UM TRADUTOR TTCN-ESTELLE PARA TESTE DE IMPLEMENTAÇÕES DE PROTOCOLOS

Mônica Paiva de Almeida*
Rua Pinheiros 98/102. Horto, Ipatinga - MG - 35.160.

José Marcos Siiva Nogueira†
UFMG - ICEx - DCC
Caixa Postal 702 - Belo Horizonte - MG - 30161.

1991

Resumo

Este trabalho descreve um tradutor de casos de teste abstratos, especificados na notação TTCN, para casos de teste executáveis escritos na técnica de descrição formal Estelle, a serem utilizados no teste de implementações de protocolos de comunicação. Estes casos de teste em Estelle permitem uma realização automática dos testes. A entrada do tradutor consiste das partes dinâmica, de declarações e de restrições de um caso de teste; a sua saída corresponde a um único módulo em Estelle, incluindo a definição de sua interface, canais e interações.

Abstract

This work describes a translator for abstract test cases specified in the TTCN notation into executable test cases in the formal description technique Estelle. The test cases are to be used in protocol testing. Such test cases in Estelle made possible an automatic realization of tests. The input to the translator consists of a dynamic part, a declaration part and a constraint part of a test case; the output corresponds to one module in Estelle including definitions of interface, channels and interactions.

1 INTRODUÇÃO

Devido à larga aceitação do modelo OSI (*Open Systems Interconnection*) desenvolvido pela ISO (*International Organization for Standardization*) e seus protocolos padronizados para as várias camadas, validar as implementações destes protocolos tem se tornado uma atividade muito importante e de intensa pesquisa.

Testar uma implementação de um protocolo significa aplicar casos de teste (ou equivalentemente, seqüências de teste) que são um conjunto completo de ações necessárias para se alcançar um objetivo específico.

Um teste de conformidade procede-se pela aplicação de um cenário de teste (*test suite*), que é definido como um conjunto de grupos de teste que são usados para fornecer uma ordenação lógica dos casos de teste, tais como estabelecimento de conexão, transferência de dados, etc. Um caso de teste, por sua vez, é estruturado como um número de eventos que são unidades indivisíveis do teste. O objetivo deste tipo de teste é verificar se a implementação sob teste

*Mestre em Ciência da Computação - UFMG e Analista de Sistemas da USIMINAS.
†Professor do Departamento de Ciência da Computação da UFMG.

está em conformidade com o protocolo padrão apropriado, sem considerar seu desempenho e sua eficiência.

Para cada protocolo definido pela ISO, cenários de teste de conformidade têm sido padronizados para serem usados pelos implementadores, usuários e outros grupos de testadores. Estes padrões podem levar a uma larga comparabilidade e aceitação dos resultados produzidos por diferentes testadores, e, portanto, podem minimizar a necessidade de repetidamente testar a conformidade de um mesmo sistema.

O subgrupo de teste de conformidade da ISO definiu uma notação chamada *Tree and Tabular Combined Notation* (TTCN) para abstratamente especificar os cenários de teste de conformidade padronizados.

Em um procedimento de teste, os cenários de teste padronizados a serem aplicados devem ser selecionados e parametrizados, de acordo com algumas informações contidas nos dois seguintes documentos, que são fornecidos pelo implementador: PICS (*Protocol Implementation Conformance Statement*) e PIXIT (*Protocol Implementation Extra Information for Testing*). Os cenários selecionados e parametrizados estão em uma forma abstrata e devem ser traduzidos em cenários executáveis para serem utilizados na execução dos testes.

O objetivo deste trabalho é apresentar o desenvolvimento de um tradutor de casos de teste abstratos especificados em TTCN para casos de teste executáveis escritos na técnica de descrição formal Estelle (*Extended State Transition Language*).

Estelle foi desenvolvida pela ISO para especificar sistemas distribuídos, em particular protocolos e serviços de comunicação. A escolha de Estelle como linguagem alvo foi feita baseando-se no fato que muito trabalho com relação a ela tem sido desenvolvido, resultando em diversos tradutores que ajudam na automatização de sistemas de teste. Utilizando uma destas ferramentas, um compilador Estelle-C desenvolvido na *University of British Columbia* [14], que se encontra disponível, um caso de teste em Estelle pode então ser aplicado em uma especificação de um protocolo também especificado em Estelle, e uma forma executável pode ser obtida.

Um projeto com relação à derivação de casos de teste executáveis escritos em Estelle e na linguagem ITL (*Interactive Test Language*), a partir de casos de teste abstratos em TTCN, desenvolvido recentemente, pode ser encontrado em [17].

Na seção a seguir são apresentadas as arquiteturas utilizadas em testes de protocolos; as seções 3 e 4 descrevem brevemente Estelle e TTCN respectivamente; na seção 5 é apresentado o tradutor e alguns exemplos, e na seção 6 algumas conclusoēs.

2 ARQUITETURAS DE TESTE

As implementações dos protocolos podem ser testadas considerando uma entidade com uma única camada ou uma entidade com múltiplas camadas. A entidade sob teste deve ser estimulada pelas camadas superior e inferior e as reações da implementação sob teste (IUT - "Implementation Under Test") devem ser observadas. O estímulo ou a observação são feitos pelo envio ou recebimento de primitivas de serviço para uma entidade da camada N, por uma entidade chamada *testador*. Tais primitivas são também chamadas de *primitivas de serviço abstratas* (ASPs-(N) e ASPs-(N-1)). Os pontos de acesso a serviços (SAPs) usados pelo testador com este objetivo são chamados de *pontos de controle e observação* (PCOs) [1]. O modelo geral, conceitual, é mostrado na figura 1, onde uma entidade de protocolo da camada N é testada em separado.

Um testador pode ser funcionalmente dividido em dois testadores associados com cada um dos pontos de acesso a serviço, que são referenciados como **testador superior** e **testador inferior**.

O testador inferior (LT) é o meio de fornecer, durante a execução do teste, controle e observação no PCO apropriado, abaixo da IUT ou remotamente à IUT. Envia e recebe primitivas

ASPs-(N-1) e ASP"s-(N-1), bem como unidades de dados de protocolo PDUs-(N). Na notação, ASP indica que o testador é local à IUT, enquanto ASP" indica que o testador é remoto. O testador superior (UT) é o meio de fornecer controle e observação do limite superior da IUT e de qualquer primitiva local abstrata relevante (i.e.: ASPs-(N), ASPs-(N+1),etc.).

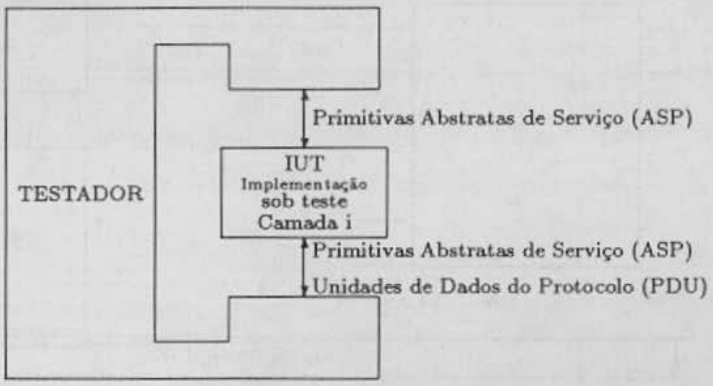

Figure 1: Arquitetura conceitual de teste

É necessário coordenar as ações entre os testadores superior e inferior para se alcançar os objetivos do teste. As regras para se alcançar tais objetivos são chamadas de **procedimentos de coordenação de teste**. Estes procedimentos sincronizam as atividades dos testadores e garantem que os eventos do teste são transferidos na seqüência apropriada. Os procedimentos necessários para um dado protocolo OSI são especificados no cenário de teste deste protocolo .

As arquiteturas de teste de protocolos podem ser *locais* ou *externas*, sendo estas referenciadas como *distribuídas*, *coordenadas* ou *remotas*. Associado com cada arquitetura de teste, tem-se um método de teste utilizado. Assim, há o método de teste local, o método de teste externo coordenado, etc. [18].

Na arquitetura de teste local os testadores superior e inferior residem localmente à implementação sob teste, como mostrado na figura 2. Esta arquitetura define os PCOs como estando nos limites de serviço acima e abaixo da IUT. Os eventos de teste são especificados em termos das ASPs-(N) acima da IUT e das ASPs-(N-1) e PDUs-(N) abaixo da IUT.

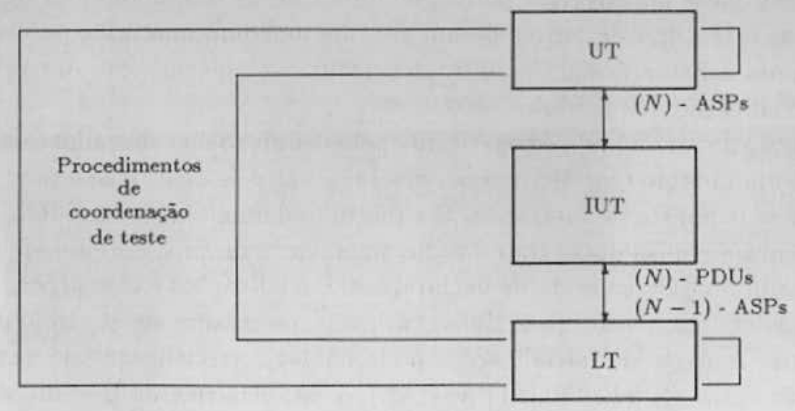

Figure 2: Arquitetura de teste local

Na arquitetura externa o testador inferior atua como uma entidade par do protocolo que está

sendo testado, e reside externamente ao sistema sob teste (SUT). O testador inferior controla e observa as ASP"s-(N-1) no sistema remoto. O testador superior tipicamente existe localmente ao SUT e funciona como o usuário dos serviços fornecidos pelo SUT. Um exemplo de uma arquitetura externa é mostrado na figura 3.

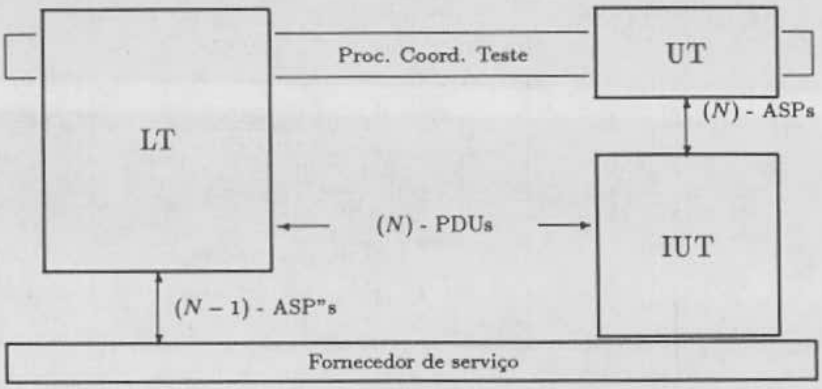

Figure 3: Arquitetura de teste externa distribuída

Uma descrição detalhada sobre as arquiteturas de teste de implementações de protocolo, mostrando as várias realizações dos testadores superior e inferior pode ser encontrada em [4]. Outras descrições de arquiteturas de teste são apresentadas em [5,6,7,9,10,11,12,13].

3 A LINGUAGEM DE ESPECIFICAÇÃO ESTELLE

Estelle é uma linguagem procedimental baseada em uma máquina de estados finita estendida (MEFE) e em elementos da linguagem de programação PASCAL, para manipular estruturas de dados e operações mais complexas.

Estelle descreve uma entidade de um protocolo como um conjunto de *módulos*. Módulos trocam primitivas de comunicação (interações). Um módulo pode enviar uma primitiva, por um de seus pontos de interação, a um outro módulo, desde que ambos estejam ligados através de um *canal*. Uma primitiva ao ser recebida por um módulo é anexada a uma fila do tipo FIFO (*first in - first out*). Uma fila FIFO pode estar associada a um único ponto de interação ou pode ser compartilhada por diversos pontos de interação de um módulo. A figura 4 mostra uma descrição apenas estrutural de partes de um sistema de communicação, para efeito de exemplo.

O canal é uma construção que permite desvincular a especificação das primitivas de comunicação da especificação dos módulos.

A especificação de um módulo é constituída pela definição do cabeçalho e do corpo do módulo. O cabeçalho de um módulo representa o seu nível mais alto de abstração e sua definição é baseada na descrição de seus pontos de interação. O cabeçalho define um tipo módulo. Variáveis do tipo módulo representam cópias desse tipo e todas possuem a mesma visibilidade externa.

O corpo de um módulo consiste de declarações, inicializações e transições. A parte relativa a declarações pode conter: constantes, tipos, variáveis, os estados de controle da MEFE, funções e procedimentos. A parte de inicializações pode conter a inicialização da variável de estado, a inicialização das variáveis adicionais (PASCAL), o estabelecimento de "*links*" de comunicação, etc. A parte de transições descreve o comportamento do módulo através de um conjunto de transições. Cada transição é constituída por duas partes: condições e ações. As condições são expressas como cláusulas próprias a Estelle, que incluem: **priority**, **from**, **provided**, **when** e **delay**. Estas podem estar relacionadas em qualquer ordem. As ações são expressas pela cláusula

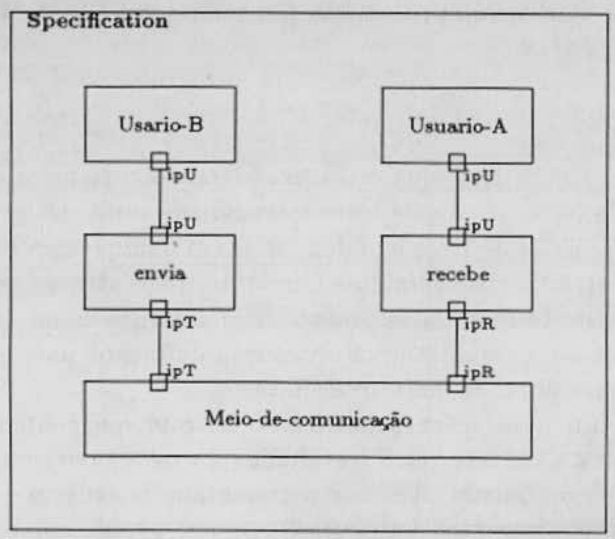

Figure 4: Diagrama estrutural exemplo, como descrito em Estelle

to e por um bloco de transição, que é constituído por uma seqüência de instruções PASCAL, extensões Estelle e restrições [2,3]. A título de ilustração, segue um exemplo de especificação de transições.

```
trans when pi.mens(p)
    from OCIOSO to ESPERA
        provided (p.sequencia > s)
            begin
                salva-copia(p);
                output pi.ack
            end;

trans when pi.mens(p)
    from OCIOSO to ESPERA
        provided not (p.sequencia > s)
            begin
                output p1.resp
            end;

trans when pi.mens(p)
    from OCIOSO
        begin
        end;
```

4 A NOTAÇÃO TTCN

TTCN é uma notação semi-formal com semânticas claramente definidas (mas não formalmente), desenvolvida pelo subgrupo de teste de conformidade da ISO. Foi projetada para especificar casos de teste abstratos, de tal forma que seja independente dos métodos de teste, camadas e protocolos. É apresentada em duas formas: uma forma gráfica (TTCN.GR), adequada para o

entendimento humano e uma forma processável por máquina (TTCN.MP), que apresenta uma sintaxe formal para TTCN.GR [8].

ESTRUTURAÇÃO

Um cenário de teste em TTCN tem uma estrutura hierarquizada pelos seguintes componentes: grupos de teste, casos de teste, passos de teste e eventos de teste. Os grupos de teste fornecem uma ordenação lógica dos casos de teste e podem estar em qualquer profundidade. Cada caso de teste tem um objetivo estritamente definido e é modularizado através de subdivisões chamadas passos de teste. Passos de teste comuns podem estar agrupados em bibliotecas de passos de teste. Os passos de teste consistem de uma ordenação de outros passos/eventos de teste. Um evento, por sua vez, é uma unidade indivisível do teste.

O nível mais importante nesta hierarquia é o caso de teste, que contém uma descrição precisa das seqüências de eventos e os *veredictos* (resultados de teste) relacionados. Esta descrição é estruturada como uma árvore, onde os nodos representam os eventos e as folhas os veredictos atribuídos. Esta estrutura é mostrada na figura 5.

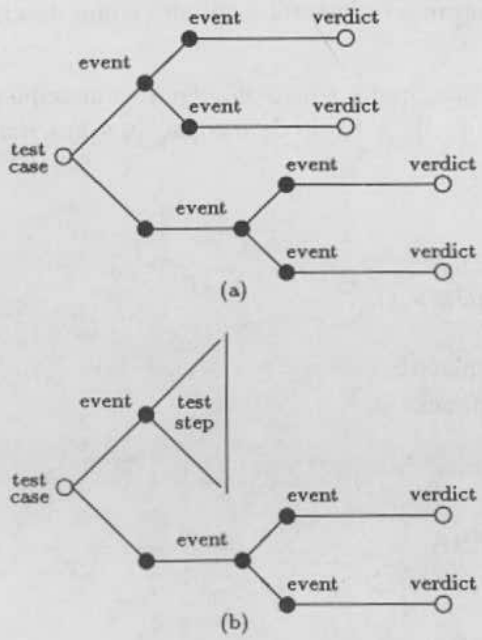

Figure 5: Comportamento de um caso de teste

COMPONENTES DE UM CENÁRIO DE TESTE

Uma especificação de um cenário de teste consiste de quatro seções : Sumário do Teste, Declarações, Parte Dinâmica e Parte de Restrições.

No sumário do teste são descritas as informações necessárias para o entendimento do teste e o seu objetivo.

Na parte de declarações são descritos todos os componentes referenciados na parte dinâmica e na parte de restrições. São declaradas as constantes, variáveis, parâmetros, Pontos de Controle e Observação (PCOs), Unidades de Dados de Protocolo (PDUs), Primitivas de Serviços Abstratas (ASPs), temporizadores, etc.

A parte dinâmica corresponde ao corpo principal do cenário de teste. Nesta parte são definidos os casos de teste, os passos de teste e os comportamentos *default*. O comportamento dinâmico de um caso ou passo de teste é descrito através da notação de árvore, consistindo de combinações de seqüências de eventos observáveis que são julgados possíveis pelo especificador do cenário de teste. Uma linha de evento inclui os eventos de teste inicializados ou recebidos pelo testador, pseudo-eventos e instruções TTCN. Os eventos TTCN são as ASPs ou PDUs inicializadas ou recebidas pelo testador inferior ou superior, eventos **timeout**, **otherwise** e **elapse**. TTCN também suporta as instruções **goto**, **attach** e **repeat**. Os eventos podem ser acompanhados por expressões booleanas, cláusulas de atribuição e operações de temporização. Devem ser especificados os veredictos relacionados com as possíveis seqüências de eventos. Estes veredictos podem ser *pass, fail, inconclusive* ou *result*. A figura 6 apresenta um exemplo da descrição do comportamento dinâmico de um caso de teste na notação TTCN.GR.

Comportamento dinâmico

Referência: Protocolo de Transporte/Testador superior/Transferência de dados
Identificador: TP-UT-DTI
Objetivo: Transferência de dados inicilizada pelo UT
Referência default: nenhuma

Descrição de comportamento	Label	Rest.	Resultados	Comentário
TEST-TP[L,U]				
U!TCONreq				req. aceitável
U?TDISind			não passou	
L?CR				
L!CC				
U?TCONresp				conexão estável
U?DATAreq				
U?TDISind			inconclusivo	
L?DT				
L!DT				
U?TDATAind				
U!TDISreq				
L?DR				
L!DC			passou	

Figure 6: Especificação em TTCN de um caso de teste para o protocolo de transporte

Na parte de restrições são especificados os valores de parâmetros de ASPs e campos de PDUs. Estas restrições são referenciadas na parte dinâmica para cada ASP ou PDU enviada ou recebida.

5 O TRADUTOR TTCN-ESTELLE

No processo de derivação de casos de teste executáveis adotado, os casos de teste abstratos devem inicialmente ser selecionados a partir das informações contidas no PICS. Em seguida, estes casos escolhidos são parametrizados pelos valores de parâmetros contidos no PIXIT. Estes casos de teste, já selecionados e parametrizados, são convertidos em casos de teste executáveis (Figura 7).

A tradução TTCN-Estelle foi desenvolvida a partir da forma TTCN processável por máquina, TTCN.MP, definida através de uma gramática BNF descrita em [8]. A entrada para o tradutor é um arquivo contendo todas as declarações, restrições e descrições de comportamento dinâmico de um caso de teste.

O tradutor foi implementado na linguagem C e sobre o sistema operacional XENIX. A tradução foi realizada em um único passo, englobando as seguintes fases: análise léxica, análise sintática, análise semântica e geração de código alvo. Durante a execução da tradução, a

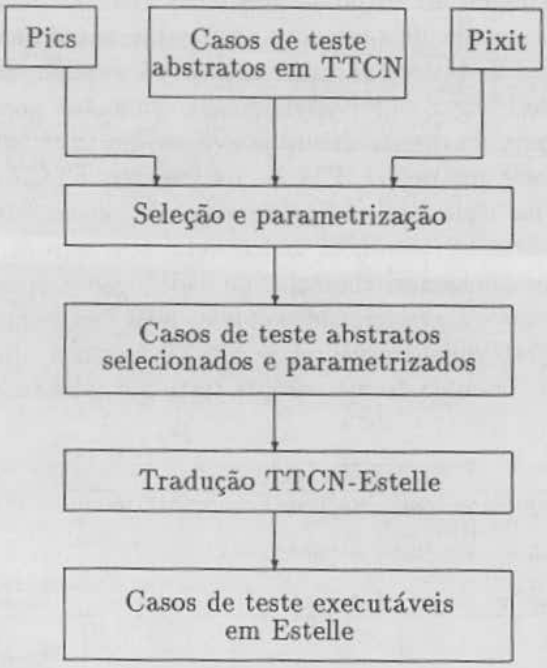

Figure 7: Derivação dos casos de teste executáveis

ocorrência de um erro em qualquer uma das fases, interrompe o processamento com uma mensagem indicando o tipo de erro e a sua localização na especificação do caso de teste.

A análise léxica é a fase mais básica da tradução, onde a especificação do caso de teste em TTCN é subdividida em seus constituintes elementares : identificadores, delimitadores, números, palavras chave, comentários,etc.

O analisador léxico foi desenvolvido utilizando a ferramenta LEX do XENIX [16]. A entrada do analisador léxico consiste de regras e fragmentos de programas associados a estas regras. Inicialmente uma especificação de um analisador léxico é preparada criando um programa na linguagem LEX, e este, após compilado, gera o analisador léxico que transforma os caracteres da entrada em seqüências de *tokens*.

Na análise sintática as estruturas da especificação, tais como declarações e expressões, são identificadas utilizando os *tokens* que foram gerados anteriormente.

A análise semântica verifica se a especificação em TTCN contém erros semânticos e reúne informações necessárias na fase subsequente.

Na fase de geração do código alvo, a especificação do caso de teste em Estelle é obtida a partir das rotinas escritas em C associadas às produções da gramática.

As fases de análise sintática, semântica e geração do código foram realizadas utilizando o *Parser Generator YACC (Yet Another Compiler-Compiler)* [16], também disponível no XENIX. Como a gramática TTCN é ambígua, inicialmente, através de regras de desambiguidade, todas as ambiguidades são eliminadas, anulando desta forma as ações conflitantes. Em seguida é preparado um arquivo contendo uma especificação YACC, com todas as regras das produções da gramática e as rotinas semânticas e de geração de código Estelle associadas. Este arquivo, depois de compilado, gera um programa que faz a tradução especificada.

A TRADUÇÃO

Neste projeto, um caso de teste em TTCN corresponde a um único módulo em Estelle. Não estão

sendo considerados comportamentos defaults e bibliotecas de passos de teste em TTCN, ou seja, não é permitida a utilização do comando **attach** que faz a ligação desses a um determinado caso de teste. Desta forma, as especificações dos casos de teste devem ser previamente expandidas.

A especificação obtida em Estelle é constituída pela definição do módulo e de sua interface com o ambiente externo, incluindo também, a definição dos canais e definição dos tipos de interações associadas a estes canais. Para cada um dos casos de teste de um determinado cenário de teste, corresponde a definição de um módulo. Entretanto, as interfaces para estes módulos são todas idênticas. A figura 8 ilustra esta correspondência. O código em Estelle gerado pelo tradutor tem a estrutura mostrada a seguir.

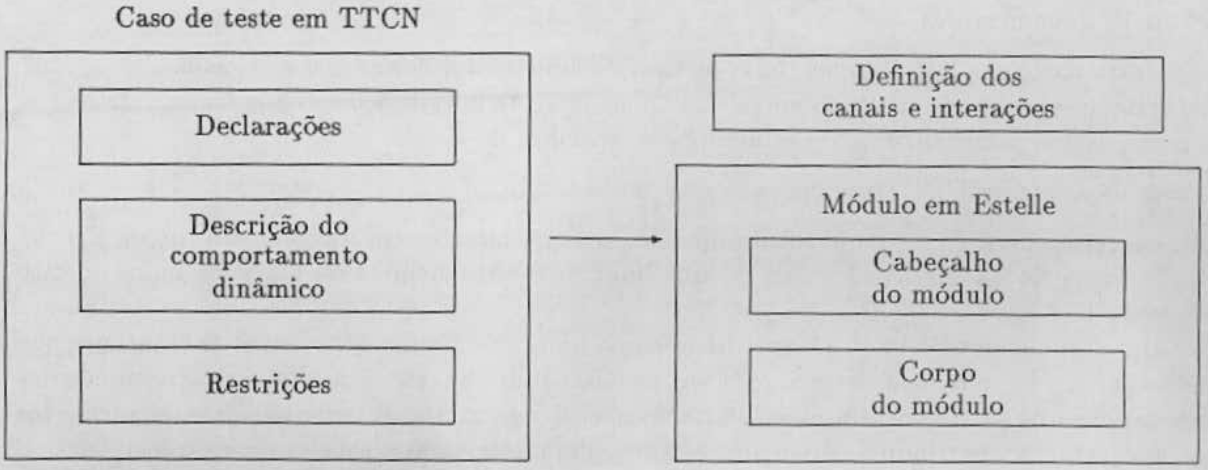

Figure 8: Correspondência entre caso de teste em TTCN e em Estelle.

Definição dos Canais e as Respectivas Interações

Os Pontos de Controle e Observação (PCOs) declarados para um cenário de teste em TTCN são os mesmos para todos os casos de teste. Como um caso de teste TTCN corresponde a um módulo em Estelle, estes PCOs são correspondentes aos pontos de interação que definem a interface do módulo. Para cada PCO de um caso de teste é definido um canal em Estelle que interliga pontos de interação entre módulos.

A partir das declarações TTCN de Unidades de Dados de Protocolo (PDUs) e Primitivas de Serviços Abstratas (ASPs), que constituem as mensagens trocadas entre PCOs, são definidos os tipos das interações que são enviadas e recebidas através dos canais. Para uma definição de canal são associados dois papéis opostos, **user** e **provider**, e é considerado o mesmo conjunto de interações para ambos.

Definição da Interface do Módulo

Em Estelle, a definição da interface do módulo consiste de um cabeçalho contendo o nome do módulo e os seus pontos de interação. O nome do módulo corresponde ao identificador do caso de teste em TTCN, e os pontos de interação são obtidos a partir das declarações de PCOs do caso de teste. A classe do módulo considerada é **process** e o tipo de fila de PCOs é **individual queue**.

Definição do Corpo do Módulo

Parte de Declarações

Nesta parte são declaradas todas as variáveis, constantes, estados, procedimentos e funções referenciadas na parte de declarações de transições do módulo. As variáveis e constantes são obtidas diretamente da parte de declarações da especificação em TTCN. Os estados são obtidos a partir da indentação dos eventos especificados na parte dinâmica do caso de teste. Os procedimentos e funções são necessários para lidar com as referências às restrições correspondentes aos eventos **send** e **receive** em TTCN, e são obtidos da parte de restrições do caso de teste. A sintaxe de todas as declarações é a mesma da linguagem Pascal, com exceção da sintaxe da variável estado (**state**) que é específica de Estelle.

Parte de Inicializações

Nesta parte são inicializadas todas as variáveis que foram declaradas e inicializadas no caso de teste, e especificado o estado inicial das transições. As inicializações têm a mesma sintaxe de Pascal, exceto a inicialização do estado que é específica de Estelle.

Parte de Declaração de Transições

Corresponde à parte principal do módulo, com declarações de todas as transições Estelle. Cada transição é declarada a partir de uma linha de evento definida na parte dinâmica do caso de teste TTCN.

Um caso de teste TTCN é estruturado como uma árvore. Assim, eventos alternativos, que estão em um mesmo nível nesta árvore, são irmãos. Todas as transições Estelle correspondentes a estes eventos partem de um mesmo estado especificado na cláusula **from**. Eventos sucessivos geram transições partindo de diferentes estados. Para dois eventos sucessivos, as transições chegam a diferentes estados especificados na cláusula **to**. Para a obtenção dos estados referenciados nestas cláusulas, é considerado o nível na árvore em que o evento se encontra e o nodo pai deste evento. O nível do evento na árvore é especificado como o nível de indentação da linha de evento no caso de teste.

Um evento **receive** em TTCN corresponde em Estelle à cláusula **when** especificada na parte de condições da transição. A ASP ou PDU associada ao **receive** é a interação especificada em **when** em um determinado ponto de interação. Uma referência a uma restrição para este evento é traduzida como uma cláusula **provided** onde a expressão booleana é uma função que retorna um valor booleano. Esta função verifica se a interação recebida está de acordo com as restrições estabelecidas no caso de teste. A presença de uma expressão booleana associada a este evento corresponde a uma expressão acrescentada à cláusula **provided** através de um **and**.

Um evento **send** em TTCN corresponde à expressão Estelle **output** especificada na parte de ações de uma transição. Uma referência a uma restrição para este evento é traduzida como uma chamada a um procedimento que faz atribuições de valores impostos aos campos/parâmetros das PDUs/ASPs a serem enviadas. Após estas atribuições, a interação está preparada para ser enviada e é referenciada em **output** no ponto de interação adequado.

Devido a não existência, em Estelle, de uma estrutura similar ao evento **otherwise** de TTCN, este foi traduzido como uma série de transições aninhadas. É especificada uma transição com a cláusula **when** para cada uma das interações que podem ser recebidas através do ponto de interação associado ao evento **otherwise**, e que não foram ainda relacionadas nos eventos **receive** alternativos. Estas transições têm todas as cláusulas de condições herdadas, exceto a cláusula **when** que é única para cada transição. A parte de ações das transições também são idênticas para todas elas.

O evento **elapse** em TTCN é traduzido como a cláusula **delay** em Estelle. O **elapse** fornece uma maneira da execução do caso de teste ciclar em torno de eventos alternativos até que um

destes eventos ocorra ou não, em um intervalo de tempo limitado. **Delay** provoca um atraso na execução de uma transição durante um intervalo de tempo entre um valor mínimo e máximo, sendo que, antes deste tempo expirar, qualquer outra transição pode ser selecionada se um evento ocorrer. A correspondência entre estas expressões é possível considerando-se as durações de tempo mínimo e máximo do **delay** idênticas à duração de tempo do **elapse**. Se não existir algum veredicto relacionado com o **elapse**, a parte de ações para esta transição é vazia.

Expressões de atribuições e expressões booleanas em TTCN, quando ocorrem sozinhas em linhas de evento (pseudo-eventos), correspondem a transições espontâneas em Estelle. A ocorrência de uma expressão booleana é traduzida como uma transição espontânea com uma cláusula **provided**, e com a parte de ações vazia. A ocorrência de uma expressão de atribuição gera uma transição espontânea com as expressões de atribuições relacionadas na parte de ações. Ambas expressões podem ocorrer simultaneamente em uma linha de evento.

O comando **goto** em TTCN é introduzido como uma transição com a parte de ações vazia. O *label* referenciado corresponde ao estado especificado na cláusula **to**.

A operação de temporização **start timer** é aplicada a um temporizador particular indicando que este deve começar a temporizar. Esta operação é traduzida para Estelle como o envio de uma interação START para um módulo de temporização através de um ponto de interação específico T. Para tal, é utilizado o comando **output** especificado na parte de ações da transição.

O evento **timeout** é aplicado para verificar a expiração de um temporizador específico. A sua tradução para Estelle corresponde ao recebimento de uma interação TIMEOUT no ponto de interação T, especificada na cláusula **when**.

O módulo de temporização deve ser especificado à parte, sendo responsável pelas operações de temporização. Ele deve estar preparado para receber as interações START e enviar as interações TIMEOUT para o caso de teste. Para atingir estes objetivos é acrescentado ao módulo do caso de teste o ponto de interação T, para possibilitar a sua comunicação com este módulo de temporização. Para este ponto de interação é definido um canal com as interações START e TIMEOUT.

A operação **cancel timer** é traduzida da mesma forma que a operação **start timer**. Uma interação CANCEL para um temporizador específico é enviada para o módulo temporizador através do comando **output**.

Devido ao limite de tempo para a execução do tradutor, e como a notação TTCN é bem extensa, algumas restrições foram feitas com relação à esta. Não foram traduzidos o comando **attach**, passos de teste, comportamentos *default*, a operação **read timer** e as declarações na notação ASN.1. Consequentemente, não foi traduzido o comando **repeat** que depende diretamente do **attach**.

TTCN está em uma norma cujo *status* é DIS ("Draft International Standard"), que é a última etapa antes de se tornar IS ("International Standard"). Por esta razão, alguns itens na norma ainda não estão muito bem esclarecidos e, portanto, não foram levados em consideração. Estes correspondem ao evento **implicit send**, expressões **encode** e **decode** e tipos e operações definidos pelo usuário.

O exemplo a seguir ilustra as traduções das linhas de eventos já relacionadas. Este é um trecho do caso de teste DL1-101 do cenário de teste de conformidade X25-DTE para a camada de enlace de dados [19]. Para tornar mais claro o exemplo, a notação do caso de teste utilizada é a TTCN.GR.

Em TTCN :

Descrição de comportamento	Label	Rest.	Resultados	Comentários
L!DISC (p:=1) ... L?DM [f:=1] L!RR(p:=1,N_R:=0) L?DM [f:=1] L?OTHERWISE ELAPSE 10 ...		DISC(S31) DM(S31) RR(S31) DM(S31)	 passou nao passou passou	

Em Estelle :

```
{MODULE HEADER DEFINITION}

module DL1_101_type process ;
    ip
        L : L_acess_point (provider) individual queue ;
        T : L_acess_point (user) individual queue ;
    end ;
{ MODULE BODY }
body DL1_101_body for DL1_101_type ;
{ DECLARATION PART }
{CONSTANT DECLARATION}
...
{ VARIABLE DECLARATION}
...
{ INITIALIZATION PART }
initialize
    to state100
{ TRANSITION DECLARATION PART }
trans
    from state100
        to state200
            begin
                p := 1 ;
                Assignment_ConstraintDISC1(xDISC,p);
                output L.DISC(xDISC) ;
            end ;
        ...
    from state400
        to state500
            when L.DM
            provided ConstraintDM1_Ok(xDM,1)
                begin
                end ;
    from state500
        to state600
            begin
                p := 1 ;
                N_R := 0 ;
                Assignment_ConstraintRR1(xRR,p,N_R);
                output L.RR(xRR) ;
            end ;
```

```
    from state600
        to state700
            when L.DM
            provided ConstraintDM1_Ok(xDM,1)
            begin
                verdict := PASS ;
            end ;
    from state600
        to state701
            when L.RR
            begin
                verdict := FAIL ;
            end ;
            when L.UA{  Transições aninhadas correspondentes
            begin       ao evento otherwise }
                verdict := FAIL ;
            end ;
            ...
            when L.DISC
            begin
                verdict := FAIL ;
            end ;
    from state600
        to state702
            delay (10,10)
            begin
                verdict := PASS ;
            end ;
    ...
end; {body}
```

6 CONCLUSÕES

Este artigo descreveu um tradutor de casos de teste abstratos, escritos na notação TTCN.MP, para casos de teste executáveis, escritos na FDT Estelle. Estes, por sua vez, podem ser utilizados na realização automática dos testes de implementações de protocolos.

A versão atual do tradutor encontra-se em fase de testes. Para testar a implementação está sendo utilizado o cenário de teste de conformidade X25-DTE para a camada de enlace de dados.

Os testes dos códigos Estelle gerados pelo tradutor estão sendo executados por meio do compilador Estelle-C citado anteriormente. Para que estes se tornem possíveis, é especificado um módulo protocolo em Estelle para enviar e receber as interações do módulo caso de teste, um módulo temporizador e um módulo "Specification". Veja a figura 9.

Um problema para a realização dos testes é que estes estão padronizados na notação TTCN.GR e, portanto, devem ser transcritos inicialmente para a notação TTCN.MP.

Um trabalho já iniciado relativo a este tópico consiste na construção de um pré-processador de TTCN para preparar uma especificação para ser traduzida, cujas maiores tarefas serão a ligação de sub-árvores (comando **attach**) e a anexação dos comportamentos *defaults*.

Outro trabalho em estado inicial é a construção de um editor TTCN, para facilitar a edição de em modo gráfico ou em modo processável por máquina, bem como a conversão automática para um modo a partir do outro.

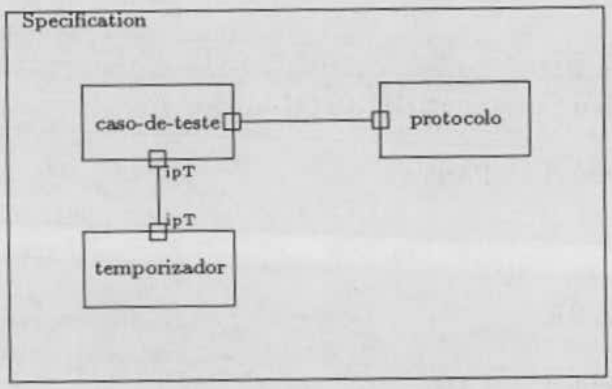

Figure 9: Esquema para teste da tradução

Ainda em fase de estudos está a geração automática de especificações de teste em TTCN, no sentido de se construir um ambiente automatizado de teste de conformidade de protocolos.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Sarikaya B., "Protocol Testing: Architectures and Test Sequences", Concordia University, Montreal, September 1987, 35 pg.

[2] Budkowski, S., Dembinski, P., "An Introduction to Estelle: A Specification Language for Distributed Systems", Computer Networks and ISDN Systems, 14(1987), pg. 3-23.

[3] Linn, R.J., "The Features and Facilities of Estelle", Proc. IFIP Wg 6.1 5th International Workshop on Protocols, 1986, pg. 271-296.

[4] Rayner D., "OSI Conformance Testing",Computer Networks and ISDN Systems , (14) 1987, pg. 79-98.

[5] ISO/TC 97/SC 21, "OSI Conformance Testing Methodology and Framework - Parte 1:General Concepts",Vancouver, December 1987.

[6] Cerny, E., Bochmann, G.V., Maksud, M., Léiveillé A., Serre, J.M., Sarikaya, B., "Experiments in Testing Communication Protocol Implementations", IEEE Proc. 14th FTCS, November 1983, pg.204-209.

[7] Sarikaya, B., Bochamnn, G.V., Maksud, M., Serre, J., "Formal Specifications Based Conformance Testing", SIGCOMM'86 Communication Architectures & Protocols, vol. 16, no. 3, Vermont, August 1986, pg. 236-240.

[8] ISO/IEC JTC1/SC21, "The Tree and Tabular Combined Notation (TTCN) ", September 1988.

[9] "The Ferry Clip - A Generalized Application of Ferry Principle", University of British Columbia, IDACOM Electronics Ltd., July 1988, 21 pg.

[10] Rafiq, O., Chraibi, C., Castanet, R., "Experimental Testing of Transport Protocol", Computer Communication Review,vol.16, no. 4, July/August 1986, pg. 23-33.

[11] Greco, C.E., Bion, E.C.B., Gerber, G.W., Filho, L.A., "Teste de Implementações de Protocolos e Comunicação - Uma Técnica Posta em Prática", XIV Seminário Integrado de Software e Hardware - VII Congresso da SBC Salvador, Julho 1987, pg. 602-611.

[12] Ural, H., Probert R.L., "Architectures for Testing Protocol Implementations", University of Ottawa, TR-83-13,21 pp, Ottawa, Canadá, 1983. 327

[13] Rafiq O., "A Good Approach for Testing Protocol Implementations",Proc. of the International Seminar on Computer Networking, pp 305-314, Japan, 1986.

[14] Chan, R.I.M.H., "An Estelle-C Compiler for Automatic Protocol Implementation", Technical Report 87-36, University of British Columbia, Vancouver, Canadá, November 1987.,

[15] Johnson,S.C., "YACC - Yet Another Compiler-Compiler", Comp. Sci.Tech. Rep. No. 32, Bell Laboratories (July 1975).

[16] Lesk, M.E., "LEX - A Lexical Analyzer Generator",Comp. Sci. Tech. Rep. No. 39, Bell Laboratories (October 1975).

[17] Sarikaya, B., Van Houte P., Berriman T., Eswara S., "Towards Execution of TTCN Test Cases", Tenth International IFIP WG 6.1 Symposium on Protocol Specification, Testing and Verification", June 1990, pg. 12-15.

[18] Bush, M., Rasmussen, K., Wong F., "Conformance Testing Methodologies for OSI Protocols", AT&T Technical Journal, January/February 1990, pg. 07-16.

[19] X.25-DTE Conformance Testing : Revised Text for Data Link Layer Test Suite for DP8882 Part 2, ISO JTC1 SC6 WG1, April 1988.