8º SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES

UM EXEMPLO DE APLICAÇÃO DE TTCN (The Tree Tabular Combined Notation) NA DESCRIÇÃO DE CENÁRIOS DE TESTE DE UMA IMPLEMENTAÇÃO DO PROTOCOLO DE TRANSPORTE

Autores:

José Pedro de Oliveira
Escola Politécnica da Universidade de São Paulo
Centro de Computação Eletrônica da Universidade de São Paulo

Tereza Cristina Melo de Brito Carvalho
Escola Politécnica da Universidade de São Paulo
Fundação para o Desenvolvimento Tecnológico da Engenharia

Stefania Stiubiener
Escola Politécnica da Universidade de São Paulo

Sumário:

Este artigo apresenta a aplicação prática de um método de geração de seqüência de teste para uma máquina de protocolo e a descrição do cenário de teste correspondente utilizando-se a linguagem TTCN (The Tree Tabular Combined Notation).

Este trabalho foi desenvolvido dentro do **Programa de Desenvolvimento Conjunto de Pesquisas em Comunicação de Dados** do qual participam o Centro Científico Rio da IBM do Brasil e o Laboratório de Sistemas Digitais do Departamento de Engenharia de Eletricidade da Escola Politécnica da Universidade de São Paulo.

1. INTRODUÇÃO

Os testes de protocolo têm como objetivo principal verificar se uma dada máquina de protocolo (PM - Protocol Machine) comporta-se de acordo com a especificação utilizada. Dentro deste contexto, são efetuados os testes de operacionalidade, conformidade e interoperabilidade.

Para realizar tais testes é necessário definir os métodos de teste (arquiteturas dos testadores) e gerar os cenários de teste a serem aplicados concomitantemente com tais métodos.

Este artigo apresenta o cenário de teste obtido aplicando-se um método específico de geração de seqüência de teste e apresenta um exemplo de aplicação da linguagem TTCN (The Tree Tabular Combined Notation) usada na descrição deste cenário.

2. GERAÇÃO DE CENÁRIOS DE TESTE

Um cenário de teste (Test Suite) corresponde a uma seqüência completa de testes, que inclui primitivas de serviço e mensagens de protocolo usadas para exercitar e verificar o comportamento da implementação sob Teste (IUT - Implementation Under Test), ou seja, da máquina de protocolo implementada.

Existem vários métodos que podem ser usados para gerar as seqüências de teste a serem aplicadas no teste de máquinas de protocolo modeladas como máquinas de estado finitas (SFM - State Finite Machine). Dentre estes métodos podem-se citar: Método de Turnê de Transições (Transition Tour), Método W (Método de Seqüência de Caracterização) e Método D (Método de Seqüência de Verificação).

Neste caso optou-se por aplicar o Método D ou Método de Seqüência de Verificação para gerar o cenário de teste a ser usado nos testes de uma implementação do protocolo de transporte classe 2. A escolha deste método foi motivada pelo fato de gerar uma seqüência de teste que permite detectar tanto erros funcionais como erros de transição e pelo fato desta seqüência não ser muito grande.

Nos próximos itens são apresentados o método utilizado na geração do cenário de teste e o cenário de teste obtido.

2.1 Método de Geração de Cenários de Teste

O Método de Seqüência de Verificação ou Método D é aplicável para máquinas de estado que sejam fortemente conectadas (todos estados alcançáveis) e tenham uma seqüência de distinção Xd. A seqüência de distinção corresponde a uma seqüência de entrada

que produz uma seqüência de saída diferente para cada estado. Desde que tal seqüência exista, este método pode ser aplicado também em máquinas de estado incompletas.

A seqüência de teste usada no caso deste método é constituída de duas partes:

- Seqüência de Reconhecimento de Estados - nesta etapa é determinada a resposta (seqüência de saída) de cada estado à seqüência de distinção (Xd);

- Seqüência de Verificação das Transições - nesta etapa são testadas todas as transições da máquina de estado correspondente. Isto é feito colocando, através de uma seqüência de transferência, a máquina de estado em cada um de seus estados e aplicando, para cada estado Ei, uma seqüência Xi.Xd, onde Xi é uma seqüência de entrada que provoca a transição do estado Ei para o estado desejado Ej.

O comprimento máximo de uma seqüência de teste gerada aplicando-se este método é obtido a partir da seguinte relação:

$U <= I(a) + I(b)$, sendo $I(a) = 2nL + 2(n - 1)$ e

$I(b) = q(n + L)$

i.e., $U < 2nL + 2(n - 1) + q(n + L)$ (2.1)

onde:

U - comprimento máximo da seqüência de teste

I (a) - comprimento máximo da seqüência de reconhecimento de estados

I (b) - comprimento máximo da seqüência de verificação das transições

n - número de estados

L - comprimento da seqüência de distinção (Xd)

q - número total de entradas válidas na tabela de estados

Através da aplicação da seqüência correspondente ao Método de Seqüência de Verificação ou Método D é possível detectar erros funcionais e de transição, contudo este método não considera a existência de estados extras (incluídos na fase de implementação) nem de estados perdidos.

2.2 Aplicação do Método de Seqüência de Verificação

Este Método de Seqüência de Verificação foi utilizado na ge

ração das seqüências de teste para uma implementação do protocolo de transporte classe 2. A máquina de protocolo correspondente foi especificada através de três máquinas de estado relativas às componentes de gerenciamento da conexão de rede (fase de conexão de rede), de gerenciamento da conexão de transporte (fase de estabelecimento e liberação da conexão de transporte) e de controle de fluxo (fase de transferência de dados) respectivamente. Para efeito de ilustração, são apresentadas na Figura 2.2 (a) e (b) as seqüências de teste de reconhecimento de estados e de verificação das transições, geradas para a máquina de estado correspondente à componente de gerenciamento da conexão de transporte (Figura 2.1). No caso deste exemplo, obteve-se a seqüência de distinção Xd: CC.TDISreq.DR por inspecção da máquina de estado correspondente.

A seqüência de teste obtida tem comprimento igual a 121 (42 + 79). Aplicando-se a relação 2.1, obtém-se, como comprimento máximo U desta seqüência, o valor igual a 196 (61 + 135).

As seqüências geradas aplicando-se este método são usadas a seguir quando é apresentado um exemplo de aplicação da linguagem TTCN na descrição de cenários de teste.

3. NOTAÇÃO TTCN (THE TREE AND TABULAR COMBINED NOTATION)

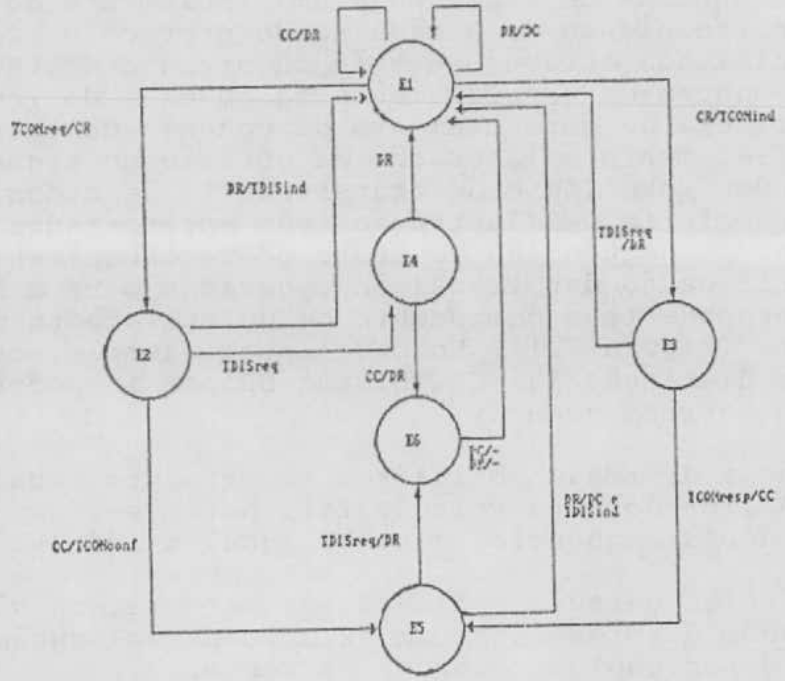
3.1 Informações Gerais

Esta notação foi formulada pela CCITT (International Telegraph and Telephone Consultative Committee) para especificar Cenários de Teste. É uma técnica de descrição semi-formal, que se utiliza de tabelas para descrever as primitivas de serviço, unidades de dados de protocolo e demais elementos necessários (temporizadores, constantes, variáveis, etc) ao teste e de árvores para descrever a seqüência dos testes.

Um Cenário de Teste especificado em TTCN é constituído de quatro partes:

a) Descrição Global do Cenário de Teste

É uma tabela onde são encontradas informações gerais sobre o Cenário de Teste, como por exemplo, referências ao PICS (Protocol Implementation Conformance Statement) e PIXIT (Protocol Implementation Extra Information for Testing), informação sobre o método de teste a ser aplicado, referência aos Casos de Teste, seus identificadores, seus objetivos e onde encontrá-los dentro desta especificação.



Legenda:

Estados:

E1 (Estado Desconectado) - Não existe conexão de transporte estabelecida;

E2 (Estado Espera Confirmação Conexão) - A estação local enviou um pedido de estabelecimento de conexão à estação remota;

E3 (Estado Espera Resposta Conexão) - A estação local recebeu um pedido de estabelecimento de conexão da estação remota, e sua aceitação depende da resposta do usuário local dos serviços de transporte;

E4 (Estado Espera Enviar Desconexão) - O usuário local pediu desconexão antes de receber a confirmação do seu pedido de conexão anteriormente feito;

E5 (Estado Conectado) - Existe uma conexão de transporte estabelecida.

E6 (Estado Pediu Desconexão) - A estação local enviou um pedido de desconexão).

Primitivas de Serviço:

TCON.req - primitiva T_CONNECT_request
TCON.ind - primitiva T_CONNECT_indication
TCON.res - primitiva T_CONNECT_response
TCON.conf - primitiva T_CONNECT_confirmation
TDIS.req - primitiva T_DISCONNECT_request
TDIS.ind - primitiva T_DISCONNECT_indication

Mensagens de Protocolo (TPDU - Transport Protocol Data Unit)

CR - TPDU de pedido de conexão
CC - TPDU de confirmação de conexão
DR - TPDU de pedido de desconexão
DC - TPDU de confirmação de desconexão

Figura 2.1 - Máquina de Estado correspondente à componente de gerenciamento da conexão de transporte do protocolo de transporte classe 2.

Entrada Estado Atual/Saída	TCONreq 1/CR	DR 2/TDISind	Xd 1/Saida1	TCONreq 1/CR	2
Entrada Estado Atual/Saída	CC 2/TCONconf	Xd 5/Saida5	TCONreq 1/CR	TDISreq 2/-	4
Entrada Estado Atual/Saída	Xd 4/Saida4	CR 1/TCONind	TDISreq 3/DR	Xd 1/Saida1	1
Entrada Estado Atual/Saída	CR 1/TCONind	TCONres 3/CC	Xd 5/Saída5	TCONreq 1/CR	2
Entrada Estado Atual/Saída	TDISreq 2/-	CC 4/DR	Xd 6/Saída6	TCONreq 1/CR	2
Entrada Estado Atual/Saída	TDISreq 2/DR	DR 4/-	Xd 1/Saída1	TCONreq 1/CR	2
Entrada Estado Atual/Saída	CC 2/TCONconf	TDISreq 5/DR	Xd 6/Saída6	CR 1/TCONind	3
Entrada Estado Atual/Saída	TCONres 3/CC	DR 5/DC,TDISind	Xd 1/Saída1	TCONreq 1/CR	2
Entrada Estado Atual/Saída	TDISreq 2/-	CC 4/DR	DC 6/-	Xd 1/Saída1	1
Entrada Estado Atual/Saída	CR 1/TCONind	TCONres 3/CC	TDISreq 5/DR	DR 6/-	1
Entrada Estado Atual/Saída	Xd 1/Saída1	CC 1/DR	Xd 1/Saída1	TCONreq 1/CR	2
Entrada Estado Atual/Saída	Xd 2/Saída2	DR 1/DC	Xd 1/Saída1	CR 1/TCONind	3
Entrada Estado Atual/Saída	Xd 3/Saída3				

(a) Seqüência de Teste de Verificação das Transições

Entrada Estado Atual/Saída	TCONreq 1/CR	Xd 2/Saída2	Xd 1/Saída1	CR 1/TCONind	3
Entrada Estado Atual/Saída	Xd 3/Saída3	Xd 1/Saída1	TCONreq 1/CR	TDISreq 2/-	4
Entrada Estado Atual/Saída	Xd 4/Saída4	Xd 1/Saída1	TCONreq 1/CR	CC 2/TCONconf	5
Entrada Estado Atual/Saída	Xd 5/Saída5	Xd 1/Saída1	CR 1/TCONind	TCONres 3/CC	5
Entrada Estado Atual/Saída	TDISreq 5/CR	Xd 6/Saída6	Xd 1/Saída1	Xd 1/Saída1	1

(b) Seqüência de Teste de Reconhecimento de Estados

Legenda

Xd - Seqüência de distinção

Para a seqüência Xd igual a CC.TDISreq.DR, têm-se as seguintes saídas:

```
Estado E1 -   DR/1      ,   -   , DC/1 - Saída 1
Estado E2 - TCONconf/5 , DR/6 ,  -/1 - Saída 2
Estado E3 -     -       , DR/1 , DC/1 - Saída 3
Estado E4 -   DR/6      ,   -   ,  -/1 - Saída 4
Estado E5 -     -       , DR/6 ,  -/1 - Saída 5
Estado E6 -     -       ,   -   ,  -/1 - Saída 6
```

Figura 2.2 - Sequência de Teste do Método de Sequência de Verificação Aplicado à Componente de Gerenciamento da Conexão de Transporte do Protocolo de Transporte Classe 2

b) Declarações

Esta parte é formada por uma série de tabelas que descrevem o conjunto de eventos de teste e todos os outros atributos usados no cenário de teste.

Existem dois tipos de eventos de teste:

- ASPs (Abstract Service Primitives)
  São as primitivas de serviço que ocorrem nos PCOs (Pontos de Controle e Observação) usados pelo testador.

- Eventos de Temporização
  Gerados pelos temporizadores.

Os demais atributos a serem especificados, são os seguintes:

- Tipos Definidos pelo Usuário
- Operadores Definidos pelo Usuário
- Parâmetros do Cenário de Teste
- Constantes Globais
- Variáveis Globais
- PCOs
- Parâmetros das ASPs
- Tipos de Dados (incluindo as PDUs e seus parâmetros)

c) Parte Dinâmica

Descreve o comportamento dinâmico do teste em uma notação de árvores, fazendo-se uso dos eventos e atributos declarados na parte anterior.

Uma árvore corresponde a enumerações de possíveis seqüências de eventos de teste observáveis. Como exemplo, supõe-se que as seguintes seqüências de eventos podem ocorrer durante um teste, cujo propósito é estabelecer uma conexão, trocar dados e liberar a conexão anteriormente estabelecida:

1 - CONreq, CONcnf, DATreq, DATind, DISreq
2 - CONreq, CONcnf, DATreq, DATind, DISind
3 - CONreq, CONcnf, DATreq, DISind
4 - CONreq, CONcnf, DISind
5 - CONreq, DISind

usando a notação de árvore, tem-se:

```
------------------------> tempo

Arvore-Exemplo
  CONreq
    CONcnf
      DATreq
        DATind
          DISreq
          DISind
        DISind
      DISind
    DISind

 |  |  |  |  |   --> níveis de indentação
```

Conforme pode-se verificar são fatoradas as seqüências comuns do conjunto completo, sendo os eventos alternativos apresentados no mesmo nível de indentação.

Qualquer descrição de comportamento contém pelo menos uma árvore, que é identificada por um nome único dentro do escopo das descrições.

d) Codificação dos Parâmetros PDUs/ASPs

Uma ASP ou PDU pode ser considerada como uma lista de parâmetros. Esta parte descreve a codificação destes parâmetros.Nesta codificação pode ser usado tanto o método tabular quanto o método usado na notação ASN.1 (Abstract Syntax Notation. 1).

A seguir é dado um exemplo elucidativo, que permitirá esclarecer dúvidas quanto à notação TTCN.

3.2 Exemplo de Utilização da Notação TTCN

Na figura a seguir, é apresentada a estrutura do Cenário de Teste para o protocolo de transporte classe 2 da ISO. Observa-se três grupos de teste: fase de conexão de rede, fase de estabelecimento e liberação da conexão de transporte e fase de transferência de dados. Neste exemplo, apenas o segundo Grupo de Teste é especificado em TTCN.

Como foi dito anteriormente, o método adotado para geração das seqüências de testes foi o Método de Seqüência de Verificação.

Considerando que todas as interfaces da entidade de transporte podem ser acessadas diretamente, o método de teste adotado foi o local.

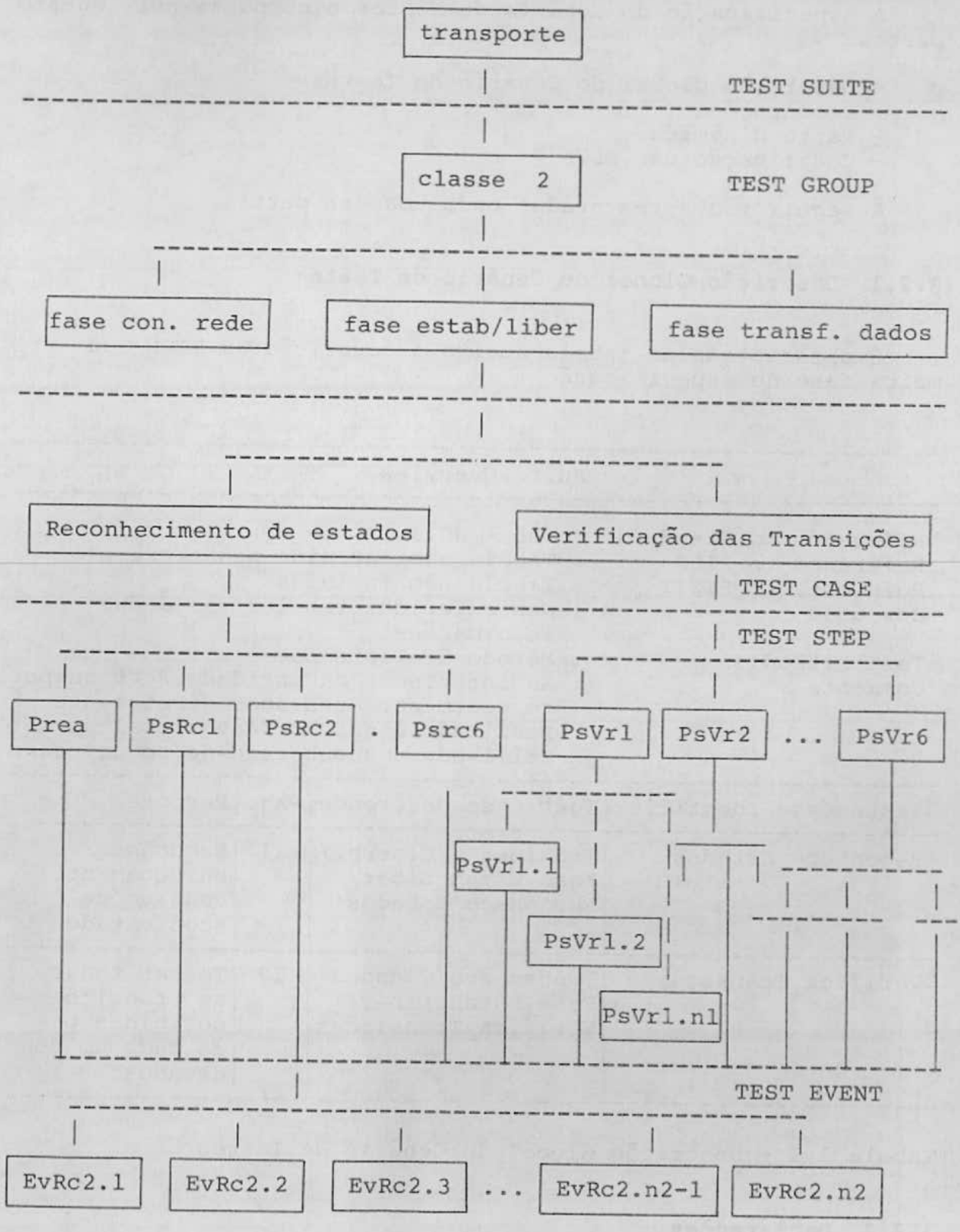


Figura 3.1 - Cenário de Teste do Protocolo de Transporte de Transporte Classe 2 da ISO

A especificação do Cenário de Testes é composta por quatro partes:

- Descrição Global do Cenário de Testes
- Declarações
- Parte Dinâmica
- Codificação das PDUs

A seguir são apresentadas cada uma das partes.

3.2.1 Descrição Global do Cenário de Teste

É apresentada na tabela abaixo a descrição em TTCN da primeira fase de especificação.

Suite Overview	
Reference to Standards	: ISO - 8073
Reference to PICS	: Tabela não incluída neste exemplo
Reference to PIXIT	: Tabela não incluída
How used	: Nas referidas tabelas são dadas informações
Test Method(s)	: Método de Teste Local
Comments	: As interfaces da entidade de transporte podem ser acessadas diretamente. É realizado apenas o teste local sem ser utilizada uma sub-rede de comunicação.

Teste Case Identifier	Test Case Reference	Page	Purpose
Reconhece_Estados	Transporte/Classe2/ Fase_Estab_Liber/ Reconhece_Estados	11	Reconhecer univocamente cada um de seus estados
Verifica_Transições	Transporte/Classe2/ Fase_Estab_Liber/ Verifica_Transições	17	Testar todas as transições definidas para cada um dos estados

Tabela 3.1 - Descrição Global do Cenário de Testes

3.2.2 Declarações

A seguir são declarados os eventos e os atributos.

3.2.2.1 Abstract Service Primitives (ASPs)

Neste item são declaradas as primitivas que ocorrem nos PCOs usados pelo testador.

ASP Declaration				
ASP	TCONreq (T_Connect_request)	PCO|UT	CEID	Não usado

Service Control Information		
Parameter Name	Type	Comments
ECD (Endereço Chamado)	END	Testador Superior
ECN (Endereço Chamante)	END	Testador Inferior
ODE (Opção de Dados Expressos)	BOOL	Valor Booleano
QOS (Qualidade de Serviço)	QDS	Lista de Parâmetros
DUT (Dados do Usuário)	DU	Até 32 octetos

Tabela 3.2 - Declaração da Primitiva T_CONNECT_request

As declarações das outras ASPs (T_CONNECT_indication, T_CONNECT_response, T_CONNECT_confirmation, T_DISCONNECT_request, T_DISCONNECT_indication, N_DATA_request e N_DATA_indication) são semelhantes à apresentada na Tabela 3.2; por isso, não são incluídas neste exemplo.

3.2.2.2 Pontos de Controle e Observação (PCOs)

Na tabela a seguir são declarados os PCOs utilizados pelos testadores inferior e superior.

PCO Declarations	
Name	Role
UT	TSAP (Transport Service Access Point)-Test.Superior
LT	NSAP (Network Service Access Point) -Test.Inferior

Tabela 3.3 - Declaração dos PCOs

3.2.2.3 Temporizadores

Na próxima tabela é especificado o temporizador usado na descrição do comportamento dinâmico da máquina de protocolo considerada.

Timer Declarations		
Timer Type Name	Duration	Comments
Event_Null	1	Define o temporizador a ser usado no pseudo_evento Elapse

Tabela 3.4 - Declaração dos temporizadores

3.2.2.4 Abreviação

Na tabela seguinte são declaradas as abreviações adotadas.

Abbreviations		
Abbreviation	Expansion	
CR	N_Data [NSDU - CR_TPDU]	CR denota qualquer N_Data (pedido ou indicação) cuja NSDU (Network Service Data Unit) for a codificação de uma Connection Request Transport Data Unit.
CC	N_Data [NSDU - CC_TPDU]	
DR	N_Data [NSDU - DR_TPDU]	
DC	N_Data [NSDU - DC_TPDU]	

Tabela 3.5 - Declaração das Abreviações

3.2.2.5 Protocol Data Units (PDUs)

A seguir são declaradas as PDUs envolvidas na fase de estabelecimento e liberação da conexão de transporte.

Data Type Declaration			
PDU	CR (Connect Request TPDU)	Comments	Não faz referência a campos padronizados em outra norma
Protocol Control Information			
Field Name			
LI (Length Indication)		Octetstring	Tamanho do cabeçalho em número de octetos
CR (Connection Request Code)		Bitstring	Código da TPDU
CDT (Initial Credit)		Bitstring	Crédito inicial
DST_REF (Destiny Reference)		Octetstring	Valor igual a zero
SRC_REF (Source Reference)		Octetstring	Identificação da conexão
Class Option		Octetstring	Seleção da classe de Protocolo
Variable Part		Octetstring	Lista de Parâmetros
User Data		Octetstring	Dados do usuário até 32 octetos

Tabela 3.6 - Declaração da CR_TPDU

As declarações das outras TPDUs (CC, DR e DC) são semelhantes à apresentada na Tabela 3.6; por isso, não são incluídas neste exemplo.

3.2.2.6 Tipos Definidos pelo Usuário

As declarações dos tipos que o usuário definiu são apresentadas a seguir.

User Type Definition			
Name	Base Type	Definiton	Comments
END	Octetstring	9	9 Octetos (Network Add. + Transport Select)
BOOL	-	(False,True)	
QDS	Octetstring	ISO - 8073	Definição encontrada nesta norma
DU	Octetstring	1..32	Até 32 octetos
DDU	Octetstring	1..64	Até 64 octetos
NDU	Octetstring	1..128	Até 128 octetos

Tabela 3.7 - Declaração dos Tipos definidos pelo Usuário

3.2.3 Parte Dinâmica

Nesta parte da especificação são apresentados os Passos e os Casos de Teste referenciados na parte "Descrição Global do Cenário de Testes". São descritos por meio de tabelas e seus comportamentos dinâmicos por meio de árvores.

3.2.3.1 Test Case: Reconhece Estados

A Tabela seguinte descreve um dos casos de teste, o "Reconhece Estados".

Dynamic Behaviour				
Reference	: Transporte/Classe2/Fase_Estab_Lib/ Reconhece_Estados			
Identifier	: Rec_Estados			
Purpose	: Reconhecer cada um dos estados			
Defaults Reference	: Inexistente			
Behaviour Description	**Label**	**Const. Refer.**	**Ver Dict**	**Comments**
FASE_EST_LIBER_TREE [UT,LT]				
+PREAMBLE [ut,LT]				Est.Inicial
+PsRc2 [UT,LT]				
+PsRc3 [UT,LT]				Reconhece
+PsRc4 [UT,LT]				todos os
+PsRc5 [UR,LT]				estados
+PsRc6 [UT,LT]				
+PsRc1 [UT,LT]				

Tabela 3.8 - Descrição do Test Case: Reconhece_Estados

Os Passos de Teste (Test Steps) apresentados acima são descritos a seguir.

A seqüência de eventos correspondente ao PREAMBLE coloca a máquina de protocolo no estado inicial a partir do qual são iniciados os testes.

Dynamic Behaviour				
Reference	: Transporte/Classe2/Fase_Estab_Lib/ Reconhece_Estados | Preamble			
Identifier	: Preamble			
Purpose	: Coloca a máquina de estado no estado inicial			
Defaults Reference	: Inexistente			
Behaviour Description	**Label**	**Const. Refer.**	**Ver Dict**	**Comments**
PREAMBLE UTITDISreq LTIDR +PsRcl [UT,LT]				Esta seqüência coloca a SFM no estado inicial

Tabela 3.9 - Descrição do Test Step : Preamble

Em seguida, tem-se os Passos de Teste para o reconhecimento dos estados 1, 2, 3, 4, 5 e 6.

Dynamic Behaviour				
Reference	: Transporte/Classe2/Fase_Estab_Lib/ Reconhece_Estados | PsRcl			
Identifier	: PsRcl			
Purpose	: Verifica qual é a seqüência de saída no estado 1 em resposta a sequência de distinção.			
Defaults Reference	: Inexistente			
Behaviour Description	**Label**	**Const. Refer.**	**Ver Dict**	**Comments**
PsRcl [UT,LT] LT!CC LT?DR UT!TDISreq Elapse LT!DR LT?DC ?Otherwise ?Otherwise ?Otherwise			 Pass Falha Falha Falha	Nome da Árvore Fica neste nível por um dado tempo. Se não ocorrer nenhum evento então este pseudo evento é disparado

Tabela 3.10 - Descrição do Test Step : PsRcl

Dynamic Behaviour				
Reference	: Transporte/Class2/Fase_Estab_Lib/ Reconhece_Estados | PsRc2			
Identifier	: PsRc2			
Purpose	: Verifica qual é a sequência de saída no estado 2 em resposta a sequência de distinção.			
Defaults Reference	: Inexistente			
Behaviour Description	**Label**	**Const. Refer.**	**Ver dict**	**Comments**
PsRc2 [UT,LT]				Nome da Árvore
UT!TCONreq				Coloca no estado2
LT?CR				
LT!CC				
UT?TCONconf				
UT!TD!Sreq				
LT?DR				
LT!DR				
Elapse				Se expirar o intervalo de tempo significa que nenhuma saída ocorreu
+PsRc1[UT,LT]				Retorna ao estado 1
?Otherwise			Falha	
?Otherwise			Falha	
?Otherwise			Falha	
?Otherwise			Falha	

Tabela 3.11 - Descrição do Test Step : PsRc2

Dynamic Behaviour				
Reference : Transporte/Classe2/Fase_Estab_Lib/ Reconhece_Estados | PsRc3				
Identifier : PsRc3				
Purpose : Verifica qual é a sequência de saída no estado 3 em resposta a sequência de distinção.				
Defaults Reference : Inexistente				
Behaviour Description	**Label**	**Const. Refer.**	**Ver Dict**	**Comments**
PsRc3 [UT,LT]				Nome da Árvore
LT!CR				Coloca no estado3
UT?TCONind				
LT!CC				
Elapse				
UT!TDISreq				
LT?DR				
LT!DR				
LT?DC				
+PsRc1[UT,LT]				Retorna ao estado1
?Otherwise			Falha	
?Otherwise			Falha	
?Otherwise			Falha	
?Otherwise			Falha	

Tabela 3.12 - Descrição do Test Step : PsRc3

Dynamic Behaviour				
Reference	: Transporte/Classe2/Fase_Estab_Lib/ Reconhece_Estados | PsRc4			
Identifier	: PsRc4			
Purpose	: Verifica qual é a seqüência de saída no estado 4 em resposta a seqüência de distinção.			
Defaults Reference	: Inexistente			
Behaviour Description	**Label**	**Const. Refer.**	**Ver Dict**	**Comments**
PsRc4 [UT,LT]				Nome da Árvore
UT!TCONreq				Coloca no estado4
LT?CR				
UT!TDISreq				
Elapse				
LT!CC				
LT?DR				
UT!TDISreq				
Elapse				
LT!DR				
Elapse				
+PsRc1[UT,LT]				Retorna ao estado 1
?Otherwise			Falha	
?Otherwise			Falha	
?Otherwise			Falha	
?Otherwise			Falha	
?Otherwise			Falha	

Tabela 3.13 - Descrição do Test Step : PsRc4

Dynamic Behaviour				
Reference : Transporte/Classe2/Fase_Estab_Lib/ Reconhece_Estados | PsRc5				
Identifier : PsRc5				
Purpose : Verifica qual é a seqüência de saída no estado 5 em resposta a seqüência de distinção.				
Defaults Reference : Inexistente				
Behaviour Description	**Label**	**Const. Refer.**	**Ver dict**	**Comments**
PsRc5 [UT,LT]				Nome da Árvore
UT!TCONreq				
LT?CR				
LT!CC				
UT?TCONconf				coloca no estado 5
TL!CC				
Elapse				
UT!TDISreq				
LT?DR				
LT!DR				
Elapse				Termina Reconhecimento e
+PsRcl[Ut,LT]				Retorna ao estado 1
?Otherwise			Falha	
?Otherwise			Falha	
?Otherwise			Falha	
?Otherwise			Falha	
?Otherwise			Falha	

Tabela 3.14 - Descrição do Test Step : PsRc5

Dynamic Behaviour				
Reference : Transporte/Classe2/Fase_Estab_Lib/ Reconhece_Estados | PsRc6				
Identifier : PsRc6				
Purpose : Verifica qual é a seqüência de saída no estado 6 em resposta a seqüência de distinção.				
Defaults Reference : Inexistente				
Behaviour Description	**Label**	**Const. Refer.**	**Ver Dict**	**Comments**
PsRc6 [UT,LT]				Nome da Árvore
LT!CR				
UT?TCONind				
UT!TCONresp				
LT?CC				coloca no estado 5
UT!TDISreq				
LT?CR				
LT!CC				
Elase				
UT!TDISreq				
Elapse				
LT!DR				
Elapse				Termina Reconhecimento e
+PsRcl[Ut,LT]				Retorna ao estado 1
?Otherwise			Falha	
?Otherwise			Falha	
?Otherwise			Falha	
?Otherwise			Falha	
?Otherwise			Falha	

Tabela 3.15 - Descrição do Test Step : PsRc6

3.2.3.2 **Test Case** : Verificação das Transições

Não está incluído neste exemplo.

3.2.4 Codificação das PDUs e ASPs

Na seqüência de teste gerada não houve necessidade de codificar as PDUs e ASPs.

4. CONSIDERAÇÕES FINAIS

Em relação ao método de geração de seqüência de teste optou-se pelo Método de Seqüência de Verificação pois ele gera seqüências de teste através das quais pode-se detectar tanto erros funcionais como de transição. Contudo, nem sempre é possível encontrar a Seqüência de Distinção Xd. No exemplo apresentado, tal seqüência foi determinada por inspecção da máquina de estado correspondente. Na realidade, o correto seria aplicar algoritmos para determinar automaticamente tal seqüência. A pesquisa e estudo de tais algoritmos devem ser objetos de trabalhos futuros.

Além da aplicação de um método de geração de seqüência de teste, neste artigo é apresentada uma visão geral sobre a notação TTCN usada para descrever ambientes de teste, incluindo o comportamento estático (descrição das ASPs e PDUs) e dinâmico da implementação sob teste. Como exemplo, é apresentada a descrição em TTCN de um Grupo de Teste do protocolo de transporte classe 2. Devido ao número de tabelas que precisam ser elaboradas na descrição do ambiente de teste de qualquer protocolo ou conjunto de protocolos, percebeu-se a necessidade de desenvolver ferramentas de auxílio para montagem de tabelas.

O exemplo utilizado neste artigo corresponde a uma parte do Cenário de Teste empregado nos testes de uma implementação do protocolo de transporte classe 2.

5. REFERÊNCIAS BIBLIOGRÁFICAS

[1] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. Modelo Básico de Referência - Sistemas de Processamento de Informação - Interconexão de Sistemas Abertos. Rio de Janeiro. Associação Brasileira de Normas Técnicas, 1986. 60 p.

[2] CARVALHO, Tereza C. M. B. Testes de uma Implementação de FTAM (File Transfer Access and Management). In : 7o . SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES. Anais Porto Alegre, 1989. p. 106 - 130.

[3] GABOS, Denis; STIUBIENER, Stefania. Aspectos de Metodologia de Geração de Seqüências de Testes para Protocolos de Comunicação de Dados. In : 7o. SIMPÓSIO BRASILEIRO DE COMPUTADORES. Anais ... Porto Alegra, 1989. p. 556 - 576.

[4] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO 7498 - Information Processing Systems - Open Systems Interconnection - Basic Reference Model. EUA, American National Standards Association, Inc., 1984. 40 p.

[5] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO 8072 - Information Processing Systems - Open Systems Interconnection - Transport Service Definition. International Organization for Standardization, 1986. 16p.

[6] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO 8073 - Information Processing Systems - Open Systems Interconnection - Transport Protocol Specification . International Organization for Standardization, 1986. 67p.

[7] INTERNATIONAL TELEGRAPH AND TELEPHONE CONSULTATIVE COMMITTEE. Draft Recommendation X.290 - OSI Conformance Testing Methodology and Framework for CCITT Applications (Final Version), 1988. 128 p.

[8] KINGHTSON, K. G. Open Systems Interconnection Conformance and Conformance Testing. Open Systems Data Transfer. EUA, Omnicom Inc., (21)118, abr. 1986.

[9] RAYNER, D. A System for Testing Protocol Implementations. In | --- Protocol Specification, Testing and Verification. Holanda, North-Holland Publishing Company, 1982. p.539-54.

[10] SARIKAYA, Behcet; BOCHMAN, Gregor V. Some Experience with Test Sequence Generation for Protocols. In | --- Protocol Specification, Testing and Verification. Holanda, North-Holland Publishing Company, 1982. p.555-67.

[11] SARIKAYA, Behcet. Recent Developments in Protocol Testing. In | 5o. SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES. Anais ... São Paulo, 1987. p. 372-92.

[12] STIUBIENER, Stefania. Testes de Protocolos para Redes de Computadores. In : 7o. SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES: Anais ... Porto Alegre, 1989. p. 208-220.