TÍTULO: AMBIENTE DE TESTE PARA PROTOCOLOS DE COMUNICAÇÃO

AUTORES: Wanderley LOPES DE SOUZA
GRC/DSC/Universidade Federal da Paraíba
(trabalho realizado junto ao IRO/Université de Montréal,
com auxílio fornecido pelo CNPq)

Rachida DSSOULI
IRO/Université de Montréal

Gregor von BOCHMANN
IRO/Université de Montréal

SUMÁRIO: A fim de que os Testes de Serviços e Protocolos de Comunicação sejam viáveis, é necessário que os sitemas que realizam tais testes, ofereçam aos seus usuários, acima de qualquer outra característica, a Confiabilidade.

Neste trabalho, é proposta uma arquitetura para um Sistema de Teste, composta basicamente de três componentes: uma Referência para o Teste, um Gerador de vetores de teste e um Reconhecedor de sequências de interações.

As vantagens e o incoveniente maior, da utilização de uma linguagem baseada em lógica, no caso Prolog, para implementação deste Sistema, são discutidos.

1. INTRODUÇÃO

Nas diferentes fases de desenvolvimento de PROTOCOLOS de Comunicação, os conceitos de **VERIFICAÇÃO, VALIDAÇÃO e TESTE**, são frequentemente empregados. Embora utilizados em momentos diferentes e com objetivos diferentes, estas noções incorporam um mesmo tipo de atividade, isto é, testes.

No contexto da "Open System Interconnection (OSI)", a palavra Teste, indica o conjunto de operações realizadas, para verificar se uma **IMPLEMENTAÇÃO DE UM PROTOCOLO SOB TESTE** ("Implementation Under Test - **IUT**"), esta de acordo com a **ESPECIFICAÇÃO** deste protocolo. O objetivo final do Teste, é a obtenção de uma **CERTIDÃO DE CONFORMIDADE** ("Assessment") para a implementação.

Devido à natureza complexa do "software" de comunicação e a fim de facilitar a tarefa de definição dos protocolos, a Especificação é dividida em:

(a) **SERVIÇO** a fornecer ao nível N+1 (N+1_nível);

(b) **PROTOCOLO** do N_nível e
(c) **INTERFACES** do N_nível. [BoSu 80]

As interfaces existem para facilitar o acesso aos níveis vizinhos. Embora muitas vezes não sejam especificadas durante a descrição de um dado nível, para a Implementação elas são uma realidade e como as demais partes, fontes incontestáveis de erros (Fig.1).

Uma vez realizada a divisão da Especificação, é natural que procedimentos análogos, sejam adotados durante as demais fases de desenvolvimento de um protocolo. Assim sendo, para a fase de Teste, tem-se:

(a) o Teste do N_Serviço, fornecido por uma N_IUT. Tal teste, consiste em verificar se a N_IUT, é capaz de oferecer os serviços previstos na N_Especificaçao do Serviço. Isto é, consiste em verificar se as ações que são visíveis aos usuários e são desencadeadas pela N_IUT, quando sujeita a determinados eventos, **N_PRIMITIVAS DE SERVIÇO**, estão previstas na N_Especificação do Serviço padronizada e
(b) o Teste da N_IUT. Tal teste, consiste em verificar se a N_IUT segue a N_Especificação do Protocolo. Isto é, consiste em verificar se as ações desencadeadas pela N_IUT, quando sujeita a determinados eventos, N_Primitivas de Serviço e **N_UNIDADES DE DADO DE PROTOCOLO** ("Protocol Data Units - **PDU**"), estão previstas na N_Especificação do Protocolo padronizada. [PaFoMo 83]

Tendo em vista o ambiente frequentemente distribuido, no qual os protocolos de comunicação são empregados, a concepção de um **SISTEMA DE TESTE** para serviços e protocolos, deve atentar para:

(a) a realidade da IUT. Por exemplo, os postos de observação disponíveis, as facilidades de detecção e localização de erros, etc...;
(b) a definição clara dos objetivos a serem atingidos com o Teste, o que implica na escolha de um **METODO DE TESTE e das SEQUÊNCIAS DE TESTE** a serem aplicadas a IUT e
(c) a escolha das ferramentas (e.g, linguagem para a implementação) a serem utilizadas pelo Sistema de Teste.

2. DEFINIÇÃO DO AMBIENTE DE TESTE (AMBT)

Um Sistema de Teste de protocolos, será utilizado pelos clientes interessados na obtenção de Certidões de Conformidade, para seus produtos. Portanto, a viabilidade econômica de desenvolvimento de um tal sistema, está sujeita a:

(a) confiabilidade, pois caso o sistema seja defeituoso, a IUT poderá ser considerada conforme a especificação normalizada, mesmo não o sendo;
(b) modularidade, pois em função da organização hierárquica dos protocolos, é interessante dispor-se de um sistema composto de módulos, que correspondam aos níveis dos protocolos. Por exemplo, o módulo utilizado para certificar o nível Rede OSI, poderá fornecer serviços para

o módulo que certificará o nível Transporte. Desta forma, pode-se atingir um sistema de teste, capaz de certificar toda uma arquiteura;
(c) versatilidade, isto é, um sistema capaz de certificar um grande número de impleméntações, utilizando diferentes métodos de teste;
(d) transportabilidade, possibilitando assim a adaptação do sistema, aos diferentes contextos impostos pelas IUT's;
(e) 'informatividade', isto é, além de certificar implementações, o sistema deve ser capaz, no caso de falha, de fornecer informações precisas sobre as causas desta falha e
(f) automatização do sistema. [Harv 83]

De acordo com as arquiteturas propostas para Teste, uma IUT localizada num Sítio Cliente, é testada por um Sítio Fornecedor de Certidões, através de um meio de comunicação interligando os dois sítios. Além da IUT, que é testada como uma 'caixa preta', existe no primeiro sítio, um 'Interlocutor de Teste' ("Test Responder - TR"), que age como o usuário dos serviços fornecidos pela IUT. A configuração do segundo sítio, é basicamente constituida de um Testador Ativo ("Active Tester - AT"), responsável pela geração e controle de sequências de teste (ou vetores de teste). O Teste da IUT, é realizado através da observação das sequências de interações entre o AT e o TR, resultantes da aplicação dos vetores de teste. [Rayn 82]

AMBIENTE DE TESTE, é um Sistema de Teste, equivalente ao Testador Ativo, composto de um conjunto de módulos, capazes de realizar um Teste completo de de uma implementação. Tal sistema, possue as seguintes atribuições:
(a) geração e controle dos vetores de teste, a que estará sujeita a IUT;
(b) observação do comportamento da IUT, em resposta aos vetores de teste a ela aplicados;
(c) análise dos resultados obtidos e
(d) conclusão sobre o bom ou mal funcionamento da IUT.

O Ambiente de Teste (**AMBT**), pode ser definido como um processo, que dispõe, como entradas, de uma **DESCRIÇÃO FORMAL** (**DF**) e de um Método de Teste (**MT**) e fornece, como saídas, um **DIAGNÓSTICO** e uma 'história' da execução do Teste, chamada **TRAÇO** (Fig.2).

A Descrição Formal, representa a especificação completa ou parcial da IUT. Em ambos os casos, ela deve permitir a geração dos vetores de teste, assim como o reconhecimento das interações provenientes da IUT.

Uma vez que os vetores de teste são gerados a partir da Descrição Formal e sendo a DF uma derivação da especificação da IUT, é natural que o método utizado para tal especificação, influencie no Método de Teste a ser escolhido. Por exemplo, caso a DF seja modelada através de uma Máquina de Estado Finito, três possíveis Métodos de Teste para o AMBT são: "Transitions-Tour", "Checking-Sequence" e "W-Method". [SaBo 82]

O Diagnóstico deve ser o mais claro possível. Duas alternativas existem:

(a) caso não sejam verificadas anomalias, a Implementação será conforme à Descrição Formal, para o conjunto de testes aplicados ou
(b) sendo verificado pelo menos um erro, a IUT será declarada não conforme a DF, pelas seguintes razões:
    - tipo de erro detectado e
    - localização do erro, se possível.

Independentemente do tipo de diagnóstico fornecido, existe sempre um Traço indicando os vetores de teste aplicados, assim como um Traço indicando o conjunto completo de interações realizadas, caso o Protocolo esteja sendo testado, ou o conjunto de ações observáveis pelo usuário, caso o Serviço esteja sendo testado.

Dois modos de Teste são possíveis para este Ambiente:

(a) passivo, onde o sistema desempenha o papel de um observador, que recolhe informações, analísa-as e fornece um Diagnóstico e um Traço ou
(b) ativo, onde os próximos vetores de teste a serem aplicados, serão escolhidos em função do que se observa. Neste caso, o AMBT terá decisões a tomar, tais como:
    - se os testes devem continuar ou não;
    - em que ordem os próximos vetores de teste devem ser aplicados e
    - etc...

3. MÓDULOS QUE COMPÕEM O AMBIENTE DE TESTE

O primeiro elemento a ser definido no AMBT, é a Descrição Formal. A DF é considerada o componente de referência (análoga ao circuito de referência em "hardware"). Se tanto a DF quanto a IUT são submetidàs às mesmas sequências de teste, tódas as respostas provenientes da IUT, devem ser consequências de comportamentos descritos na DF (Fig.3). A DF pode ser derivada através:

(a) das Especificações da IUT, se estas descrições são realizadas numa linguagem executável, tal que permita a geração de vetores de teste e a observação do comportamento da IUT ou
(b) de uma parte das Especificações da IUT, no caso de desejar-se verificar um tipo particular de comportamento, o qual deve ser definido previamente (e.g., Trajetória). [ProUra 83]

Caso as Especificações da IUT estejam numa linguagem de difícil manipulação, ou seja, que não permita a derivação dos vetores de teste, pode-se transformar tal descrição, num conjunto de ações e relações, as quais passarão a ser as referências para a análise.

A transformação de uma Descrição Formal numa **FORMA NORMAL ANALISÁVEL (FNA)**, pode ser feita:

(a) manualmente, onde uma FNA (árvore,autômato,etc...) pode ser escrita numa linguagem de altíssimo nível (e.g., **PROLOG**). O inconveniente deste modo, é a demonstração da equivalência funcional entre a DF e a FNA ou

(b) automaticamente, através de um tradutor que aceite uma DF e forneça uma FNA (Fig.4).

O segundo módulo a compor o Ambiente de Teste, é o **GERADOR** de vetores de teste. Para a especificação deste módulo, é necessario antes definir os Métodos de Teste aceitáveis e os tipos de testes a serem realizados.

O Teste Lógico, por permitir verificar se a IUT realiza bem as suas funções, é o que melhor se adapta aos protocolos de comunicação. Tais testes podem ser:

(a) exaustivos, se possível;

(b) guiados, onde as informações de direcionamento, podem ser provenientes de um módulo **OBSERVADOR** ou

(c) parciais, seja segundo o tipo de erro a ser detectado, ou segundo o caminho a ser explorado.

No caso de realizar-se testes guiados, o Ambiente de Teste é dito ativo. Para os demais casos, o AMBT é passivo.

Tendo em vista a maior generalização dàs DF's em relaçao às IUT's, isto é, uma implementação geralmente não comporta toda a especificação correspondente, falhas durante determinados testes, podem não implicar na não conformidade da IUT (e.g., opções presentes na especificação e que não são atendidas pelo protocolo). Para evitar este problema, um teste de sondagem deve ser realizado, a fim de delimitar as possibilidades da IUT em relação a DF.

O último módulo presente no AMBT representado na Fig.5, é o Observador. Baseando-se na DF (ou FNA) e nos vetores de teste gerados, este modulo é responsavel pelo reconhecimento das interações provenientes da IUT. Ele realiza a análise de coerência e detecta os erros que permitem alcancar ou não a conformidade. Neste último caso, o Observador deve fornecer informações relativas à natureza dos erros detectados e, se possível, a sua localização.

A título de exemplificação, um comportamento a ser observado de uma IUT, que implementa o Protocolo de Transporte classe 0 (nível de Transporte OSI), é descrito através de uma Máquina de Estado Finito (Fig.6). A mesma técnica, é utilizada na Especifição do Protocolo de Transporte, que sera a referência (DF) para a realização dos testes

(Fig.7). [Boch 83]

Em função do Método de Teste escolhido, em função das sequências de teste a serem aplicadas e baseando-se na DF e nas sequências de PDU's provenientes da IUT, os seguintes tipos de erros podem ser detectados:

(a) erros de transferência, envolvendo a função de transição de um autômato;
(b) erros de saída, envolvendo a função de saída de um autômato e
(c) erros envolvendo o número de estados, diferença entre os estados da máquina da DF e da máquina da IUT. [AnRaCaGu 84]

Numa dada sequência de teste, para toda PDU proveniente da IUT, deve ser verificado, se ela é uma PDU de saída válida, em relação ao seu suposto estado, e se ela é uma PDU de entrada válida, em relação a esta sequência de teste. Por exemplo, no comportamento a ser observado, se após o envio pelo Gerador da primitiva CR, a primeira PDU recebida da IUT for DT, o Observador deverá declarar a IUT não conforme. De acordo com a DF, o estado suposto da IUT e Wait_for_TCONresp e neste estado, há somente duas PDU's de saída válidas, isto é, CC e DR.

4. A LINGUAGEM DE IMPLEMENTAÇÃO DO AMBIENTE DE TESTE

O Ambiente de Teste, conjuntamente com a IUT e o módulo simulador do usuário, constituem uma espécie de sistema conversacional. Ao AMBT, é atribuida a tarefa de responder às seguintes questões:

(a) quais são as possíveis sequências de interações, permitidas pela Especificação Normalizada do Serviço (ou Protocolo) correspondente a IUT ? e
(b) é uma dada sequência de interações, proveniente da IUT, permitida pela Especificação de Serviço (ou Protocolo) correspondente ?

O módulo Gerador do AMBT, na medida em que produz, a partir da Forma Normal Analisável do Serviço (ou Protocolo), os vetores de interação, responde à primeira pergunta. Por outro lado, o módulo Obervador, na medida em que reconhece, baseado na FNA do Serviço (ou Protocolo), as sequências de interações provenientes da IUT, responde à segunda questão.

Uma vez que estes módulos, são responsáveis pela geração e reconhecimento de sequências válidas de interações, seria interessante, que as suas implementações fossem realizadas numa linguagem, que permitisse uma fácil derivação, a partir da FNA, destas funções. Tal derivação, seria ainda facilitada, caso a própria FNA fosse implementada nesta linguagem.

O uso de técnicas de programação em lógica, pode ser util à especificação, geração e reconhecimento das sequências válidas de interações, que ocorrem entre duas entidades que implementam o mesmo protocolo. Tais técnicas, permitem a definição de um programa, em duas distintas partes:

(a) lógica, que expressa o conhecimento necessário para a resolução do problema e
(b) controle, que expressa como utilizar este conhecimento para a resolução do problema.

A linguagem de programação **PROLOG**, é baseada na filosofia de programação em lógica e na 'interpretação procedural de Kowalski para cláusulas 'Horn'. Tais cláusulas, são do tipo:

(a) H :- B1,B2,.....,Bm. ou
(b) H .

onde H e Bi's são metas primitivas. A interpretação da primeira cláusula é: a cabeça (H) da cláusula é verdadeira, se o corpo (B1,B2,...,Bm.) também o for. No segundo tipo de cláusula, H é sempre verdadeira.

Um programa em Prolog, é constituido de um conjunto de cláusulas Horn. A execução deste programa, corresponde à construção de uma prova, ou sucessão de deduções, para uma dada meta global (?- A1,A2,...,An.). Tal prova, é realizada através de uma pesquisa sequêncial, associada a **"BACKTRACKING"**, visando a unificação das submetas Ai's às cláusulas do programa. As ordens, nas quais as submetas e as cláusulas são escolhidas, são da esquerda para a direita e de cima para baixo, respectivamente. A execução pode ter dois resultados: sucesso ou falha. No primeiro caso, os valores dos parâmetros, presentes na meta global, constituem a saída da execução. [CloMel 81]

Muitas Descrições Formais de Serviços e Protocolos, tem sido realizadas, utilizando-se a noção de Máquina de Estado Finito Extendida (EFSM). As extensões, introduzidas através de variáveis de estado, permitem a inclusão e o armanezamento de informações relativas ao Serviço (ou Protocolo). O estado de uma máquina, pode mudar em função da execução de uma transição (entrada, saída ou espontânea).

As DF's baseadas em EFSM, podem ser codificadas em Prolog, atrvés da definição de um conjunto de cláusulas, onde cada cláusula maior corresponde a uma transição da EFSM. Esta definição lógica, corresponde ao conhecimento necessário da Especificação do Serviço (ou Protocolo), para a geração e o recohnecimento das sequências válidas de interações. Isto é, corresponde à Forma Normal Analisável do AMBT.

A geração e o reconhecimento de sequências válidas de interações, podem ser implementadas através de dois conjuntos distintos de claúsulas de controle, ou de um único conjunto, no caso de desejar-se um

programa reversível (gerador e reconhecedor ao mesmo tempo). Tais cláusulas, associadas às que já foram definidas para a FNA, podem realizar as tarefas atribuídas ao AMBT.

O fato de Prolog utilizar backtracking, durante as tentativas de unificação de submetas e cláusulas, é positivo, por um lado, pois permite a geração e o reconhecimento exaustivos das sequências de interações. Por outro lado, na medida em que se deseja um certo desempenho para o AMBT, esta mesma característica torna-se incoveniente, pois a forma atual de utilização de backtracking por Prolog, não é eficiente.

5. CONCLUSÃO

O Teste, sendo a última fase de desenvolvimento de um Protocolo de Comunicação, tem por objetivo o fornecimento de uma Certidão de Conformidade da IUT, em relação às Especificaçoes de Serviço e Protocolo que ela implementa.

O Ambiente de Teste (AMBT), é um sistema responsável pelo Teste completo de uma IUT. Localizado num Sítio Fornecedor de Certidões, ele se comunica com a IUT, localizada num Sítio Cliente, através de um meio de comunicação. Tal sistema recebe, como entradas, uma Descrição Formal (DF) e um Método de Teste(MT) e fornece, como saídas, um Diagnóstico e um Traço.

O AMBT, é basicamente constituido de uma Forma Normal Analisável (FNA), que é a referência para o Teste, de um módulo Gerador de vetores de teste e de um Observador, responsável pelo reconhecimento das interações provenientes da IUT. Para que tal sistem seja viável, é necessário atentar para requisitos, tais como: confiabilidade, modularidade, versatilidade, transportabilidade, 'informatividade' e automatização.

A implementação do AMBT em Prolog, permite a divisão do programa em dois segmentos: o lógico, onde é especificada a referência (FNA) para a geração e o reconhecimento de sequências válidas de interações e o segmento de controle, onde são definidas as formas de utilização da FNA, para a geração e o reconhecimento destas sequências (Gerador e Observador).

A utilização de backtracking pela linguagem Prolog, é uma outra característica importante, pois permite a geração e o reconhecimento exaustivos de sequências válidas de interações, o que acarreta numa maior credibilidade para o AMBT. No entanto, esta mesma característica, da forma como é empregada atualmente pelos interpretadores de

Prolog, torna-se indesejável, quando o AMBT é observado sob o ponto de vista de Desempenho.

6. REFERÊNCIAS

[AnRaCaGu 84] : J.P. Ansart, O. Rafiq, R. Castanet, P. Guitton, "Some Operational Tools in a OSI Protocols Study Environment", ACM SigCOMM'84 Communication Architectures & Protocols, vol 4, No 2, Jun 84, pp 156-161.

[Boch 83] : G.V. Bochmann, "Formal Description Techniques for OSI: an Example", publicação interna No 493, IRO/Université de Montréal, Nov 83, apresentada no INFOCOM'84.

[BoSu 80] : G.V. Bochmann, C.A. Sunshine, "Formal Methods in Communication Protocol Design", IEEE Trans., COM-28, No 4, Abr 80, pp 624-631.

[CloMel 81] : W.F. Clocksin, C.S. Mellish, "Programming in Prolog", livro editado por Springer-Verlag Berlin Heidelberg, 81.

[Harv 83] : G.A. Harvey, "The Routing Certification System", anais do "Protocol, Specification, Testing and Verification, III", editado por H. Rudin e C.H. West, North-Holland, 83, pp 465-476.

[Kowa 79] : R. Kowalski, "Algorithm = Logic + Control", Comm. ACM, Vol 22, No 7, Jul 79, pp 424-436.

[PaFoMo 83] : S. Palazzo, P. Fogliata, G. Le Moli, "A Layer-Independent Architecture for a Testing System of Protocol Implementations", anais do "Protocol, Specification, Testing and Verification, III", editado por H. Rudin e C.H. West, North-Holland, 83, pp 393-406.

[ProUra 83] : R.L. Probert, H. Ural, "Requirements for a Test Specification Language for Protocol Implementation Testing", anais do "Protocol, Specification, Testing and Verification, III", editado por H. Rudin e C.H. West, North-Holland, 83, pp 437-443.

[Rayn 82] : D. Rayner, "A System for Testing Protocol Implementations", anais do "Protocol, Specification, Testing and Verification, II", editado por C. Sunshine, North-Holland, 82, pp 539-553.

[SaBo 82] : B. Sarikaya, G.V. Bochmann, "Some Experience with Test Sequence Generation for Protocols", anais do "Protocol, Specification, Testing and Verification, II", 82, editado por C. Sunshine, North- Holland, 82, pp. 555-567.

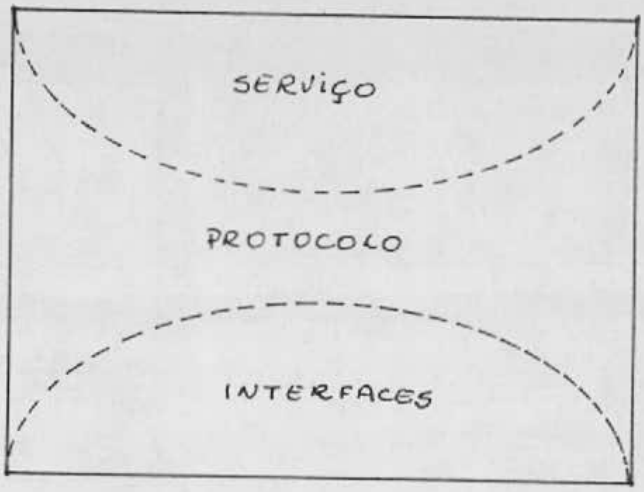

Fig. 1 FONTES DE ERROS NUMA IMPLEMENTAÇÃO

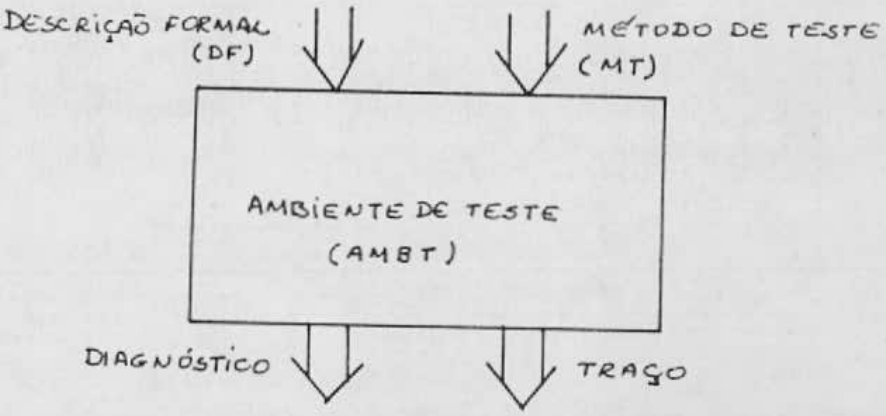

Fig. 2 DEFINIÇÃO DE UM AMBIENTE DE TESTE

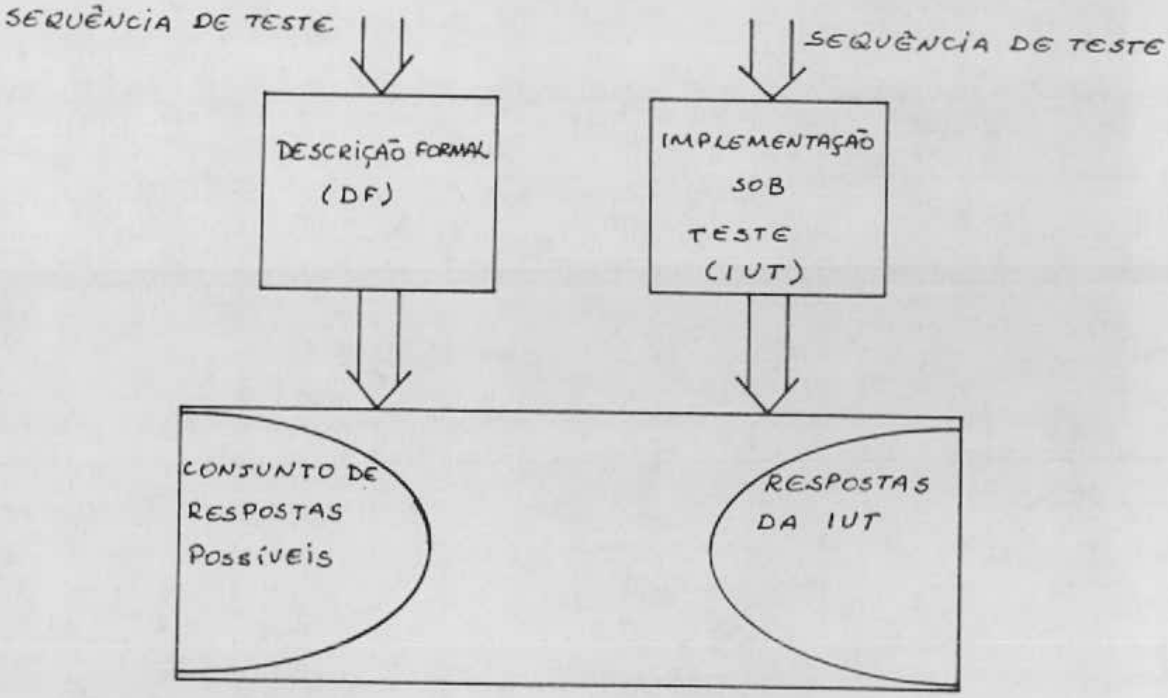

Fig. 3 INTERAÇÕES ENTRE A DF E A IUT

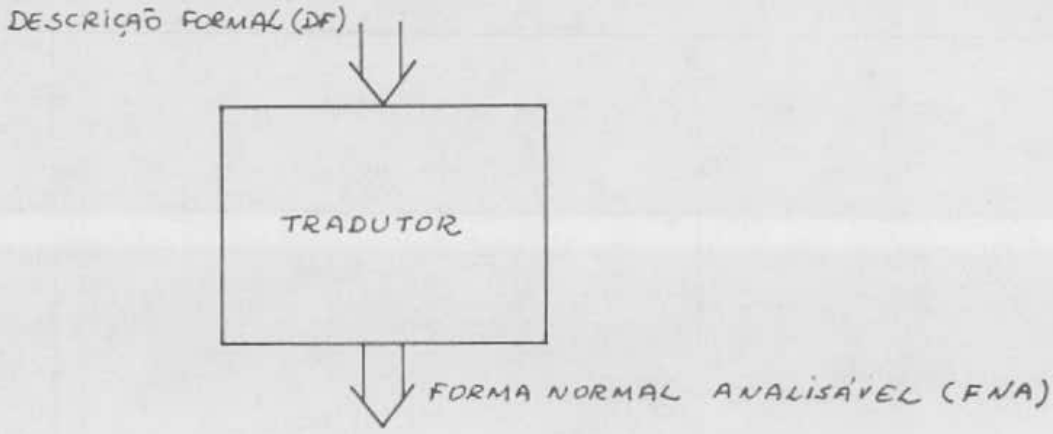

Fig 4 TRADUÇÃO DE UMA DF NUMA FNA

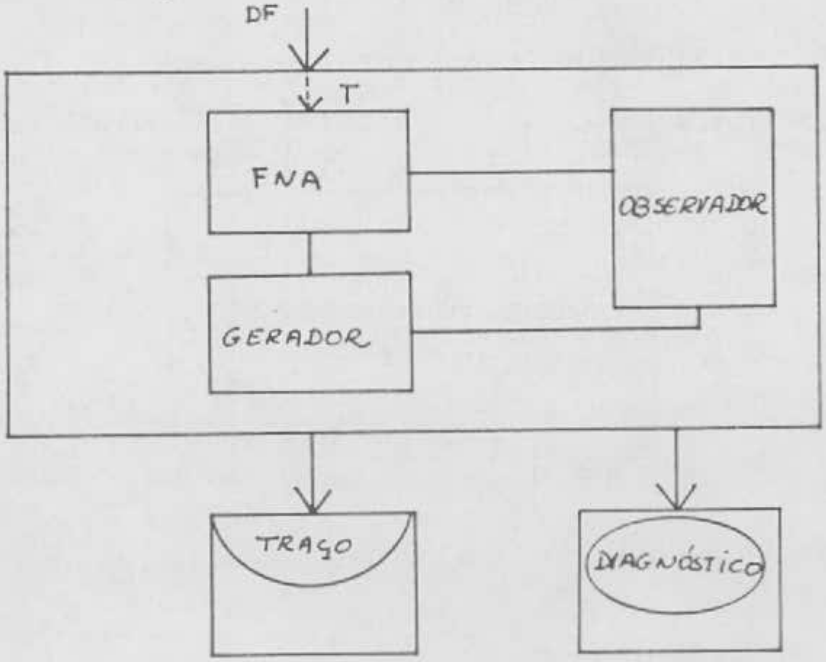

Fig 5 MÓDULOS DE UM AMBIENTE DE TESTE

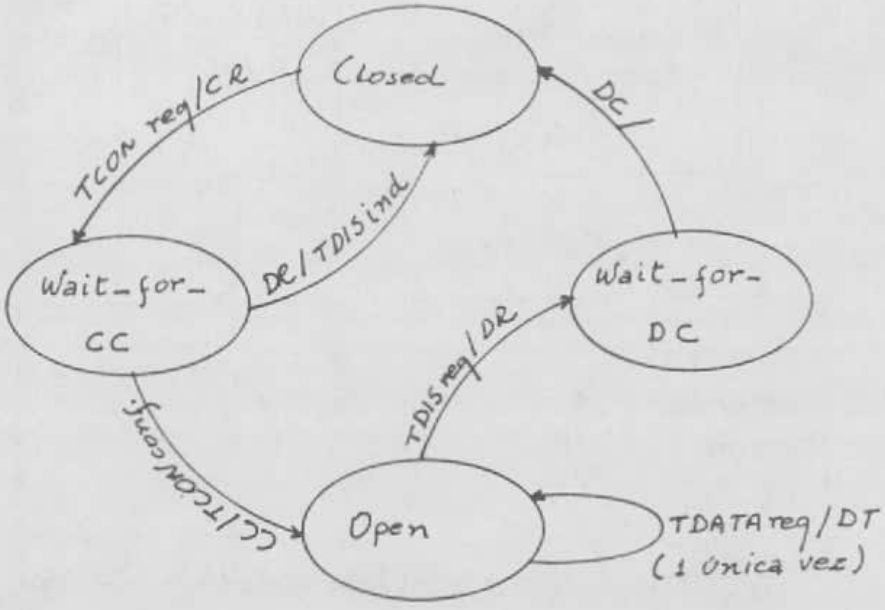

Fig 6 EX. DE UM COMPORTAMENTO DO PROTOCOLO DE TRANSPORTE
(notação: "Entrada/saída")

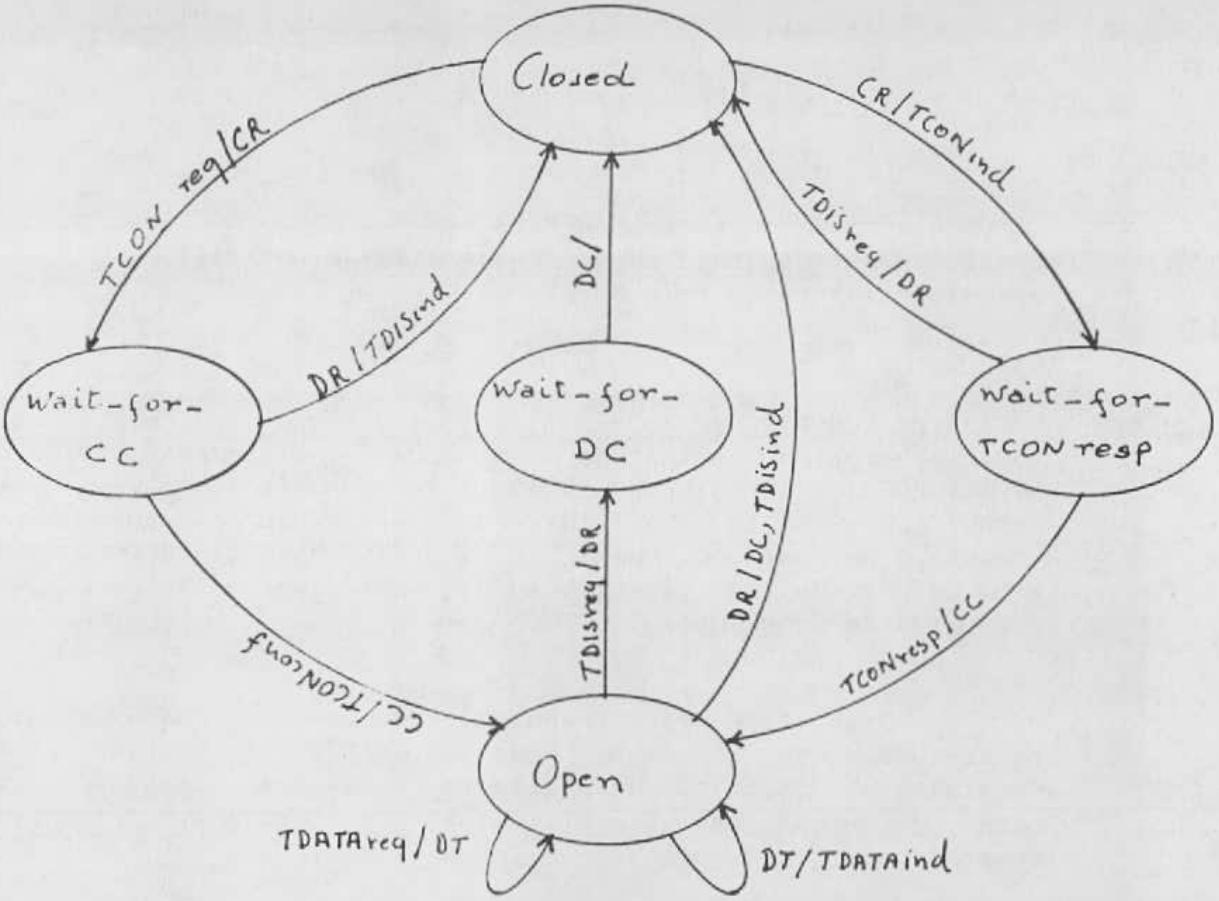

Fig. 7 MÁQUINA DE ESTADO FINITO PARA O PROTOCOLO DE TRANSPORTE (notação: "Entrada / saída")