso de modificações diretas no código fonte, será difícil. Mas, o código é um produto da especificação do projeto e, portanto, qualquer modificação será feita na especificação, a qual é muito mais explicativa, modular e documentada (tanto texto como gráfica) que o código final.

(9) Com relação ao projeto SISDI_MAP, no qual está inserido este projeto, podem-se fazer os seguintes comentários:

(a) A estrutura de dados do SISDI_MAP, utilizada para especificar os parâmetros das primitivas de serviço e das SPDU's, facilitou enormemente a implementação final porque, no caso das SPDU's, não se precisou especificar um codificador/decodificador[16]. Mas, a estrutura das SPDU's

não está de acordo com o padrão estabelecido pela ISO.

(b) Devido a que o bloco de dados contêm parâmetros de todas as camadas (Aplicação, Apresentação, Sessão e Transporte) pertencentes a um par de contextos que representam dois usuários que estão se comunicando, a estrutura de dados da camada depende das outras, porque qualquer mudança no comprimento dos parâmetros mais longos fará mudar o comprimento das máscaras. Para evitar esta situação, poderá ser melhor que cada camada crie e utilize seu próprio bloco de dados para cada par de usuários comunicantes, utilizando a mesma estrutura já definida para a camada e apagando-a depois de desligar a conexão.

(10)Como conclusão final pode-se dizer que a obtenção de um documento que especifica a implementação de um protocolo, neste caso camada de sessão, permite ter uma visão mais concreta sobre o protocolo especificado pela ISO. Como por outro lado a especificação escrita em EPOS pode ser convertida em qualquer linguagem de programação, obtem-se um documento mais próximo da implementação.

# REFERÊNCIAS BIBLIOGRÁFICAS


[1] Manuel de Jesus Mendes, "IV Escola Brasileiro Argentina de Informática (EBAI)", 1989.

[2] Durval Jacintho, "Aspectos de especificação e implementação da estrutura de mensagens de um sistema didático do protocolo MMS", Tese Mestrado, DCA-FEE-UNICAMP, Novembro 1989.

[3] Maria C. Amim Zabeu, "Um modelo para configuração de Núcleos para Tempo Real", Tese Mestrado, DCA-FEE-UNICAMP, 1989.

[4] R.J.Lauber e P.R.Lempp, "EPOS OVERVIEW", Setembro 1986.

[5] Manual EPOS-R, versão 4.0,1988.

[6] Manual EPOS-S, versão 5.0,1989.

[7] Manual EPOS-P, versão 5.0,1989.

[8] Manual EPOS-C, versão 5.0,1989.

[9] Manual EPOS-A, versão 5.0,1989.

[10] Manual EPOS-M, versão 5.0,1989.

[11] Manual EPOS-D, versão 5.0,1989.

[12] Manual EPOS Code Transformation C (Composer-C), para EPOS versão 5.0, Agosto 1989.

[13] Andrew S. Tanenbaum, "Computer Network", 2° Edição, 1987, Prentice-Hall.

[14] "Basic Connection Oriented Session Service Definition", ISO 8326.

[15] "Basic Connection Oriented Session Protocol Specification", ISO 8327.

[16] Liba Svobodova, "Implementing OSI Systems", IEEE Journal on Selected Areas in Communications, Vol.7, No7, Setembro 1989, pp.1115-1130.

[17] Paulo M. Inazumi, Tese de Mestrado, realizado em paralelo com a presente tese, sobre especificação da implementação da camada de apresentação.

[18] J.A.B.Moura, J.P.Sauvé, W.F.Giozza e J.F.Marinho de Araujo, "Redes Locais de Computadores: Protocolos de Alto Nível e Avaliação de Desempenho", Editorial McGraw-Hill, 1986.

[19] F.Caneshi,"Hints for the interpretation of the ISO Session Layer", ACM SIGCOMM Computer Communication Review, Vol.16, No4, Julho/Agosto 1986, pp.34-72.

[20] F.Halsall, "Data Communications, Computer Networks and OSI", Editorial Addison-Wesley, 2° Edição, 1988.

# ANEXO A : TABELAS DE ESTADO DA SPM QUE FORAM IMPLEMENTADAS.

Tabela 1: Tabela de estado para a fase de estabelecimento da conexão de sessão:

| State / Event | STA01 idle no TC | STA01A await AA | STA01B await TCONcnf | STA01C idle TC con | STA02A await AC | STA08 await SCONrsp | STA16 await TDISind |
|---|---|---|---|---|---|---|---|
| AC | // | STA01A | // | TDISreq STA01 | SCONcnf + [5] [11] STA713 (6) | | STA16 |
| CN | // | TDISreq [3] STA01 | // | ^p01 SCONind STA08<br>p01 TDISreq STA01 | | | TDISreq [3] STA01 |
| RF-nr | // | STA01A | // | TDISreq STA01 | SCONcnf – TDISreq STA01 | | STA16 |
| RF-r | // | STA01A | // | TDISreq STA01 | ^p02 SCONcnf – TDISreq STA01<br>p02 SCONcnf – STA01C | | STA16 |
| SCONreq | TCONreq [2] STA01B | | | p01 CN STA02A | | | |
| SCONrsp + | | | | | | AC [5] [11] STA713 | |
| SCONrsp – | | | | | | ^p02 RF-nr [4] STA16<br>p02 RF-r STA01C | |
| TCONcnf | // | // | CN STA02A | // | // | // | // |
| TCONind | TCONrsp [1] STA01C | // | // | // | // | // | // |

Tabela 2: Tabela de estado para a fase de transferência de dados:

| State / Event | STA01A await AA | STA01C idle TC con | STA03 await DN | STA09 await SREL rsp | STA16 await TDIS ind | STA713 data transfer |
|---|---|---|---|---|---|---|
| DT | STA01A | TDIS req STA01 | p05 & p10 SDT ind STA03 | | STA16 | p05 SDT ind STA713 |
| SDT req | | | | p04 DT STA09 | | p03 DT STA713 |

Tabela 3: Tabela de estado para a fase de liberação da conexão de sessão:

| State / Event | STA01A await AA | STA01C idle TC con | STA03 await DN | STA09 await SREL rsp | STA16 await TDIS ind | STA713 data transfer |
|---|---|---|---|---|---|---|
| DN | STA01A | TDIS req STA01 | ^p66 SRELcnf+ TDISreq STA01<br>p66 SRELcnf+ STA01C | p69 & ^p01 SRELcnf+ [32] STA09 | STA16 | |
| FN-nr | STA01A | TDIS req STA01 | ^p65 SRELind [8] [18] STA09 | | STA16 | p68 SRELind [8] STA09 |
| FN-r | STA01A | TDIS req STA01 | ^p65 & ^p01 & p16 SRELind [8] [18] STA09 | | STA16 | p68 & ^p01 & p16 SRELind [9] STA09 |
| SRELreq | | | | ^p65 FN-nr [8] [18] STA09 | | p63 & ^p64 FN-nr [8] STA03<br>p63 & p64 FN-r [7] STA03 |
| SRELrsp+ | | | | ^p66 & p75 DN [4] STA16<br>p66 DN STA01C<br>p69 & p01 DN STA03 | | |

Tabela 4: Tabela de estado no caso de aborto da conexão de sessão:
(continua na próxima folha)

| State / Event | STA01 idle no TC | STA01A await AA | STA01B await TCON cnf | STA01C idle TC con | STA02A await AC | STA03 await DN |
|---|---|---|---|---|---|---|
| AA | // | [3] STA01C | // | TDIS req STA01 | | |
| AB-nr | // | [3] TDIS req STA01 | // | TDIS req STA01 | SxABind (3) TDIS req STA01 | SxABind (3) TDIS req STA01 |
| AB-r | // | [3] STA01C | // | ^pO2 TDIS req STA01 pO2 AA STA01C | ^pO2 SxABind (3) TDIS req STA01 pO2 SxABind (3) AA STA01C | ^pO2 SxABind (3) TDIS req STA01 pO2 SxABind (3) AA STA01C |
| SUABreq | | | TDIS req STA01 | | ^pO2 AB-nr [4] STA16 pO2 AB-r [4] STA01A | ^pO2 AB-nr [4] STA16 pO2 AB-r [4] STA01A |
| TDISind | // | [3] STA01 | SPABind STA01 | STA01 | SPABind STA01 | SPABind STA01 |
| TIM | // | TDIS req STA01 | // | // | // | // |

Continuação da Tabela 4: Tabela de estado no caso de aborto da conexão de sessão:

| State / Event | STA 08 await SCONrsp | STA09 await SRELrsp | STA 16 await TDISind | STA 713 data transfer |
|---|---|---|---|---|
| AA | | | [3] TDISreq STA01 | |
| AB-nr | SxABind (3) TDISreq STA01 | SxABind (3) TDISreq STA01 | [3] TDISreq STA01 | SxABind (3) TDISreq STA01 |
| AB-r | ^p02 SxABind (3) TDISreq STA01 p02 SxABind (3) AA STA01C | ^p02 SxABind (3) TDISreq STA01 p02 SxABind (3) AA STA01C | [3] TDISreq STA01 | ^p02 SxABind (3) TDISreq STA01 p02 SxABind (3) AA STA01C |
| SUABreq | ^p02 AB-nr [4] STA16 p02 AB-r [4] STA01A | ^p02 AB-nr [4] STA16 p02 AB-r [4] STA01A | | ^p02 AB-nr [4] STA16 p02 AB-r [4] STA01A |
| TDISind | SPABind STA01 | SPABind STA01 | [3] STA01 | SPABind STA01 |
| TIM | // | // | TDISreq STA01 | // |

# ANEXO B : ESTRUTURA DE DADOS DOS PARÂMETROS DAS PRIMITIVAS E SPDUs PARA A CAMADA DE SESSÃO.

```
/*-------------------------------------------------------------
          MENSAGENS NAS INTERFACES DO SISTEMA DIDATICO
 -------------------------------------------------------------*/

/* OBS. : Todos os tipos declarados como apontador para "char"
 *        terao um tamanho limitado pelo Sistema Didatico
 */


/*--------------------- INTERFACES DE PROTOCOLO ---------------*/


/*************************************************************
                    TIPOS   ESTRUTURADOS
 *************************************************************/


/*-------------------------------------------
         Estrutura do servico "S_CONNECTION"
  -------------------------------------------*/


/*-----------------------------------------------
  Estrutura para S_CONNECTION.request/indication
  -----------------------------------------------*/

typedef struct
        {
        Boolean                 valid_pdu; /* so para indication */
        Session_con_id          session_connection_ident;
        S_address               calling_sess_address;
        S_address               called_sess_address;
        uint8                   quality_of_service;
        uint16                  session_requirements;
        Boolean                 init_point_serial_number_valid;
        uint32                  init_point_serial_number;
        Boolean                 init_assigment_of_tokens_valid;
        char                    init_assigment_of_tokens;
        Boolean                 user_data_valid;
        CP_ppdu                 user_data; /* 108 bytes */
        } S_connect_ri_struct;  /* 144 bytes comprimento estrutura */


/*-----------------------------------------------
  Estrutura para S_CONNECTION.response/confirm
  -----------------------------------------------*/

typedef struct
        {
        Boolean                 valid_pdu; /* so para confirm */
        Session_con_id          session_connection_ident;
```

```
        S_address               called_sess_address;
        S_con_result            result;
        uint8                   quality_of_service;
        uint16                  session_requirements;
        Boolean                 init_point_serial_number_valid;
        uint32                  init_point_serial_number;
        Boolean                 init_assigment_of_tokens_valid;
        char                    init_assigment_of_tokens;
        Boolean                 user_data_valid;
        union
          {
          CPA_ppdu              cpa_ppdu; /* conexao aceita    */
          CPR_ppdu              cpr_ppdu; /* conexao rejeitada (58b)*/
          } user_data;
        } S_connect_rc_struct; /* 93 bytes comprimento estrutura */

/*-----------------
  Estrutura do servico de Sessao : S_CONNECTION
  -----------------*/

typedef union
        {
        S_connect_ri_struct     s_connection_req;
        S_connect_ri_struct     s_connection_ind;
        S_connect_rc_struct     s_connection_resp;
        S_connect_rc_struct     s_connection_conf;
        } S_connection_struct;



/*-----------------------------------------
        Estrutura do servico "S_RELEASE"
  -----------------------------------------*/


/*--------------------------------------
  Primitiva S_RELEASE.request/indication
  --------------------------------------*/

typedef struct
        {
        Boolean                 valid_pdu; /* somente na indication */
        Boolean                 user_data_valid;
        A_release_pdu           user_data;      /* 1 a 512 octetos */
        } S_release_ri_struct;


/*-------------------------------------
  Primitiva S_RELEASE.response/confirm
  -------------------------------------*/

typedef struct
        {
        Boolean                 valid_pdu; /* somente na confirm */
        P_release_result        result;
        Boolean                 user_data_valid;
```

```
        A_release_pdu           user_data;
        } S_release_rc_struct;


/*-----------------------------------------
  Estrutura do servico de Sessao : "S_RELEASE"
  -----------------------------------------*/
typedef union
        {
        S_release_ri_struct     s_release_req;
        S_release_ri_struct     s_release_ind;
        S_release_rc_struct     s_release_resp;
        S_release_rc_struct     s_release_conf;
        } S_release_struct;


/*-----------------------------------------
        Estrutura do servico "S_U_ABORT"
  -----------------------------------------*/

/*-----------------------------------------
  Estrutura dos parametros do servico S_U_ABORT.req/ind
  -----------------------------------------*/

typedef struct
        {
        Boolean                 valid_pdu; /* so no indication */
        Boolean                 user_data_valid;
        ARU_ppdu                user_data;  /* 1 a 9 octetos */
        } S_parameter_u_abort_struct;

/*-----------------------------------------
  Estrutura do servico de Sessao : "S_U_ABORT"
  -----------------------------------------*/

typedef union
        {
        S_parameter_u_abort_struct      s_u_abort_req;
        S_parameter_u_abort_struct      s_u_abort_ind;
        } S_U_abort_struct;


/*-----------------------------------------
         Estrutura do servico "S_P_ABORT.ind"
  -----------------------------------------*/


typedef struct
        {
        char    reason[40];
        } S_parameter_p_abort_struct;


typedef struct
        {
        S_parameter_p_abort_struct   s_p_abort_ind; /* somente primit.
```

```
                                                        indication */
        } S_P_abort_struct;


/*-------------------------------------------
         Estrutura do servico "S_DATA"
  -------------------------------------------*/

/*-------------------------------------------
  Estrutura dos parametros do servico S_DATA.req/ind
  -------------------------------------------*/

typedef struct
        {
        Boolean          valid_pdu;
        Boolean          user_data_valid;
        union {
                Mms_pdu_struct          pdu_valid;
                invalid_pdu_struct      pdu_invalid;
              } user_data;
        } S_parameter_data_struct;


/*-------------------------------------------
  Estrutura do servico de Sessao : "S_DATA"
  -------------------------------------------*/

typedef union
        {
        S_parameter_data_struct          s_data_req;
        S_parameter_data_struct          s_data_ind;
        } S_data_struct;


/*-------------------------------------------
  Estrutura de Servicos de Sessao nas Interfaces N6/N5
  -------------------------------------------*/

typedef struct
        {
         Session_service                 service_name;
         Dummy                           dummy_para_transporte[48];
                                         /* dummy A */

         union
              {
              S_connection_struct        s_connection_serv;
              S_release_struct           s_release_serv;
              S_U_abort_struct           s_u_abort_serv;
              S_P_abort_struct           s_p_abort_serv;
              S_data_struct              s_data_serv;
              } prim_ses; /* 83 bytes maximo */
         Dummy                           dummy_para_apresentacao[254];
                                         /* dummy B */
        } Session_struct;
```

```
        /*-----------------------
           Estrutura das SPDU's
        -------------------------*/

/*---------------------------------------
  Estrutura das SPDU's CONNECT e ACCEPT
---------------------------------------*/

typedef struct
        {
        Session_con_id          session_connection_ident;
        Boolean                 protocol_options_valid;
        uint8                   protocol_options;
        Boolean                 tsdu_max_size_valid;
        unsigned short          tsdu_max_size_to_responder;
        unsigned short          tsdu_max_size_from_responder;
        Boolean                 version_number_valid;
        uint8                   version_number;
        Boolean                 init_serial_number_valid;
        uint32                  init_serial_number;
        Boolean                 token_setting_item_valid;
        uint8                   token_setting_item;
        Boolean                 session_user_requirements_valid;
        uint16                  session_user_requirements;
        S_address               calling_session_selector;
        S_address               called_session_selector;
        Boolean                 token_item_valid;
        uint8                   token_item; /* so no ACCEPT SPDU */
        } CN_AC_SPDU_struct; /* estabelece o comprimento de  */
                            /* 45 bytes que sera o maximo   */
                            /*  comprimento                 */


/*---------------------------
  Estrutura da SPDU REFUSE
---------------------------*/

typedef struct
        {
        Session_con_id          session_connection_ident;
        uint8                   transport_disconnect;
        Boolean                 session_user_requirements_valid;
        uint16                  session_user_requirements;
        Boolean                 version_number_valid;
        uint8                   version_number;
        Boolean                 reason_code_valid;
        S_con_result            reason_code;
        } RF_SPDU_struct; /* 29 bytes */

/*---------------------------
  Estrutura da SPDU FINISH
---------------------------*/

typedef struct
        {
```

```
        uint8           transport_disconnect;
        } FN_SPDU_struct;

/*-------------------------------
  Estrutura da SPDU DISCONNECT
-------------------------------*/

/* A SPDU DISCONNECT nao tem parametros */

/*--------------------------
  Estrutura da SPDU ABORT
--------------------------*/

typedef struct
        {
        uint8           transport_disconnect;
        uint8           origem_aborto;
        } AB_SPDU_struct;

/*----------------------------------
  Estrutura da SPDU DATA TRANSFER
----------------------------------*/

typedef struct
        {
        Boolean         enclosure_item_valid;
        uint8           enclosure_item;
        } DT_SPDU_struct;

/*--------------------------------
  Estrutura da SPDU ABORT ACCEPT
--------------------------------*/

/* A SPDU ABORT ACCEPT nao tem parametros */
```

# ANEXO C : ESTRUTURA DE DADOS DOS PARÂMETROS DAS PRIMITIVAS PARA A CAMADA DE TRANSPORTE.

```
/*-------------------------------------------------------------------
          MENSAGENS NAS INTERFACES DO SISTEMA DIDATICO
 -------------------------------------------------------------------*/

/* OBS. : Todos os tipos declarados como apontador para "char"
 *        terao um tamanho limitado pelo Sistema Didatico
 */
/*-------------- INTERFACES DE PROTOCOLO ----------------------*/

/*****************************************************************
        TIPOS   ESTRUTURADOS
 *****************************************************************/

/*-----------------------------------------
         Estrutura do servico "T_CONNECTION"
  ----------------------------------------*/

/* Obs : Os campos dummy reservados para parametros que so
 * existem nas primitivas indication/confirm
 */

/*-----------------------------------------
  Primitiva T_CONNECTION.request/indication
  ----------------------------------------*/

typedef struct
        {
        T_address       called_tsap_address;
        T_address       calling_tsap_address;
        Boolean         expedited_data_options;
        uint8           quality_of_service;
        } T_connect_ri_struct;

/*-----------------------------------------
  Primitiva T_CONNECTION.response/confirm
  ----------------------------------------*/

typedef struct
        {
        T_address       responding_tsap_address;
        Boolean         expedited_data_options;
        uint8           quality_of_service;
        } T_connect_rc_struct;

/*-----------------------------------------------------
  Estrutura do servico de Transporte : T_CONNECTION
  ----------------------------------------------------*/

typedef union
        {
        T_connect_ri_struct     t_connection_req;
        T_connect_ri_struct     t_connection_ind;
```

```
        T_connect_rc_struct     t_connection_resp;
        T_connect_rc_struct     t_connection_conf;
        } T_connection_struct;
/*-----------------------------------------
         Estrutura do servico "T_DATA"
  -----------------------------------------*/

/*-------------------------------------
  Primitiva T_DATA.request/indication
  -------------------------------------*/

typedef struct
        {
        Boolean         valid_pdu; /* so para indication */
        Boolean         user_data_valid;
        int             codigo_tipo_SPDU;
           union
                { /* As SPDU's DN e AA nao tem parametros */
                CN_AC_SPDU_struct       cn_spdu;
                CN_AC_SPDU_struct       ac_spdu;
                RF_SPDU_struct          rf_spdu;
                FN_SPDU_struct          fn_spdu;
                AB_SPDU_struct          ab_spdu;
                DT_SPDU_struct          dt_spdu;
                }user_data;
        } T_data_ri_struct; /* comprimento maximo: 49 bytes */


/*-----------------------------------------------
  Estrutura do servico de Transporte : "T_DATA"
  -----------------------------------------------*/

typedef union
        {
        T_data_ri_struct        t_data_req;
        T_data_ri_struct        t_data_ind;
        } T_data_struct;

/*-------------------------------------
  Estrutura do servico "T_DISCONNECT"
-------------------------------------*/
/*-------------------------------------------
  Primitiva  T_DISCONNECT.request/indication
-------------------------------------------*/

typedef struct
        {
        uint8           disconnect_reason; /* mandatorio so no
                                              indication */
        }T_disconnect_ri_struct;

/*-------------------------------------------------
  Estrutura do servico de Transporte: T_DISCONNECT
-------------------------------------------------*/

typedef union
```

```
        {
        T_disconnect_ri_struct  t_disconnect_req;
        T_disconnect_ri_struct  t_disconnect_ind;
        }T_disconnect_struct;


/*-------------------------------------------------------
  Estrutura de Servicos de Transporte nas Interfaces N5/N4
  -------------------------------------------------------*/

typedef struct
        {
         Transport_service                  service_name;
         union
              {
              T_connection_struct           t_connection_serv;
              T_disconnect_struct           t_disconnect_serv;
              T_data_struct                 t_data_serv;
              } prim_transp;
         Dummy                              dummy[398];
        } Transport_struct;
```