ESPECIFICAÇÃO DOS SERVIÇOS DA CAMADA SESSÃO DO MODELO OSI NA LINGUAGEM ESTELLE

Maria Fernanda Rodrigues Vaz
ESCA, Engenharia de Sistemas de Controle e Automação

Stefania Stiubiener
Departamento de Engenharia de Eletricidade, Escola Politécnica da Universidade de São Paulo

RESUMO

Na área de sistemas distribuídos faz-se necessário o uso de abordagens formais e sistemáticas que possibilitem visualizar de modo amplo e claro as consequências da solução proposta, facilitem a padronização e permitam a automatização de algumas fases do projeto.

Em 1981, a ISO criou um grupo de estudo de Técnicas de Descrição Formal com o objetivo de padronizar o modo de descrição dos protocolos. Desse grupo surgiu a Estelle ("Extended Transition Language") que é uma linguagem de especificação formal baseada em extensões do conceito de Máquina de Estados Finita e na linguagem Pascal.

O presente trabalho se propõe a apresentar a linguagem Estelle, sua arquitetura, sintaxe e utilização através de um exemplo. Foi escolhida como exemplo a especificação dos serviços oferecidos pela camada sessão do modelo OSI da ISO, exemplo este que integra uma atividade completa de especificação da camada de sessão desenvolvida no âmbito de pós-graduação da Escola Politécnica da USP.

Inicialmente será apresentada a máquina de estados finitos extendida na qual se baseia Estelle, a arquitetura de uma especificação em Estelle, a sintaxe e finalmente o exemplo mencionado.

1. Introdução

Uma Máquina de Estados Finita (MEF) é uma 5-Tupla (X, I, O, N, M) onde [5]:

X - é um conjunto finito de estados

I - é um conjunto finito de entradas (eventos externos)

O - é um conjunto finito de saídas

N - é uma função de transição de estados (N : I x X --> X)

M - é uma função de saída e de ações executadas (M : X x I --> O)

M e N expressam o comportamento da máquina. As transições são definidas pela função de transição de estados e a ocorrência de ações e saídas para o exterior da máquina são definidas pela função de saída e de ações executadas.

A especificação completa de um protocolo complexo exige um número tão grande de estados que torna impraticável o uso de um modelo simples. Tendo isso em vista, optou-se por criar extensões que facilitem a utilização de uma máquina de estados finita para especificações de protocolos de modo a previnir a explosão de estados.

Associando a uma MEF algumas extensões chega-se à Máquina de Estados Finita Extendida (MEFE). As extensões são:

1 - um estado passa a ser composto por um contexto adicional de variáveis e processamento. Na Máquina normal pode-se associar à cada estado apenas uma variável (variável de estado) e na Máquina extendida pode-se associar um conjunto de variáveis de contexto à variável de estado. A variável de estado da máquina extendida passa a ser chamada "variável de estado de controle".

   De modo simples, pode-se dizer que um estado da máquina extendida é um conjunto de estados da máquina normal.

2 - o evento de entrada (evento externo) numa máquina normal é um evento simples ao qual a função de transição de estados associa ações determinísticas, isto é, um mesmo evento provoca a mesma ação. Na máquina extendida o evento de entrada pode provocar ações diferentes dependendo dos seus parâmetros. De modo simples, pode-se dizer que um evento de entrada na máquina extendida é composto por um conjunto de eventos da máquina normal.

3 - Na máquina de estados finita extendida existem regras de

prioridade de ocorrência do disparo das transições. Essas regras foram criadas para resolver os possíveis conflitos gerados pelo agrupamento de vários estados em um único estado e de vários eventos de entrada em um único evento com parâmetros. Esses conflitos levam à possibilidade de ocorrência de várias transições prontas para disparo.

4 - Na máquina de estados finita extendida existe a possibilidade do disparo das transições, após a verificação de suas condições de disparo, ser atrasado de a um determinado período de tempo fixo não nulo (cláusula delay).

2. O Modelo

Uma especificação em Estelle descreve um sistema estruturado hierarquicamente em componentes sequenciais cuja ordem de execução é não determinística, definindo os vários componentes do sistema (módulos) intercambiando mensagens (interações) por meio de enlaces bidirecionais (canais) entre dois pontos de interação [8].

Os módulos podem ser refinados hierarquicamente criando "instâncias filhas". Tanto a criação como a interconexão dos módulos são dinâmicas, isto é, podem ser alteradas no tempo de acordo com as regras de criação e interconexão dos módulos [8].

Os módulos são especificados como máquinas de estado finita extendida usando as facilidades regidas por um conjunto de extensões da linguagem Pascal da ISO [8].

2.1 O Módulo

Em Estelle define-se um módulo por um cabeçalho e por um corpo. Um cabeçalho pode fazer referência a mais de um corpo, assim, em uma dada situação o cabeçalho é associado a um corpo e em outra situação a outro corpo diferente. A relação cabeçalho-corpo é um para vários. Esta facilidade de mapeamento é muito útil, apesar de parecer um tanto incomum, como por exemplo, no caso do protocolo de transporte da ISO [25] com cinco classes; pode-se especificar um corpo para cada classe, e em determinada situação associar ao cabeçalho o corpo da classe selecionada [8]. Um outro exemplo é o caso do protocolo de sessão da ISO [16]. Este protocolo possui um conjunto de unidades funcionais. Na fase de

estabelecimento de uma conexão de sessão é feito um acordo sobre as unidades usadas nas fases de transferência de dados e finalização da conexão. Por isso podem existir vários corpos e após o acordo realizado associa-se ao cabeçalho do módulo o corpo correspondente ao acordo.

O módulo realiza acessos ao ambiente externo por meio dos seus pontos de interação ou suas "variáveis exportadas". A cada ponto de interação está associada uma fila do tipo FIFO (first-in first-out). Uma mesma fila pode ser associada a vários pontos de interação de um mesmo módulo [8].

Tanto os pontos de interação como as variáveis exportadas são declaradas no cabeçalho do módulo. Apenas a instância pai pode realizar acessos externos às variáveis exportadas pelos filhos. Variável exportada é uma maneira de permitir que os módulos compartilhem variáveis, um deles exporta a variável e outro pode acessá-la [8].

O comportamento interno de um módulo pode ser caracterizado em termos de um sistema de transição de estados não determinísticos, ou seja:

- pelo conjunto de estados;
- pelo subconjunto de estados iniciais;
- relação do próximo estado (transição).

O estado de um módulo caracteriza-se pela sua variável de estado de controle e por um conjunto de variáveis de contexto. Associa-se a variável de estado de controle à palavra reservada "state". As variáveis de contexto influenciam na determinação do estado via verificação de seus valores, de acordo com uma função pré-determinada.

Quando se faz referência à variável de estado de controle (state) utiliza-se a expressão "estado de controle" que representa apenas a parte controle do estado.

O estado inicial de um módulo é definido na parte de iniciação do módulo. Esta parte do corpo do módulo descreve [8]:

- estado de controle inicial
- o modo iniciação das variáveis de contexto
- hierarquia e estrutura de interconexão iniciais das instâncias-filhas do módulo, se estas existirem.

A relação próximo estado também é chamada transição de estado. Uma transição é composta sintaticamente de duas partes [8]:

- cabeçalho;
- bloco de transição.

O cabeçalho da transição compõe-se de cláusulas que determinam:

- condições de disparo à partir do estado atual de controle;
- próximo estado de controle.

Cada cláusula tem uma função diferente. A cláusula "from" indica o estado de controle origem, a cláusula "to" indica o próximo estado de controle, a cláusula "priority" indica a prioridade de disparo, a cláusula "provided" permite a utilização de um predicado composto de vários elementos e faz parte das condições de disparo. Existem outras cláusulas que serão descritas posteriormente.

O bloco de transição descreve as ações ou operações executadas quando a transição é disparada.

O corpo associado ao cabeçalho do módulo, é formado por três partes: Iniciação, Transição e Finalização, podendo ser refinado em submódulos cuja criação pode ocorrer na parte Iniciação ou na parte Transição do módulo. A estrutura é chamada estática quando a criação ocorre na parte Iniciação e é dita dinâmica quando a criação ocorre na parte Transição [8].

Quanto à sincronização classificam-se os módulos em [8]:

- fortemente acoplados:
  quando eles compartilham variáveis comuns (variáveis exportadas)
- fracamente acoplados:
  quando eles não compartilham variáveis comuns. A sincronização ocorre via as interações trocadas através dos pontos de interação.

2.2 Regras de Criação de Módulos

Os módulos podem ser classificados em:

- módulos sem atributo;
- módulos com atributo, ou atribuidos.

O atributo representa a classe do módulo e pode ser:

- "process" (processo);
- "activity" (atividade);
- "systemprocess" (sistema-processo);
- "systemactivity" (sistema-atividade).

Pode-se dizer que os módulos sistema-processo e sistema-

atividade são hierarquicamente superiores aos módulos processo e atividade. Os módulos sistema-processo e processo permitem a execução paralela com outros módulos e os módulos sistema-atividade e atividade permitem a execução sequencial de atividades.

Define-se um módulo como ativo quando sua parte Transição é não nula, isto é, existe pelo menos uma transição que pode ser ativada, em uma dada condição. Define-se um módulo como inativo quando sua parte Transição é nula [8].

O módulo denominado "specification" é o módulo hierarquicamente superior a todos os outros podendo ser atribuida ou não. Quando atribuído aceita apenas os atributos "systemprocess" ou "systemactivity".

As regras de criação de um módulo são [8]:

1 - Todo módulo ativo tem que possuir um atributo
2 - Um módulo sistema ("systemprocess" e "systemactivity") não pode ser criado dentro de um módulo com atributo. Consequentemente o seu módulo hierarquicamente superior tem que ser inativo e não atribuído.
3 - Os módulos com atributo "process" ou "activity" têm que ser criados dentro de um módulo sistema.
4 - Um módulo "systemprocess" ou "process" pode ser estruturado apenas em módulos "process" ou "activity".
5 - Um módulo "activity" ou "systemactivity" pode ser estruturado apenas em módulos "activity".

O módulo hierarquicamente superior (módulo pai) age como supervisor de seus filhos. As transições do módulo pai são prioritárias em relação às dos módulos filhos, isto, para evitar conflitos. Este é o princípio da prioridade hierárquica [8].

Das regras de criação de módulos e do princípio da prioridade hierárquica pode-se concluir que: os módulos sistema não são supervisionados, pois, o seu superior é inativo e assim não há conflito de transições. Além disso, os módulos sistema são assíncronos entre si. A diferença entre "systemprocess" e "systemactivity" encontra-se na estrutura interna do módulo com esses atributos.

2.3 Interconexão de módulos

A estrutura de um sistema é definida por um conjunto de módulos e pelo sistema de interconexão que os interliga. Esta é a arquitetura usada pela linguagem Estelle.

Um ponto de interação é uma interface abstrata associada a um conjunto de regras locais e a uma fila que armazena as interações recebidas de outros módulos.

Os pontos de interação possuem os seguinte atributos [15]:

- tipo de fila: atributo que define se a fila é individual ou compartilhada com mais algum ponto de interação do módulo. Todas as filas são do tipo FIFO.
- papel: atributo que define o papel representado. Existem dois tipos de papéis, sempre opostos. Por exemplo: "send" e "receive"; "user", "provider"; entrada e saída, etc.
- grupos: atributo que define uma coleção de interfaces com regras idênticas. Os grupos de pontos de interação são tratados como uma matriz e podem ser indexados como tal para identificar os elementos individuais do grupo.
- tipo de canal: atributo que define o identificador associado ao tipo de canal. Associando-se o ponto de interação a uma canal associa-se também o conjunto de interações deste canal.

O ponto de interação pode ainda ser externo ou interno. É externo quando definido no cabeçalho do módulo e interno quando definido na parte declaração das variáveis do corpo do módulo. Os pontos de interação externos são visíveis ao ambiente externo do módulo em questão.

Especifica-se um canal em duas partes. A primeira diz respeito aos papéis representados pelos pontos de interação, que este interliga. E a segunda às possíveis interações permitidas no canal, inclusive a ordem de ocorrência dessas interações [15].

Um canal liga um ponto de interação de um módulo a um ponto de interação de outro módulo, ou do mesmo módulo. Um desses pontos de interação representa o transmissor de interações e o outro ponto de interação representa o receptor de interações [15].

2.4 Operações de ligação dos pontos de interações (connect, disconnect, attach, detach)

Os pontos de interação podem ser internos ou externos. Os pontos externos dos módulos filhos podem ser ligados apenas pelos seus pais, e da mesma maneira, só os módulos superiores podem liberar a ligação entre dois pontos externos de seus filhos. Os pontos internos são ligados dentro do próprio módulo.

Um módulo pode ligar através da operação connect [8]:

- um de seus pontos de interação internos a um ponto de interação externo de um de seus filhos;
- dois de seus pontos de interação internos;
- dois pontos de interação externos de seus filhos.

A operação connect une pontos de interação representantes de papéis opostos (ação recíproca). A operação inversa é a disconnect. Esta operação tem que ser realizada pelo mesmo módulo que realizou a operação connect.

Quando executa-se a operação disconnect, a fila de recepção do ponto de interação permanece inalterada, isto significa que, as interações podem ser usadas pelo módulo receptor mesmo após a liberação da conexão. No caso do módulo transmissor, após a liberação da conexão, descarta-se a interação destinada à fila do ponto de interação desconectado. As filas do módulo são destruidas apenas na liberação dos módulos [8].

Existe um outro tipo de ligação entre os pontos de interação. É a operação attach, que liga pontos de interação externos dos pais e pontos de interação externos dos filhos, ambos os pontos representando o mesmo papel. Nesta operação ocorre o seguinte:

- o módulo A ligou-se, via operação connect, a um dos pontos de interação externos do seu filho Ai
- o módulo Ai liga esse ponto de interação externo, via operação attach, a um dos pontos de interação externos de um de seus filhos.

Observa-se que a operação attach não é simples exigindo anteriormente a execução de uma operação connect. Colocando-se várias interações na fila do ponto de interação externo do módulo que realiza operação attach, estas interações estarão imediatamente disponíveis para o filho após da operação.

A operação detach é inversa à operação attach. Em primeiro

lugar, as interações que estão na fila do módulo hierarquicamente inferior são colocadas à disposição do módulo que executa a operação detach, e só então, os pontos de interação são liberados.

Existe ainda uma outra característica: as interações podem ser enfileiradas para um módulo que esteja muitos níveis hierárquicos abaixo, por meio de várias operações attach.

Considere-se como exemplo o sistema formado por quatro módulos (W, X, Y e Z) com os seguintes pontos de interação (fig. 1 e 2) :

Módulo	Ponto de interação	Hierarquia	Fila Associada
W	"a" interno	pai de X	fila_a
X	"b" externo	filho de W	fila_b
Y	"c" externo	filho de X	fila_c
Z	"d" externo	filho de Y	fila_d

E as seguintes operações executadas nessa ordem:

Assertiva	Módulo de execução
connect a.u to X.b.p;	W
attach b.p to Y.c.p;	X
attach c.p to Z.d.p;	Y

onde u e p são os papéis desempenhados pelos pontos de interação.

u - usuário , p - provedor.

Após a execução dessas operações os pontos de interação "a","b" e "c" estão ligados ao ponto "d". Qualquer interação colocada nas filas de "a", "b" ou "c" são transferidas para a fila_d e deixam de ser visíveis aos pontos "a", "b" e "c".

Se a seguir for executada, pelo módulo X, a assertiva:
detach b; ou a assertiva
detach Y.c;

Os pontos "b" e "c" são desligados. As interações na fila_d endereçadas ao ponto "b" (e não iniciadas através de "c") são retiradas da fila_d e colocadas na fila_b (módulo X) com a ordem de colocação preservada.

3. Especificação dos Serviços da Camada Sessão em Estelle

De acordo com o Modelo de Referência OSI, a camada Sessão oferece serviços à camada Apresentação e usa os serviços da camada Transporte. Como todas as camadas desse modelo, a camada Sessão possui um conjunto de funções relacionadas aos serviços oferecidos a sua camada superior e um conjunto de regras que definem o

protocolo entre as entidades de sessão [17].

A especificação em Estelle dos serviços da camada Sessão, no contexto deste trabalho, encontra-se definida em Sistema que engloba as camadas Transporte e Apresentação. O refinamento da camada Sessão compõe-se de dois módulos: o módulo Serviço-sessão responsável pelos serviços de sessão e o módulo Protocolo-sessão responsável pelos procedimentos do protocolo de sessão. Das unidades funcionais acordadas durante a fase "estabelecimento da conexão de sessão" e usadas nas fases "transferência de dados" e "finalização da conexão", especifica-se apenas o Subconjunto Combinado Básico composto pelas unidades: Núcleo, Semi-Duplex e Liberação Negociada [19].

A especificação do Sistema descrito não detalha os corpos dos módulos Apresentação, Transporte e Protocolo-Sessão. Os pontos de interação externos a esses módulos e os canais de interconexão são definidos para fornecer uma visão ampla do contexto do módulo Serviço-Sessão e seu refinamento (fig. 3.a e 3.b).

Devido à falta de espaço não serão descritos detalhadamente os serviços de sessão especificados a seguir. Sugere-se ao leitor consultar a bibliografia correspondente [20].

3.1 Especificação do Sistema

```
specification sistema systemprocess; timescale segundos;
   {A especificação do sistema tem atributo "systemprocess" e os
   seus filhos (Apresentação, Sessão e Transporte) têm atributo
   "process", a escala de tempo é segundos}
const
  {definição das constantes usadas nesta especificação}
type
  {definição dos tipos usados nesta especificação}
  {as interações dos canais TS-ponto-acesso e SS-ponto-acesso
    usadas nesta especificação não estão detalhadas}
  channel TS-ponto-acesso (TS-usuário, TS-provedor);
  channel SS-ponto-acesso (SS-usuário, SS-provedor);
{Definição dos cabecalhos dos módulos}
Module Apresentação-tipo process
  (A-CEP : Cep-tipo);   {parâmetro identificador da instância}
     ip AS : SS-ponto-acesso (SS-usuário) individual queue;
```

```
end;
(O módulo Apresentação-tipo possui um ponto de interação AS que
  representa o papel usuário. Esse ponto de interação não
  compartilha a fila com outros pontos de interação)
Module Transporte-tipo process
  (T-CEP : Cep-tipo);  (parâmetro identificador da instância)
     ip ST : TS-ponto-acesso (TS-provedor) individual queue;
(O módulo Transporte-tipo possui um ponto de interação ST que
  representa o papel provedor. Esse ponto não compartilha a fila
  com outros pontos de interação)
Module Sessão-tipo process
  S-CEP : Cep-tipo);     parâmetro identificador da instância)
     ip AS : SS-ponto-acesso (SS-provedor) common queue;
        ST : TS-ponto-acesso (TS-usuário) common queue;
(O Módulo Sessão-tipo possui dois pontos de interação: AS que
  representa o papel provedor e TS que representa o papel usuário)
(Definição dos corpos dos módulos)
body Apresentação-corpo for Apresentação-tipo : external;
(O corpo do módulo Apresentação é externo)
body Transporte-corpo for Transporte-tipo : external;
(O corpo do módulo Transporte é externo)
body Sessão-corpo for Sessão-tipo;
(O corpo do módulo sessão encontra-se especificado no item a
  seguir)
(Declaração das variáveis do módulo)
modvar
  Instância-Apresentação : array [cep-tipo] of Apresentação-tipo;
  Instância-Transporte : array [cep-tipo] of Transporte-tipo;
  Instância-Sessão : array [cep-tipo] of Sessão-tipo;
(Parte Iniciação da especificação sistema)
initialize
  begin
     all cep : cep-tipo do
     begin
        (inicia as instâncias do módulos)
        init  Instância-Apresentação [cep] with  apresentação-tipo
                    (cep);
        init   Instância-Transporte  [cep]  with   Transporte-tipo
                    (cep);
```

```
        init Instância-Sessão [cep] with Sessão-tipo (cep);
        (liga os pontos de interação por meio de canais)
        connect  Instância-Apresentação [cep] .  AS to  Instância-
                        Sessão [cep] . AS;
        connect   Instâncias-Sessão  [cep]  .   ST  to  Instância-
                        Transporte [cep] . ST;
    end;
  end;                  (fim da iniciação)
end;                    (fim da especificação)
```

3.2 Corpo Associado ao Módulo Sessão

```
body Sessão-corpo for Sessão-tipo;
(Definição do canal SPS-ponto-acesso)
channel SPS-ponto-acesso (SPS-usuário, SPS-provedor);
  by SPS-usuário:
  by SPS-provedor:
  (As  interações ou mensagens possíveis nos pontos  de  interação
    que definem o canal são as mesmas)
        S-CONNECT-request (SCONREQ : S-conexão-req-ind);
        S-CONNECT-response (SCONRESP : S-conexão-resp-conf);
        S-RELEASE-request (SRELREQ : S-liberação-req-ind);
        S-RELEASE-response (SRELRESP : S-liberação-resp-conf);
        S-DATA-request (SDTREQ : S-transf-dado);
        etc.
(Definição dos cabeçalhos dos módulos criados pelo módulo sessão)
module Serviço-Sessão-tipo process;
  ip
     SSS : SS-ponto-acesso (SS-provedor) common queue;
     SSPS : SPS-ponto-acesso (SPS-usuário) common queue;
end;
(O  módulo  Serviço-Sessão  possui dois pontos  de  interação  que
  compartilham a mesma fila)
module Protocolo-Sessão-Tipo process;
  ip
     SSPS : SPS-ponto-acesso (SPS-provedor) common queue;
     PST : TS-ponto-acesso (TS-usuário) common queue;
end;
(O  módulo  Protocolo-Sessão possui dois pontos de  interação  que
```

```
  compartilham a mesma fila)
(Declaração das variáveis do módulo)
modvar
  instância-serviço : Serviço-Sessão-tipo;
  instância-protocolo : Protocolo-Sessão-tipo;
(Declaração dos corpos associáveis aos cabecalhos dos módulos
  criados)
body Serviço-Sessão-corpo for Serviço-Sessão-tipo;
  (O corpo do módulo Serviço-Sessão encontra-se especificado no
    item seguinte)
body Protocolo-Sessão-corpo for Protocolo-Sessão-tipo : external;
  (O corpo do módulo Protocolo-Sessão não será detalhado neste
    trabalho)
(Parte iniciação do módulo Sessão)
initialize
  begin
     (inicia os módulos)
     init instância-serviço with Serviço-Sessão-tipo;
     init instância-protocolo with Protocolo-Sessão-tipo;
     connect instância-serviço . SSPS to instância-protocolo .
                        SSPS;
     attach Instância-Sessão . AS to instância-serviço . SSS;
     attach Instância-Sessão . ST to instância-protocolo . PST;
  end;                  (fim da iniciação)
end;                    (fim do corpo do módulo Sessão)
```

3.2.3 Corpo Associado ao Módulo Serviço-Sessão

```
body Serviço-Sessão-corpo for Serviço-Sessão-tipo;
(Definição do cabecalho do módulo transfere-finaliza criado pelo
  módulo Serviço-Sessão. Possui atributo "activity" para que
  apenas uma instância seja executada por vez)
module transfere-finaliza-tipo activity;
  ip
     SSTF : SS-ponto-acesso (SS-provedor) common queue;
     TFPS: SPS-ponto-acesso (SPS-usuário) common queue;
  export
     fim-transferência : boolean; (indica fim da transferência de
                        dados e da conexão)
```

```
end;
(O  módulo transfere-finaliza possui dois pontos de interação  que
  compartilham a mesma fila. Possui também uma  variável exportada
  que pode ser compartilhada pelo módulo Serviço-Sessão)
(Definição do corpo associado ao módulo transfere-finaliza)
body BCHS-corpo for transfere-finaliza-tipo: external;
  (O corpo BCHS não será detalhado neste trabalho)
(Declaração da variável do módulo)
modvar
  instância-transfere-finaliza : transfere-finaliza-tipo;
state
  STA01,                  (não conectado)
  STA02,                  (espera S-CONNECT-confirm)
  STA08,                  (espera-S-CONNECT-response)
  STA713;                 (conectado)
(Declaração dos procedimentos)
procedure conexão-aceita (num-serial : número-serial):
  (este  procedimento  inicia as variáveis relacionadas ao  número
   serial-(para   as   unidades   funcionais   de   sincronização   -
   UFSS,UFSP,UFRS)
procedure ini-controle-ficha;
(Este  procedimento inicia o controle das fichas de acordo  com os
  parâmetros de entrada)

procedure inicia-tf (req : subconjunto-funcional)
(Esta  rotina associa o corpo BCHS ao módulo transfere-finaliza de
  acordo com o parâmetro req)
begin
  init instância-transfere-finaliza with BCHS-corpo;
  (liga os pontos de interação)
  attach  instância-serviço .  SSS to instância-transfere-finaliza
                        . SSTF;
  attach instância-serviço .  SSPS to instância-transfere-finaliza
                        . TFPS;
end;

procedure finaliza-transfere;
(Esta  rotina  libera  o  corpo  associado  ao  módulo  transfere-
  finaliza)
```

```
begin
  (desliga os pontos de interação)
  detach instância-serviço . SSS;
  detach instância-serviço . SSPS;
  (libera o módulo)
  release instância-transfere-finaliza;
end;
(Parte iniciação do módulo Serviço-Sessão)
initialize
  to STA01                (estado desconectado)
(parte transição do módulo Serviço-Sessão)
trans
  from STA01              (não conectado)
     to STA2A             (espera confirmação)
        when SSS.S-CONNECT-request (pedido de conexão do usuário)
        begin
           output SSPS . S-CONNECT-request;
        end;
     to STA08             (espera resposta)
        when SSPS .  S-CONNECT-indication (indicação de conexão do
                        provedor)
        begin
           (envia indicação de conexão ao usuário)
           output SSS . S-CONNECT-indication;
        end;

  from STA08             (espera resposta)
     when  SSS  .   S-CONNECT-response  (resposta  de  conexão  do
                        usuário)
        provided (S-CONNECT-response.S-resultado = aceita)
           to STA713    (conectado)
           begin
              (inicia as variáveis da conexão)
              conexão-aceita(S-CONNECT-response.S-num-serial);
              ini-controle-fichas;
              inicia-tf(S-CONNECT-response.S-requerimentos);
              output SSPS.S-CONNECT-response;
           end;
        provided otherwise
```

```
        to STA01      (não conectado)
        begin
           output SSPS.S-CONNECT-response;
        end;
from STA2A            (espera confirmação)
   when   SSPS.S-CONNECT-confirm  (confirmação  da  conexão   do
                      provedor)
      provided(S-CONNECT-confirm.S-resultado = aceita)
        to STA713     (conectado)
        begin
           conexão-aceita(S-CONNECT-confirm.S-num-serial);
           ini-controle-ficha;
           inicia-tf;
           output SSS.S-CONNECT-confirm;
        end;
      provided otherwise
        to STA01      (não conectado)
        begin
           output SSS.S-CONNECT-confirm;
        end;
   provided delay (SMAXCONCOF) ("timeout")
      to STA01        (não conectado)
      begin
         (avisa  usuário  que  estourou  o tempo  de  espera  da
                      confirmação da conexão)
         (este procedimento fica para a implementação)

from STA713           (conectado) (transição espontânea)
   provided instância-transfere-dados.fim-transferência
      to STA01        (não conectado)
      begin
         finaliza-transfere;
      end;
end;                  (fim Serviço-Sessão)
```

CONCLUSÕES

Foi apresentada a utilização da linguagem Estelle e o exemplo de especificação parcial dos servicos da camada sessão do modelo OSI.

Observa-se que o aspecto da verificação da especificação não foi abordado no âmbito deste artigo. Uma atividade neste sentido implicaria na utilização de métodos e ferramentas adequadas de verificação: verificação da sintaxe (Compilador Estelle) e verificação lógica da especificação, através de algoritmos específicos.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BOCHMANN, Gregor V.. A Hybrid Model and the Representation of Comunication Services, 1982. Separata de COMPUTER NETWORK ARCHITECTURE AND PROTOCOLS, edited by Paul E., Green Jr., Plenum Press, NY, 1982, capítulo 23.

[2] BOCHMANN, Gregor V.. Recent developments in protocol specification, validation and testing, 4o. Simpósio Brasileiro de Redes de Computadores, março 1986.

[3] BOCHMANN, Gregor V. & SUNSHINE, Carl A.. Formal Methods in Communications Protocol Design, IEEE Transactions on Communications, vol. 28, no. 4, april 1980.

[4] BOCHMANN, Gregor V., & SUNSHINE, Carl A.. A Survey of Formal Methods, 1982. Separata de COMPUTER NETWORK ARCHITECTURE AND PROTOCOLS, edited by Paul E., Green Jr., Plenum Press, NY, 1982, capítulo 20.

[5] DANTHINE, André A. S.. Protocol Representation with Finite State Models, 1982. Separata de COMPUTER NETWORK ARCHITECTURE AND PROTOCOLS, edited by Paul E., Green Jr., Plenum Press, NY, 1982, capítulo 21.

[6] DAY, John D. & ZIMMERMANN, Hubert. The OSI Reference Model, Proceedings of the IEEE, vol. 71, no. 12, december 1983.

[7] EMMONS, Willard F. & CHANDLER, A.S.. OSI Session Layer: Services and Protocols, Proceedings of the IEEE, vol. 71 no. 12, december 1983.

[8] INFORMATION PROCESSING SYSTEMS - OPEN SYSTEMS INTERCONNECTION - ESTELLE - A FORMAL TECNIQUE BASED ON AN EXTENDED STATE TRANSITION MODEL, document ISO/TC97 SC 21 N 1575, draft proposed ISO/DP 9074, february 1986.

[9] GABOS, Denis & MATTIAS, Hamilton F. & TANAO, Sidney H. & STIUBIENER Stefânia. Estudo e Proposta de Testes de Comportamento do Protocolo de Transporte da ISO, 4o. Simpósio Brasileiro de Redes de Computadores, março 1986.

[10] HAILPERN, Brent T.. Specifying and Verifying Protocols Represented as Abstract Programs, 1982. Separata de COMPUTER NETWORK ARCHITECTURE AND PROTOCOL, edited by Paul E., Green Jr., Plenum Press, NY, 1982, capítulo 22.

[11] HENNIE, Frederick C.. Finite-State Models for Logical Machines, Massachusetts Institute of Techonology, John Wiley & Sons, Inc, New York, London, Sydey, 1968.

[12] SISTEMAS DE PROCESSAMENTO DA INFORMAÇÃO - INTERCONEXÃO DE SISTEMAS ABERTOS - MODELO BÁSICO DE REFERENCIA, Projeto de Norma da ABNT, 1986.

[13] LININGTON, Peter F.. Fundamentals of the Layer Definitions and Protocol Specifications, Proceeding of the IEEE, vol. 71, no. 12, december 1983.

[14] LINN, Jerry & NIGHTINGALE, J. Stephen. Testing OSI Protocols at the National Bureau of Standards, Proceedings of the IEEE, vol. 71, no. 12, december 1983.

[15] LINN, Richard. The features and facilities of Estelle, IFIP Protocol Specification, Testing an Verification V, 1986.

[16] MOURA, José A. B. & SAUVÉ, Jacques P. & GIOZZA, Willian F, & ARAUJO, José F. M.. Redes Locais de Computadores - Protocolos de Alto Nível e Avaliação de Desempenho, editora Mc Graw Hill, 1986, Tecnologia e Aplicações.

[17] INFORMATION PROCESSING SYSTEMS - OPEN SYSTEMS INTERCONNECTION - BASIC REFERENCE MODEL, ref. no. 7898-1984 (E), International Standard 7498, first edition 1984-10-15.

[18] PETERSON, James C.. Petri Nets, Computing Surveys, vol. 9, no. 3, september 1977.

[19] INFORMATION PROCESSING SYSTEMS - OPEN SYSTEMS INTERCONNECTION - BASIC CONNECTION ORIENTED SESSION PROTOCOL SPECIFICATION, document ISO/TC97 SC 16 N 1443, draft proposed ISO/DP 8327, october 1984.

[20] INFORMATION PROCESSING SYSTEMS - OPEN SYSTEMS INTERCONNECTION - BASIC CONNECTION ORIENTED SESSION SERVICE SPECIFICATION, document ISO/ C97 SC 16 N 1442, draft proposed ISO/DP 8326, october 1984.

[21] SOUZA, Wanderley L.. Utilização dos conceitos de Módulo, Porta e Canal em Especificações Formais de Serviços, Protocolos e Interfaces de Comunicação, 3o. Simpósio Brasileiro de Redes de Computadores, 1985.

[22] SOUZA, Wnaderley L. & STIUBIENER Stefânia. Especificação, verificação e testes de protocolos, Relatório Técnico GRC/UFPB, no. TR-01/88, fevereiro 1988.

[23] SUNSHINE, Carl. Formal Techniques for Protocol Specification and Verification, Computer, september 1979.

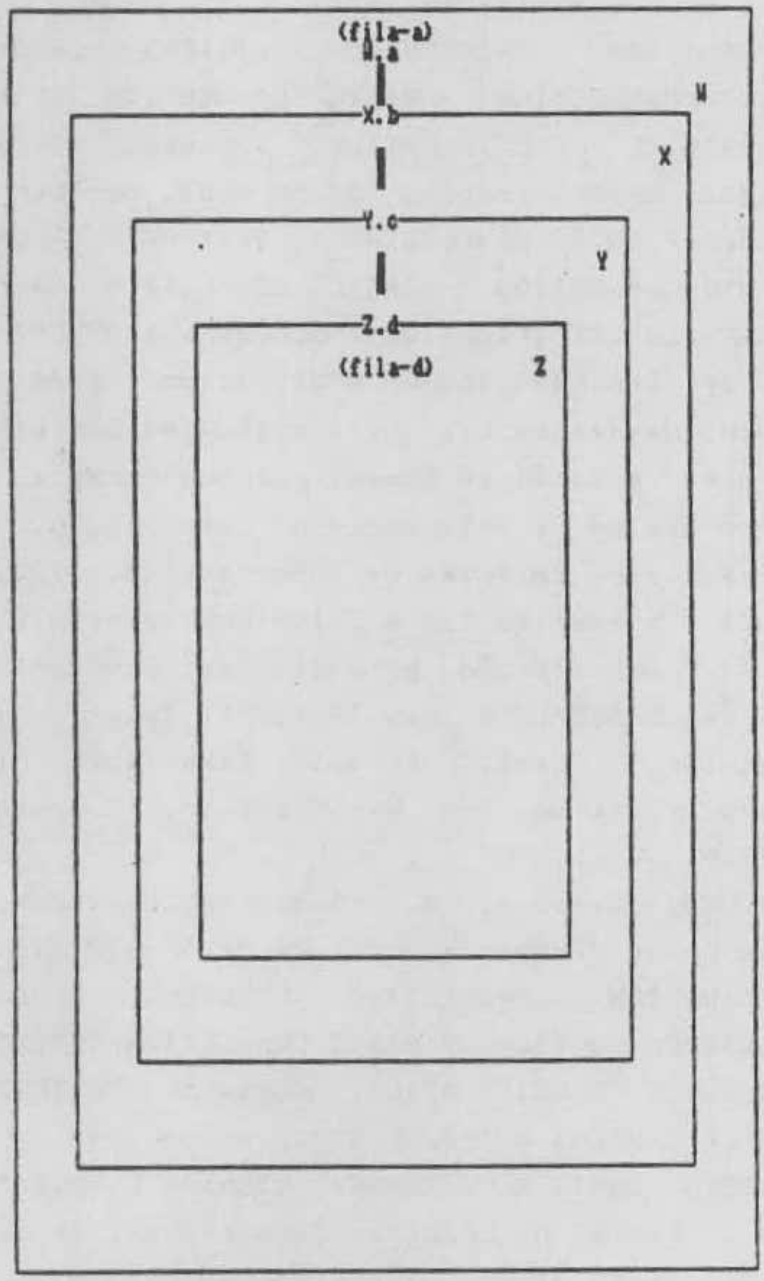
[24] TAROUCO, Liane M. R.. Redes de Computadores - Locais e de Longa Distância, editora Mc Graw Hill, 1986.

[25] INFORMATION PROCESSING SYSTEMS - OPEN SYSTEMS INTERCONNECTION - BASIC CONNECTION ORIENTED TRANSPORT SERVICE SPECIFICATION, document ISO/TC97 S 16 N 1435, draft proposed ISO/DP 8072, march 1983.

[26] VISSERS, Chris A. & TENNEY, Richard L. & BOCHMANN, Gregor V.. Formal Description Techiniques, Proceedings of the IEEE, vol. 71, no. 12, december 1983.

[27] VISSERS, Chris A.. Trends and Proliferation of Formal Description Techniques for OSI Standards, 5o. Simpósio Brasileiro de Redes de Computadores, São Paulo, 13 a 15 de abril de 1987.

[28] WEAVING, Ken. Verification of high-level Protocol Implementations, Computer Communications, vol. 4. 1981.



Legenda:

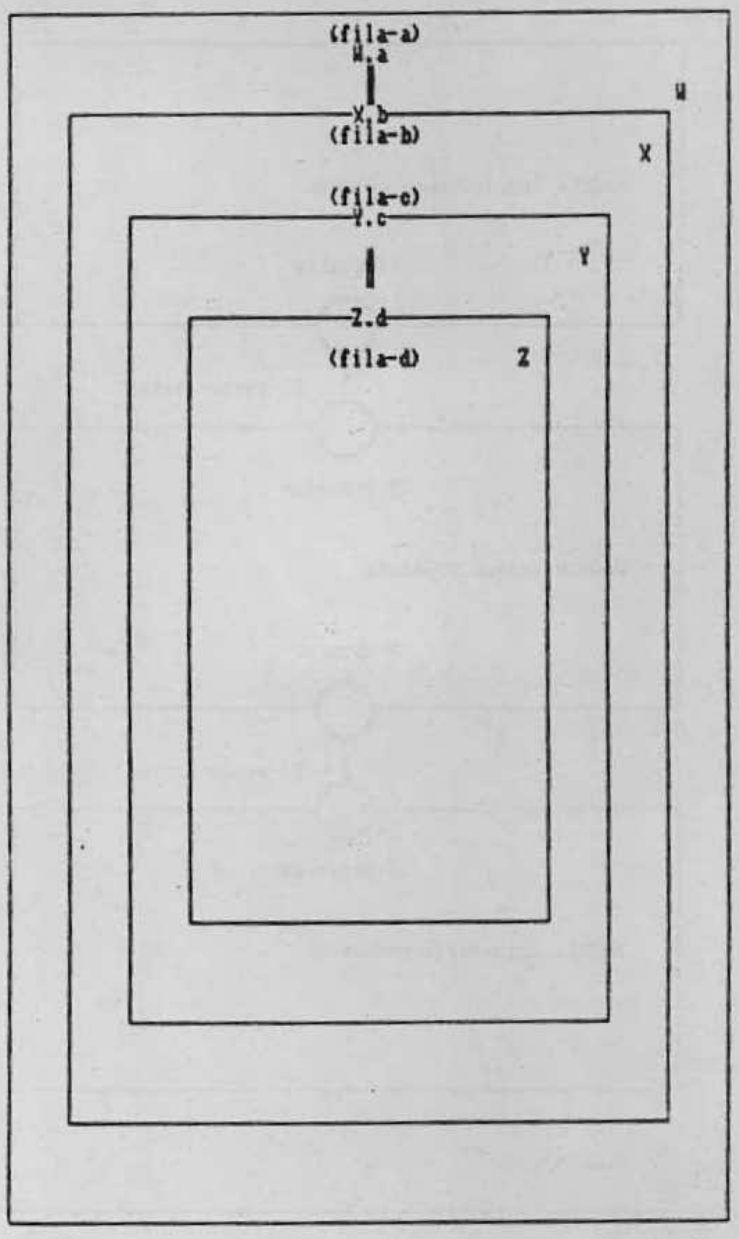
W, X, Y, Z - identificadores dos modulos

a, b, c, d - identificadores dos pontos de interacao

(.) - ponto de interacao

(▯) - ligacao entre pontos de interacao

figura 1



Legenda:

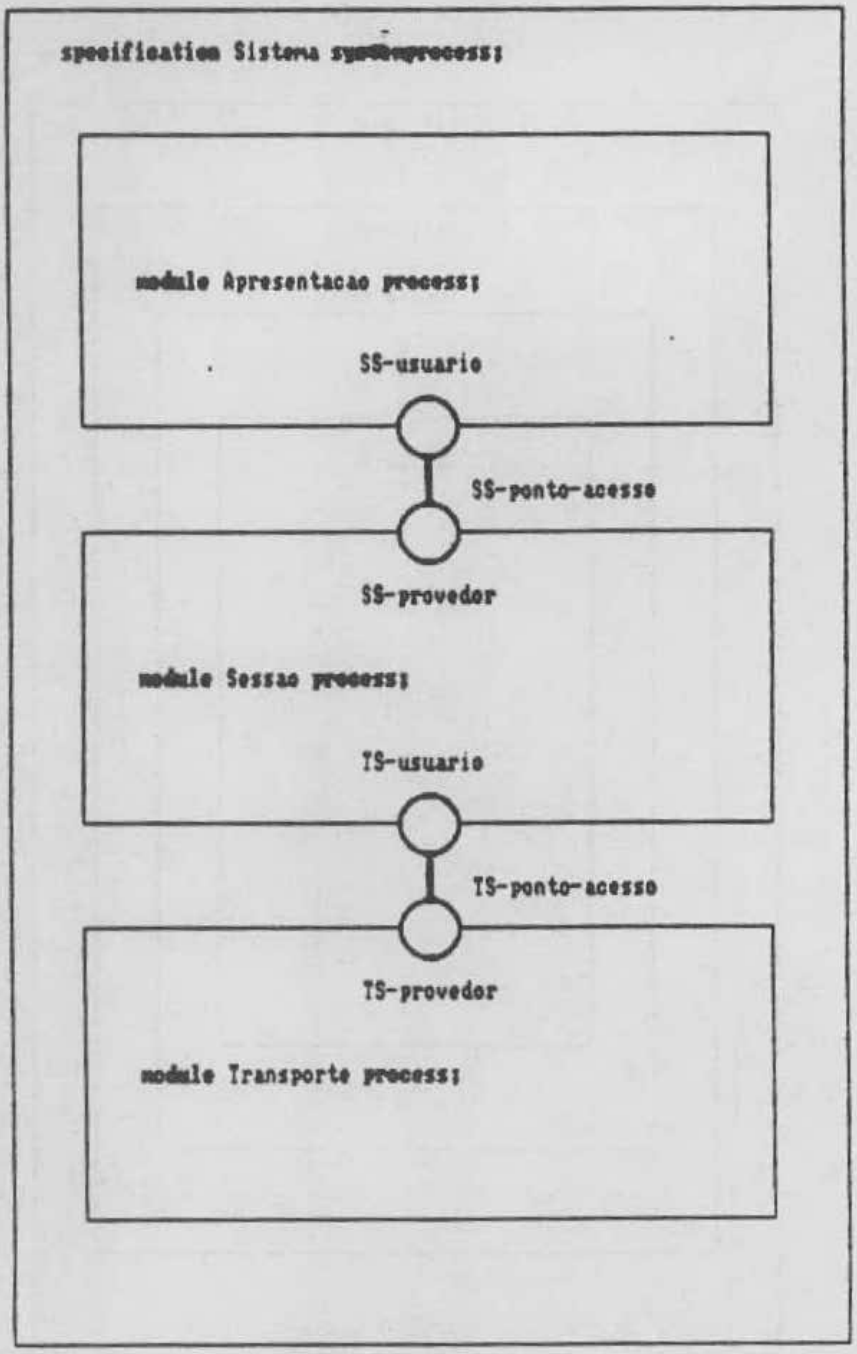
W, X, Y, Z - identificadores dos modulos

a, b, c, d - identificadores dos pontos de interacao

(.) - ponto de interacao

(▮) - ligacao entre pontos de interacao

figura 2



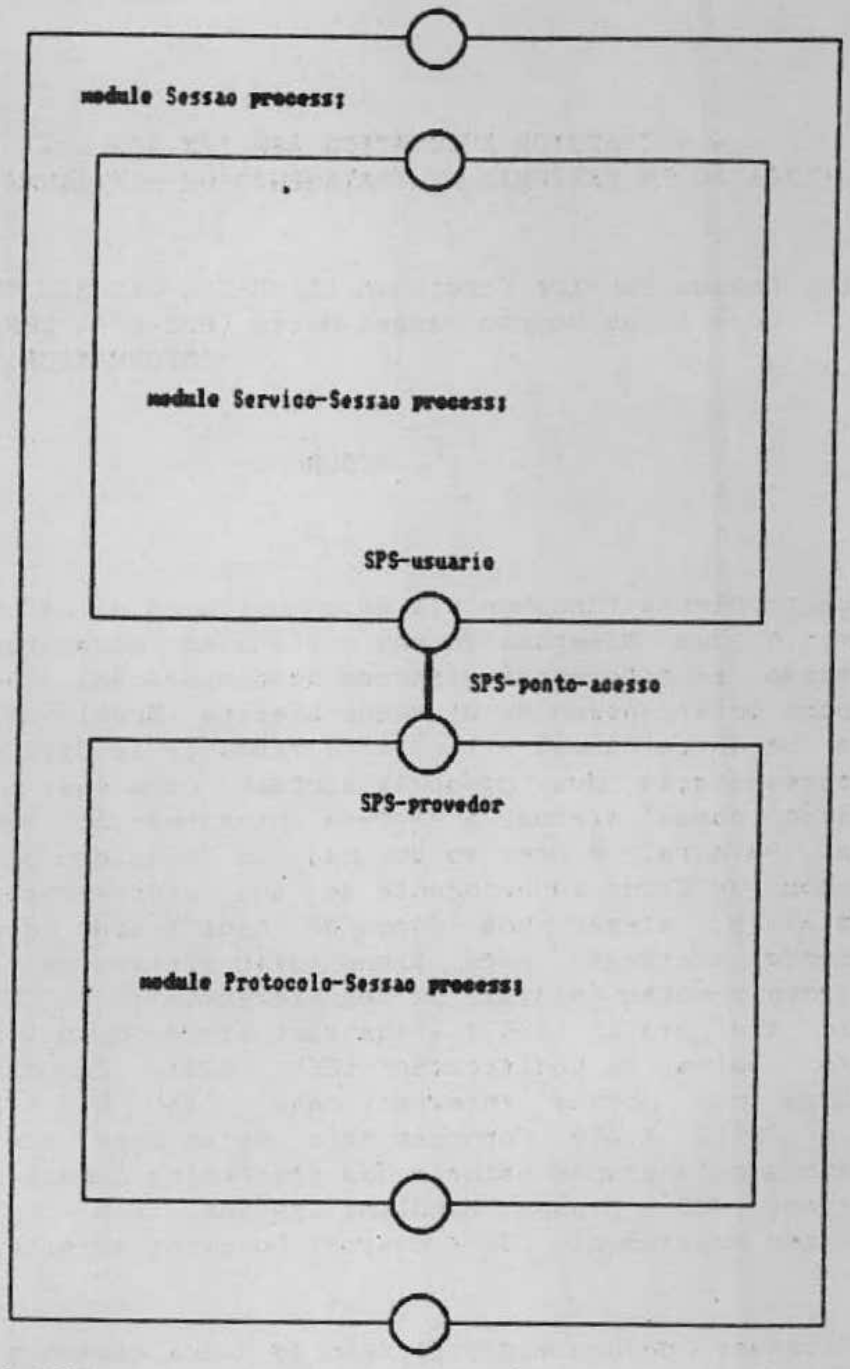
Legenda:

= Modulo

= Canal

= Ponto de Interacao

figura B.a



Legenda:

= Modulo

= Canal

= Ponto de Interacao

Figura 4.a