ANÁLISE DE MUTANTES BASEADA EM MÁQUINAS DE ESTADO FINITO

Sandra Camargo Pinto Ferraz Fabbri[1]
José Carlos Maldonado[2]
Paulo Cesar Masiero[2]
Marcio Eduardo Delamaro[3]
Karen Kiomi Nakazato[3]

1 Doutoranda do IFQSC-USP - São Carlos; Profª do Depto de Computação - UFSCar - Cx.P 676

2 Professores do. ICMSC - USP - São Carlos

3 Mestrandos do ICMSC - USP - São Carlos - Cx.P 668

E. Mail - JCMALDON@ICMSC.USP.BR

RESUMO

Este artigo explora a aplicação do critério Análise de Mutantes no contexto de especificações baseadas em Máquinas de Estado Finito. São apresentados os principais conceitos de Máquinas de Estado Finito e do critério Análise de Mutantes. A aplicação do critério foi feita manualmente, tomando-se como exemplo a especificação de um Protocolo de Transporte Classe 0 da ISO e por dois critérios de geração de seqüências de teste - o Método W e o Método VT [Gabo90]. Os resultados obtidos no experimento também são apresentados e mostram evidências de que o uso da Análise de Mutantes,nesse contexto, contribui para as atividades de validação. Finalmente, são apresentadas algumas considerações e propostas para o desdobramento deste trabalho.

ABSTRACT

The Mutation Analysis criterion application in the context of specification based on Finite State Machine is explored. The main concepts of Finite State Machine and of

Mutation Analysis are briefly introduced. An experiment was conducted manually using a class 0 ISO Transport Protocol Specification and two test sequence generator criteria - the Method W and the Method VT [Gabo90]. The results obtained are also presented and evidences are given that the use of Mutation Analysis is worthwhile in this context. Finally, further work in this direction are proposed and discussed.

1. INTRODUÇÃO

Sistemas de software têm sido utilizados em áreas de aplicação que possuem requisitos severos com relação à confiabilidade, segurança e tolerância a falhas, entre outros aspectos. A atividade de teste e validação desses sistemas assume um papel ainda mais crítico, principalmente quando as conseqüências de falhas podem envolver riscos à vida humana ou grandes perdas econômicas. Por outro lado, as atividades de teste e validação tornam-se mais dificeis em função das características dessas aplicações, tais como a concorrência, comunicação e sincronização, requeridas entre os processos (tarefas) para se obter uma solução adequada. Exemplos desses sistemas são: controle de tráfego aéreo e urbano, aeronaves, sistemas de protocolo de comunicação, e outras aplicações de tempo real.

Um importante aspecto do desenvolvimento desses sistemas é a especificação e validação de seu aspecto comportamental; a existência de técnicas e ferramentas para esse fim é de extrema importância. As técnicas Máquinas de Estado Finito [Gill62], Redes de Petri [Pete81] e Statecharts [Hare88] têm sido utilizadas para especificar o aspecto comportamental. Vários mecanismos de análise, teste e validação têm sido propostos para auxiliar no desenvolvimento desses sistemas [Leve87, Pete81, Boav92, Chow78].

Teste pode ser visto como a atividade final de verificação e validação dentro da organização de desenvolvimento de software [Jone90]. Teste é uma atividade relevante e extremamente onerosa de garantia de qualidade de software [Howd87, Linn90]; as atividades de teste consomem, em projetos típicos de programação, da ordem de 50% do tempo e do custo de desenvolvimento. A necessidade de teste é oriunda da inabilidade de se garantir que as demais atividades do projeto de software sejam realizadas adequadamente [Somm89].

O principal objetivo da atividade de teste de software é revelar a presença de erros no sistema; em outras palavras, refutar a afirmação de que o programa está

correto. Vários outros autores adotam essa mesma abordagem [Myer79, Pres92, Hall91, Cowa88]. Frankl acrescenta: " ...ou, não atingindo esse objetivo, aumentar a confiança de que o programa está correto" [Fran87]. Nesse sentido, um teste bem sucedido é aquele que revela a presença de um erro; um bom caso de teste é aquele que tem uma alta probabilidade de encontrar um erro ainda não descoberto. Deve-se ressaltar que não existe um procedimento de teste de propósito geral que possa ser usado para provar a corretude de um programa.

Duas são as questões chaves de pesquisa na área de teste: "Como os casos de teste devem ser selecionados" e "Como pode-se dizer se um programa P foi testado suficientemente".

Nesse sentido, a qualidade do conjunto de casos de teste T torna-se um problema chave. Em geral, existem três técnicas básicas para o projeto de casos de teste, que dão origem ao estabelecimento de critérios, métodos e estratégias de teste: a técnica funcional, a estrutural e a baseada em erros. Essas técnicas têm sido consideradas complementares, uma vez que abordam características distintas de um software em teste [Pres92]. Uma forma de se avaliar a qualidade de T é utilizar-se de medidas de cobertura que podem ser derivadas de critérios de teste [Linn90].

A Análise de Mutantes, um dos critérios da técnica baseada em erros, tem sido utilizada para avaliar a qualidade de um conjunto de casos de teste T; se bem que pode também ser utilizada para gerar conjuntos de casos de teste [Demi91]. Essa técnica consiste, basicamente, em gerar programas mutantes M com base em um conjunto de operadores de mutação que refletem os erros típicos e comuns cometidos por um programador competente (hipótese do programador competente).

Vários critérios de teste baseados em Máquinas de Estado Finito têm sido propostos para auxiliar na validação dos projetos do aspecto comportamental de sistemas que têm por base Máquinas de Estado Finito [Chow78, Gone70, Nait81, Sabn88]. Não se tem conhecimento do uso de Análise de Mutantes nesse contexto. O objetivo principal deste artigo é discutir a adequação do uso do critério Análise de Mutantes nessas atividades e explorar os aspectos complementares entre este critério e os critérios que têm sido utilizados para a validação desses projetos.

Na Seção 2 é introduzida a noção básica de Máquinas de Estado Finito e são apresentadas as características essenciais dos principais métodos baseados em Máquinas de Estado Finito. Uma breve introdução à Análise de Mutantes é apresentada na Seção 3. A utilização dos conceitos e princípios de Análise de Mutantes para validação de Máquinas de Estado Finito é apresentada na Seção 4, ilustrada através da aplicação dessa técnica, realizada manualmente, tomando-se como exemplo um protocolo extraído

de [Gabo90]. Nessa seção são também apresentadas uma análise e uma síntese dos resultados obtidos. As conclusões e desdobramento deste trabalho são discutidos na Seção 5.

2. MÁQUINAS DE ESTADO FINITO E MÉTODOS DE GERAÇÃO DE SEQÜÊNCIAS DE TESTE

Segundo Fujiwara [Fuji91], uma Máquina de Estado Finito M determinística pode ser representada por uma quíntupla (X,E,Y,T,O), onde:
X: conjunto de entradas, x.
E: conjunto de estados S_i, incluindo um estado S_0 chamado estado inicial.
Y: conjunto de saídas, y, incluindo a saída nula (-).
T: função de transição, X × E -> E.
O: função de saída, X × E -> Y.

A notação S_i -x/y-> S_j indica que a Máquina de Estado Finito M, no estado S_i, responde com uma saída y e transiciona para o estado S_j quando a entrada x é aplicada.

A máquina M é dita completamente especificada se para cada estado de M existe uma transição para cada símbolo de entrada em X. A máquina M é fortemente conectada se para cada par de estados (S_i,S_j) existe uma seqüência de entrada que faz M transicionar de S_i para S_j. A máquina M é dita minimal se o número de estados em M é menor ou igual ao número de estados para qualquer máquina M', que seja equivalente a M,isto é, que responda com uma saída idêntica para cada seqüência de entrada.

Segundo Chow [Chow78], um dos maiores problemas de teste é determinar um critério de seleção de casos de teste que seja válido e confiável [Good75]. Considerando-se M uma máquina correta e M' uma máquina que se deseja avaliar, ambas minimais, com o mesmo alfabeto de entrada, Chow classifica os erros de sequenciamento em 3 tipos: erros de operação (quando M' não é equivalente a M mas pode se tornar equivalente a M, mudando-se apenas a função de saída de M'), erros de transferência (quando M' não é equivalente a M mas pode se tornar equivalente a M, mudando-se apenas a função de próximo estado de M') e erros de estados extras ou ausentes (quando, a fim de tornar M' equivalente a M, o número de estados em M' deve ser reduzido ou aumentado; como M' e M são minimais, um número diferente de estados implica que M' e M não são equivalentes).

Vários critérios têm sido propostos na literatura para geração de seqüências de teste para Máquinas de Estado Finito, diferenciando-se essencialmente em relação às propriedades e características requeridas das máquinas em teste [Fuji91, Nait81, Gone70, Chow78, Sabn88]. Uma comparação entre as principais características desses métodos é apresentada em [Fuji91].

O método VT (Varredura de Transições) objetiva, essencialmente, a construção de uma seqüência de teste que exercite toda transição S_i-x/y->$S_j \in T$; este método, segundo [Chow78], não é eficaz para revelar erros de transferência.

Chow [Chow78] apresenta um critério de teste denominado "automata theoretic" (método W), válido e confiável para estruturas de controle que possam ser modeladas com Máquinas de Estado Finito, onde a próxima operação e o próximo estado dependem somente do estado corrente e da entrada, ou seja, não existem variáveis de controle ou contadores manipulados pelas operações que possam influenciar o sequenciamento das operações. Assume-se ainda que as máquinas sejam: completamente especificadas, minimais, tenham um estado inicial fixo e que todo estado seja alcançável. Esse método consiste de três passos principais: 1)estimativa do número de estados do projeto correto, 2)geração de seqüências de teste baseada no projeto e 3)verificação das respostas às seqüências geradas em 2). Segundo Chow, as seqüências de teste geradas por esse método, se respeitadas todas as condições para sua aplicação, garantem revelar todos os erros de sequenciamento.

Na Seção 4 explora-se a adequação de seqüências de teste geradas pelo método W e pelo método VT à Análise de Mutantes, cujos princípios básicos são discutidos na Seção seguinte.

3. ANÁLISE DE MUTANTES

A Análise de Mutantes surgiu na década de 70 na Yale University e Georgia Institute of Tecknology [Demi78]; tem-se mostrado, através de trabalhos empíricos e teóricos, ser um critério atrativo para teste de programas [Demi80, Budd80, Horg92]. Recentemente, com o avanço da tecnologia de hardware e da arquitetura de computadores, observou-se uma intensa atividade em torno da construção de ferramentas de apoio a esse critério [Choi89, Krau88, Math88], minimizando o custo de sua aplicação.

Basicamente, a idéia da Análise de Mutantes é criar a confiança de que um programa P está correto produzindo-se, através de pequenas alterações sintáticas, um conjunto de programas, chamados de mutantes, semelhantes a P, e construindo-se casos de teste capazes de provocar diferenças de comportamento entre P e seus mutantes. Essas alterações são feitas com base em um conjunto de operadores denominados de operadores de mutação. A cada operador pode-se associar um tipo ou uma classe de erros que se pretende revelar no programa.

A Análise de Mutantes consiste de 4 etapas principais: geração de mutantes, execução de P com base em um dado conjunto de casos de teste T, execução dos mutantes com base em T e análise dos mutantes.

Um ponto importante da aplicação da Análise de Mutantes é a geração de mutantes, ou seja, a escolha e definição dos operadores de mutação. Para esse objetivo, assume-se a chamada hipótese do programador competente, que afirma que um programa produzido por um programador competente ou está correto ou está próximo do correto; a noção de se restringir o conjunto de programas mutantes pode ser vista como uma delimitação dos tipos de erros que se deseja considerar. O testador deve construir casos de teste que mostrem que tais transformações conduzem a um programa incorreto.

Outra hipótese considerada na Análise de Mutantes, no nível de programa, é o efeito de acoplamento que, segundo DeMillo [Demi78], pode ser descrito da seguinte maneira: "Dados de teste que distinguem todos os programas que diferem de um programa correto somente em erros simples são tão sensíveis que também distinguem, implicitamente, erros mais complexos". Alguns estudos empíricos têm validado essa hipótese [Budd80, Acre79].

Todos os mutantes são executados usando-se os casos de teste T como entradas. Se um mutante P_i apresenta resultados diferentes de P diz-se que esse mutante está morto; nesse caso, T conseguiu identificar o "erro" no mutante, ou mais precisamente, conseguiu revelar a diferença entre P e P_i. Por outro lado, se P_i apresenta respostas idênticas a P, diz-se que ele continua vivo. Isto pode ocorrer por dois motivos: ou porque T não contém casos de teste capazes de distinguir P_i de P ou porque ambos os programas executam as mesmas funções, ou seja, são equivalentes. No primeiro caso, novos casos de teste podem ser adicionados a T para matar o mutante. No caso de mutantes equivalentes, nenhum caso de teste será capaz de distingui-los, pois seus resultados são sempre iguais aos de P. O problema de resolver se dois programas são equivalentes é, em geral, indecidível; essa limitação teórica não significa que o problema deve ser abandonado por não ter solução. Na verdade, alguns métodos e algumas

heurísticas têm sido propostos para determinar a equivalência de programas em uma grande porcentagem dos casos de interesse [Budd81].

O objetivo é achar um conjunto de casos de teste que consiga matar todos os mutantes não equivalentes. Tais conjuntos são considerados adequados para o teste de P, no sentido de que ou P está correto ou contém um erro sutil e inesperado, o que deve ser raro se as modificações usadas para criar os mutantes forem cuidadosamente escolhidas.

Mutantes gerados a partir de k alterações simultâneas no programa P sendo testado são chamados de mutantes de ordem k. Experiências passadas [Budd80] mostram que mutantes de ordem superior a um (k>1), além de não contribuirem de forma significativa para a construção de casos de teste melhores, têm um custo de geração e execução demasiado alto. Portanto, tem-se utilizado na Análise de Mutantes uma vizinhança composta apenas dos mutantes de primeira ordem.

Um ponto importante destacado em [Demi80] é que a Análise de Mutantes fornece uma medida objetiva do nível de confiança na adequação dos casos de testes analisados. Através do escore de mutação (mutation score), que relaciona o número de mutantes gerados com o número de mutantes mortos, pode-se avaliar a adequação dos casos de testes usados e, como conseqüência, a confiabilidade do programa testado.

Dado o programa P e o conjunto de casos de teste T, calcula-se o escore de mutação ms(P,T) da seguinte maneira:

$$ms(P,T) = \frac{DM(P,T)}{M(P) - EM(P)}$$

onde:

DM(P,T): número de mutantes mortos pelos casos de teste em T
M(P): número total de mutantes gerados
EM(P): número de mutantes gerados equivalentes a P

Ou seja, o escore de mutação pode ser obtido calculando-se a razão entre o número de mutantes efetivamente mortos por T e o número de mutantes que é possível matar-se, dado pelo número total de mutantes gerados subtraído do número de mutantes equivalentes.

Note-se que apenas DM(P,T) depende do conjunto de casos de teste utilizado. Apesar disso, não se conhece, a princípio, o número de mutantes equivalentes gerados. EM(P) é obtido iterativamente à medida que o testador decide ou decide-se

automaticamente, através da aplicação de heurísticas, marcar como equivalente um mutante M.

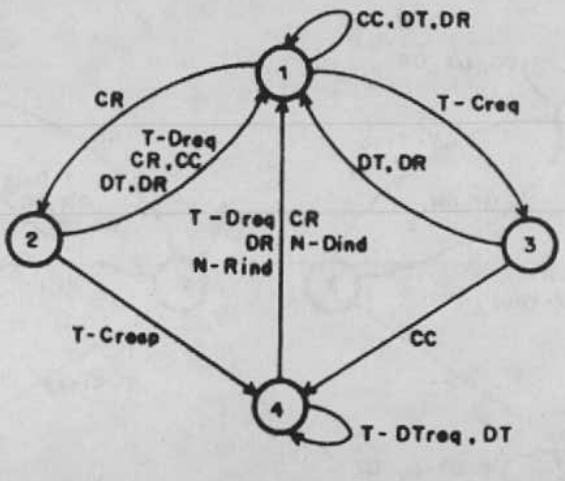
Neste ponto é importante ressaltar que a questão que se coloca no planejamento e desenvolvimento de software não é escolher qual das técnicas ou critérios aplicar, e sim como e quando utilizá-los adequadamente, uma vez que todos apresentam vantagens e possuem, por outro lado, limitações.

4. ANÁLISE DE ADEQUAÇÃO DE SEQÜÊNCIAS DE TESTE BASEADAS EM MÁQUINAS DE ESTADO FINITO

Para aplicar a Análise de Mutantes no contexto de Máquinas de Estado Finito, faz-se um paralelo com a hipótese do programador competente no nível de programas, estabelecendo a hipótese do especificador competente em Máquinas de Estado Finito, ou seja, um projeto (modelo) produzido por um especificador competente ou está correto, ou está próximo do correto. Da mesma forma, a hipótese que relaciona a eficiência do teste para erros únicos e para erros múltiplos (efeito de acoplamento) será explorada nesse contexto. Além disso, para sua aplicação, torna-se necessária a definição de um conjunto de operadores que, aplicado ao projeto original, dará origem aos mutantes. Tal conjunto deve refletir os erros mais freqüentes e comuns cometidos por um especificador ao construir uma Máquina de Estado Finito; ele é formado, preliminarmente, pelos seguintes operadores: falta de um arco, alteração do estado inicial (default), evento faltando, evento trocado, evento extra, estado faltando, estado extra, saída trocada, saída faltando, saída a mais, entre outros [Fabb93].

Com base nesse conjunto inicial de operadores, a Análise de Mutantes para Máquinas de Estado Finito foi avaliada tomando-se como exemplo o Diagrama de Estado do Protocolo de Transporte Classe 0 da ISO [Gabo90], apresentado na Figura 4.1, e as seqüências de teste geradas pelo método W [Chow78] e pelo método VT [Nait81], apresentadas também em [Gabo90].

Na Figura 4.2 apresenta-se exemplos de 1-mutante para alguns desses operadores e na Tabela 4.1 apresenta-se uma síntese da aplicação da Análise de Mutantes ao exemplo ilustrado na Figura 4.1. Ressalte-se que a definição de um conjunto de operadores de mutação é um ponto fundamental na aplicação da Análise de Mutantes e que o conjunto utilizado neste trabalho deve ser considerado como uma tentativa preliminar nessa direção.

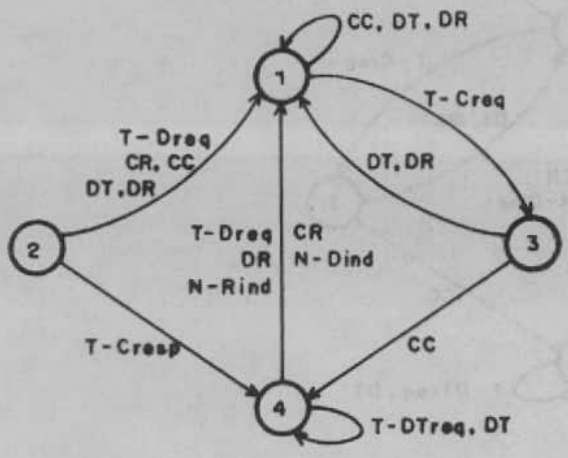


ESTADO / ENTRADA	1 LIVRE	2 ESPERA CONF. DO USUÁRIO	3 ESPERA CONF. DA REDE	4 TRANS. DE DADOS
T_Creq	3/CR	-	-	-
T_Dreq	-	1/DR	-	1/N_Dreq
T_Cresp	-	4/CC	-	-
T_DTreq	-	-	-	4/DT
CR	2/T_Cind	1/ERR	-	1/ERR
CC	1/-	1/ERR	4/T_Cconf	-
DT	1/-	1/ERR	1/-	4/T_DTind
DR	1/-	1/ERR	1/T_Dind, N_Dreq	1/N_Dreq
N_Dind	-	-	-	1/T_Dind
N_Rind	-	-	-	1/T_Dind

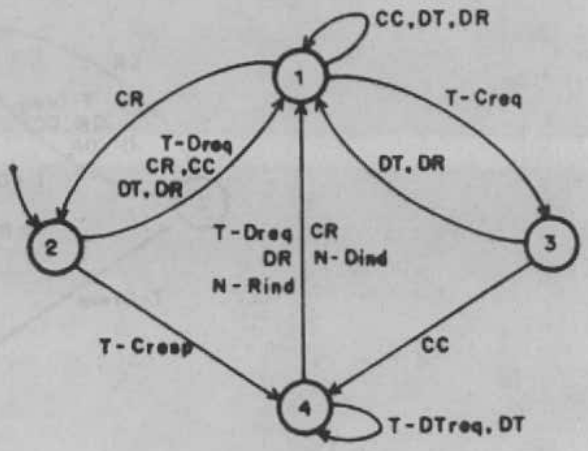
Seqüência VT = CR, T_Dreq, CC, DT, DR, CR, CR, T_Creq, DT, CR, T_Cresp, T_DTreq, DT, T_Dreq, CR, CC, CR, DT, CR, DR, T_Creq, DR, T_Creq, CC, CR, CR, T_Cresp, DR, CR, T_Cresp, N_Dind, CR, T_Cresp, N_Rind

Seqüência W = (DR, T_Creq.DR, CR.DR, CC.DR, DT.DR, DR.DR,
CR.(T_Dreq, T_Cresp, CR, CC, DT, DR).DR,
T_Creq.(CC, DT, DR).DR,
CR.T_Cresp.(T_Dreq, T_DTreq, CR, DT, DR, N_Dind, N_Rind).DR)

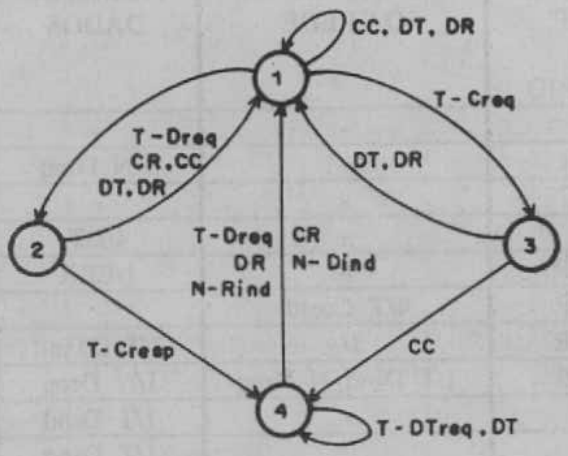
Figura 4.1- Máquina de Estado, Tabela de Estados, Seqüência VT e Seqüência W do Exemplo considerado, respectivamente.



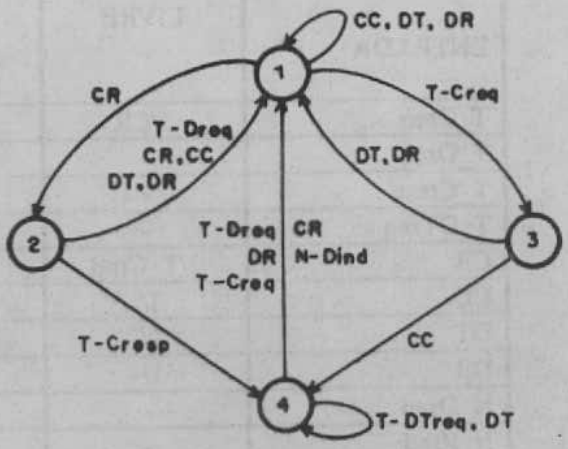
OP I - FALTA DE UM ARCO



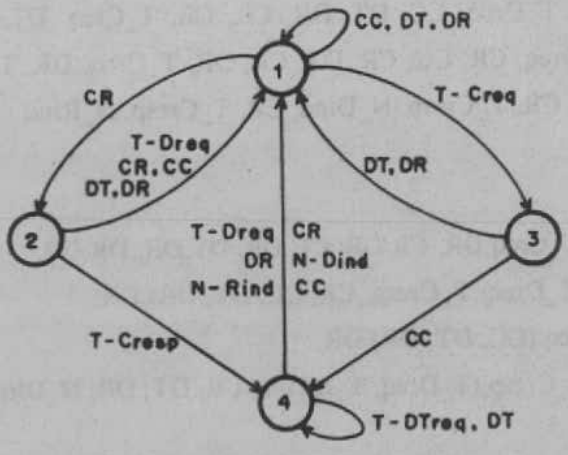
OP II - ALTERAÇÃO DO ESTADO INICIAL



OP III - EVENTO FALTANDO



OP IV - EVENTO TROCADO



OP IX - EVENTO A MAIS

Figura 4.2- Exemplo de Mutantes Gerados por Alguns Operadores.

Tabela 4.1 - Síntese da Aplicação da Análise de Mutantes ao Exemplo considerado.

CRITÉRIOS	MÉTODO VT				MÉTODO W			
	Seq. de Estado		Saída		Seq. de Estado		Saída	
OPERADORES DE MUTAÇÃO	Mortos	Vivos	Mortos	Vivos	Mortos	Vivos	Mortos	Vivos
OP I - Falta de Arco Total = 9 Equivalentes = 1	7	2	8	1	7	2	8	1
OP II - Alteração do Estado Inicial Total = 3 Equivalentes = 0	3	0	3	0	0	3	3	0
OP III - Evento Faltando Total = 21 Equivalentes = 3	16	5	18	3	16	5	18	3
OP IV - Evento Trocado Total = 139 Equivalentes = 21	70	21	76	15	70	21	76	15
OP V - Estado Faltando Total = 5 Equivalentes = 0	1	0	1	0	1	0	1	0
OP VI - Saida Trocada Total = 193 Equivalentes = 0	0	193	193	0	0	193	193	0
OP VII - Saida Faltando Total = 17 Equivalentes = 0	0	17	17	0	0	17	17	0
OP VIII - Estado a mais Analisados = 6 Equivalentes = 6	6	0	0	6	6	0	0	6
OP IX - Evento a mais Analisados = 43 Equivalentes = 0	0	43	0	43	0	43	0	43

Primeiramente, nota-se que, na prática, as especificações não satisfazem as restrições iniciais para a aplicação dos métodos de geração de seqüências de teste que, em geral, são: máquina completamente especificada, minimal, fortemente conectada e determinística. Por exemplo, a Máquina de Estado Finito considerada não satisfaz a condição de ser completamente especificada.

Caso todas as restrições iniciais fossem satisfeitas, as seqüências geradas pelo método W, em teoria, deveriam ter um escore de mutação de 100% se considerássemos somente 1-mutante pois, se as Máquinas de Estado Finito respeitassem as hipóteses para a aplicação do método W, todos os erros de sequenciamento no projeto seriam revelados [Chow78] e, consequentemente, todos os mutantes deveriam ser distinguidos da Máquina M original, por refletirem as classes de erros consideradas: erro de operação, erro de transferência e erro de estados extras ou ausentes.

Em relação ao exemplo, mesmo não sendo respeitadas as condições iniciais, a seqüência gerada pelo método W obteve um escore de mutação bastante alto, considerando-se os valores apresentados na Tabela 4.1 e a definição de escore de mutação dada na Seção 3. Observe que para distinção entre o comportamento da Máquina original M e dos mutantes, foram utilizadas duas alternativas: saída produzida e seqüência de estado. Do exemplo, pode-se observar que a saída é mais efetiva para esse propósito.

Observe que as seqüências geradas pelo método W e pelo método VT tiveram praticamente a mesma adequação em relação à Análise de Mutantes; isto é decorrente do fato da máquina ser simples e simétrica. Sem dúvida, este não seria o comportamento em geral pois, as seqüências geradas pelo método VT não são tão eficazes na revelação de certas classes de erros quanto o método W [Chow78]. A seguir, discute-se mais detalhadamente a adequação da seqüência gerada pelo método W, em relação à Análise de Mutantes.

Os mutantes que permanecem vivos (considerando-se a saída), num total de 43, são aqueles decorrentes da aplicação do operador evento-extra; este fato é possível em decorrência da Máquina de Estado Finito não ser completamente especificada. Observou-se a existência de muitos mutantes equivalentes; a existência desses mutantes equivalentes também está associada ao fato da Máquina de Estado Finito não preencher todos os requisitos para a aplicação do método; por exemplo, o operador evento-trocado explora a não completude da máquina e gera 21 mutantes equivalentes.

Essas observações contribuem para o estabelecimento de heurísticas para a identificação de mutantes equivalentes. Note-se que esse é um aspecto relevante para a aplicação da Análise de Mutantes pois, em geral, é um problema indecidível. A

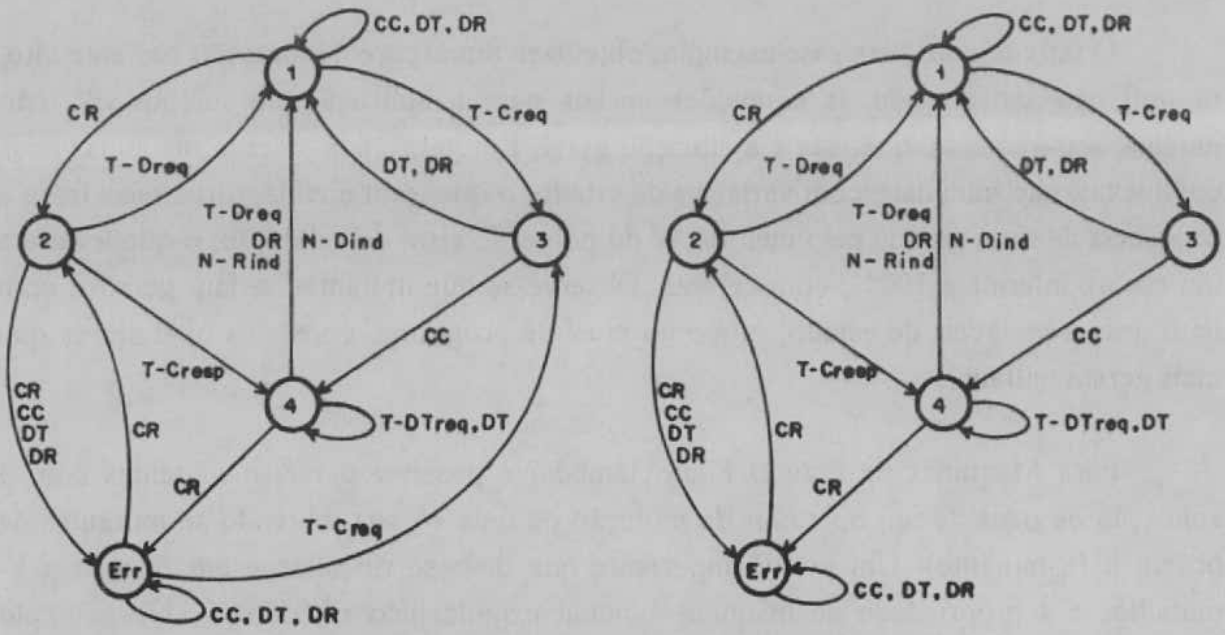
identificação de mutantes equivalentes ou é decidida através do uso de heurísticas ou através da intervenção do testador.

O fato de que, para esse exemplo, obteve-se um escore de mutação bastante alto, mesmo não satisfazendo as condições iniciais para a aplicação do método W, não implica, seguramente, que esta é a situação geral. Em geral, lida-se com máquinas mais complexas, não minimais, com variáveis de estado, o que, sem dúvida, torna mais fraca a seqüência de teste gerada pelo método W do ponto de vista de validação, o que levaria a um escore inferior a 100%, com certeza. Observe-se que mutantes seriam gerados com base nessas variáveis de estado, o que no nível de programa, é um dos operadores que mais geram mutantes.

Para Máquinas de Estado Finito também é possível gerar-se mutantes com a aplicação de mais de um operador de mutação de uma só vez, obtendo-se mutantes de ordem k (k-mutante). Um ponto importante que deve-se ressaltar é em relação a k-mutantes e a propriedade de máquina minimal exigida pelo método W. No exemplo apresentado existem evidências de que, em geral, o efeito de acoplamento seria válido também para a Análise de Mutantes baseada em Máquinas de Estado Finito, se bem que essa hipótese deve ser avaliada para um número significativo de modelos. No entanto, a combinação do operador estado-extra, levando a máquinas mutantes equivalentes não minimais, com os demais operadores de mutação, gera mutantes para os quais as seqüências de teste são ineficazes, ou seja, não são capazes de distinguir o comportamento de M e desses mutantes.

Considere uma máquina Me equivalente à máquina M, ilustrada na Fig 4.3a, concebida pressupostamente de forma a facilitar o entendimento e a manutenção da máquina M, no que concerne à emissão de mensagens de erros. A máquina Me é uma máquina não minimal pois os estados 1 e Err são equivalentes. Se considerarmos agora mutação na máquina Me, ilustrada na Fig. 4.3b, as seqüências geradas pelo método W, com base na máquina M, não distinguiria uma boa parte desses mutantes: considerando apenas o operador falta-de-arco, de 5 mutantes apenas 2 são eliminados, dando aproximadamente 40% de escore de mutação (considerando apenas a saída).

Considere a notação (S_i,S_j) para indicar a existência de uma transição $t \in T$, para algum $x \in X$ e o conjunto $X(S_i,S_j)=\{x/x \in S_i\text{-}x/y\text{->}S_j,\ y \in O\}$ para indicar as entradas que provocam a transição de S_i para S_j. Uma política para a geração de mutantes equivalentes não minimais com um estado extra, pela aplicação do operador estado-extra, seria gerar um mutante para todo par de estados (S_i,S_j) tal que $card(X(S_i,S_j)) > 1$, da seguinte forma: para cada $x \in X(S_i,S_j)$, gera-se um estado equivalente S_j' tal que $x \in X(S_i,S_j')$ e $x \notin X(S_i,S_j)$ no mutante gerado, conforme ilustrado na Fig. 4.4.



Figuras 4.3a e 4.3b- Aplicação do Operador Estado Extra para Isolar Situação de Erro e do Operador Falta de Arco sobre o Operador Estado Extra, respectivamente.

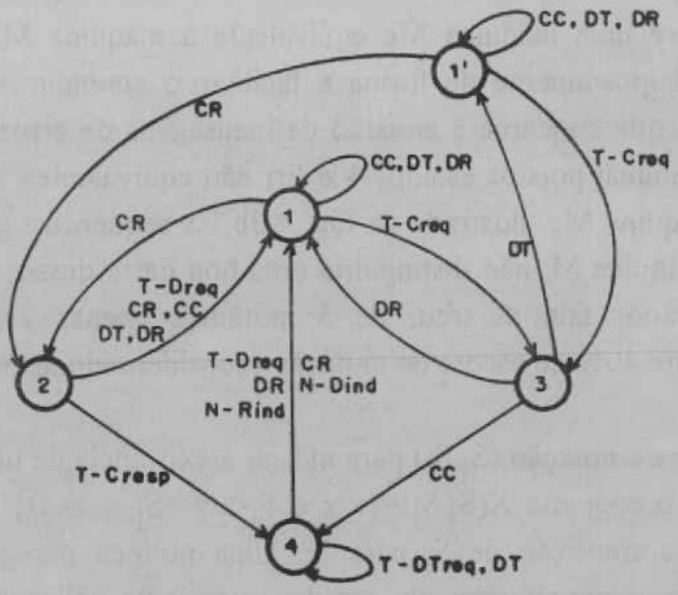


Figura 4.4- Aplicação do Operador Estado Extra para Desmembrar Transições com mais de um Evento.

Considere a notação $O(S_i,S_j)=\{y/y \in S_i\text{-}x/y\text{->}S_j, x \in X \text{ e } y \in O\}$ para indicar as possíveis saídas produzidas pela ocorrência de transições que levem de S_i para S_j. Outra forma seria gerar para cada par de estados (S_i,S_j) tal que $card(O(S_i,S_j)) > 1$, $card(O(S_i,S_j))$ mutantes equivalentes não minimais com um estado extra. Para cada $y \in O(S_i,S_j)$ gera-se um estado equivalente S_j' tal que $y \in O(S_i,S_j')$ e $y \notin O(S_i,S_j)$ no mutante gerado, e para todo $x \in X/ S_i\text{-}x/y\text{->}S_j$ na máquina M, $x \notin X(S_i,S_j)$ e $x \in X(S_i,S_j')$ na máquina mutante, conforme ilustrado na Fig.4.5.

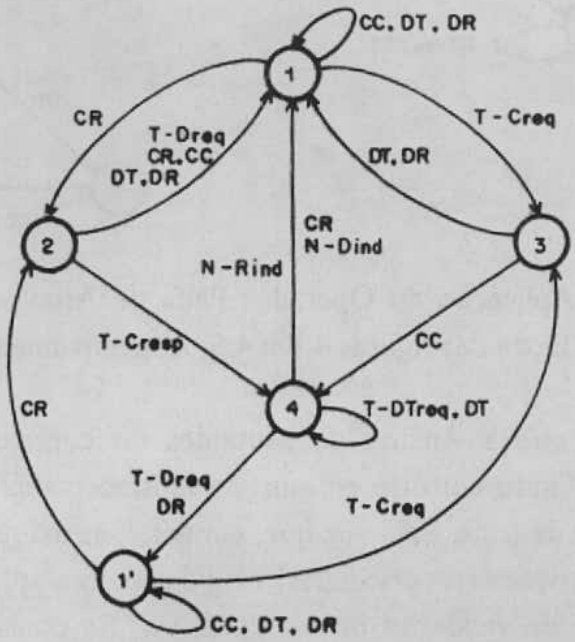
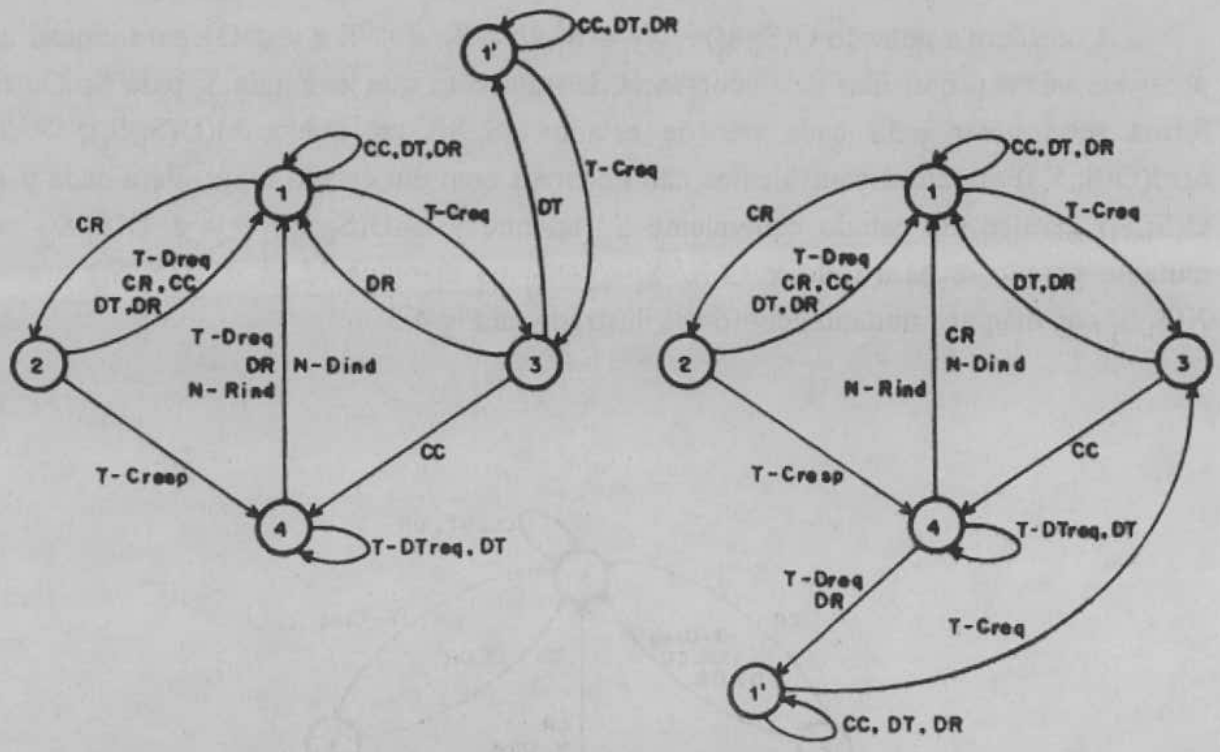


Figura 4.5- Aplicação do Operador Estado Extra para Separar Eventos de uma mesma Transição que produzem a mesma Saída.

Após a geração desses mutantes equivalentes não minimais, seriam aplicados os demais operadores de mutação, gerando-se dessa forma mutantes de ordem 2 (2-mutantes). Considere um exemplo de 2-mutante gerado pela aplicação do operador falta-de-arco sobre esses mutantes equivalentes, em particular sobre o mutante da Figura 4.5, no conjunto de transições que incluem o estado extra, ilustrado na Figura 4.6.

O escore de mutação seria de aproximadamente 25%, um escore relativamente baixo. Por exemplo, o comportamento do 2-mutante da Figura 4.6b, aplicado sobre o mutante da Figura 4.4, e o da máquina original não é distinguido pela seqüência de teste gerada pelo método W.



Figuras 4.6a e 4.6b- Aplicação do Operador Falta de Arco sobre o Operador Estado Extra das figuras 4.4 e 4.5, respectivamente.

Fica evidente que a Análise de Mutantes no contexto de teste baseado em Máquinas de Estado Finito consiste em um mecanismo complementar aos métodos de geração de seqüências de teste, uma vez que, em geral, as máquinas de estado (modelos) não preenchem as restrições (propriedades) exigidas para a aplicação desses métodos, o que torna a atividade de validação menos confiável. Se considerarmos a atividade de validação no contexto das outras técnicas, tais como Statecharts, a utilização de Análise de Mutantes justifica-se mais facilmente, uma vez que não se tem sequer métodos adequados para a geração de seqüências de teste para essas técnicas.

5. CONCLUSÕES

Explorou-se, neste artigo, a aplicação da técnica Análise de Mutantes para avaliar a adequação de seqüências de teste geradas para Máquinas de Estado Finito, em especial, para as seqüências de teste produzidas pelo Método W [Chow78] e pelo Método VT [Nait81]. A aplicação da técnica foi feita manualmente, tomando-se como exemplo, a especificação de um Protocolo de Transporte Classe 0 da ISO [Gabo90].

O exemplo, apesar de simples, mostra evidências de que a Análise de Mutantes é uma técnica que pode ser vista como complementar, uma vez que as Máquinas de Estado Finito, em geral, não satisfazem as condições iniciais para a aplicação dos métodos.

Um ponto importante a ser explorado ainda, é a determinação de uma taxonomia de erros, assim como propos Beizer [Beiz90] para o nível de programas, tomando como partida a proposta feita por Chow [Chow78], objetivando estabelecer um conjunto mais eficaz de operadores de mutação para Máquinas de Estado Finito.

Outro ponto que também será explorado refere-se à minimização do número de mutantes gerados, como é comentado em [Acre80] para programas. Uma das maneiras é verificar a eficácia dos operadores de mutação. Ressalte-se que a geração de 2-mutantes, como foi mencionado na seção anterior, parece ser bastante eficaz no contexto de Máquinas de Estado Finito.

O desdobramento deste trabalho será , em uma primeira instância, automatizar os métodos de geração de seqüências de teste e a Análise de Mutantes para Máquinas de Estado Finito. Isso será feito dentro do ambiente STATSIM, em desenvolvimento pelo Grupo de Engenharia de Software do ICMSC-USP, que dá apoio ao desenvolvimento de sistemas baseado na técnica Statecharts. Nesse ambiente, já encontram-se implementados um editor gráfico, uma linguagem de especificação e um simulador de Statecharts, dentre outras ferramentas [Masi91, Boav92] Nessa primeira etapa, a técnica Análise de Mutantes será restringida a Máquina de Estado Finito e, numa segunda etapa, os métodos de geração de seqüências de teste e a Análise de Mutantes serão estendidos, procurando-se explorar todos os recursos oferecidos pela técnica Statecharts tais como: hierarquia, história, paralelismo e mecanismos de sincronização.

BIBLIOGRAFIA

[Acre79] Acree, A. et al *Mutation Analysis*, Technical Report GIT-ICS-79/08, Georgia Institute of Technology, set., 1979.

[Beiz90] Beizer, B. *Software Testing Technics*, 2nd Edition, Van Nostrand Reinhold, N.Y., 1990.

[Boav92] Boaventura, I.A.G. *Propriedades Dinâmicas de Statecharts*, Dissertação de Mestrado, ICMSC-USP, São Carlos, 1992.

[Budd80] Budd, T.A.; DeMillo, R.A.; Lipton, R.J.; Sayward, F.G. *Theoretical and Empirical Studies on Using Prog Mutation to Test the Functional Correctiness of Prog.*, 7th ACM Symposium on Principles of Programming Languages, jan., 1980.

[Budd81] Budd, T.A. Mutation Analysis: *Ideas, Examples, Problems and Prospecties*, Computer Program Testing, North-Holand Publishing Company, 1981.

[Choi89] Choi, B.J. et al *The Mothra Tool Set*, Proceedings of the 22nd Hawai International Conference on Systems and Software, Kona, Hawai, jan., 1989.

[Chow78] Chow, T.S. *Testing Software Design Modeled by Finite-State Machines.* IEEE Transactions on Software Engineering, SE(4(3)), pp. 178-187, 1978.

[Cowa88] Coward, P.D. *A Review of Software Testing*, Information and Software Technology, Vol. 30, N. 3, 1988, pp. 189-198.

[Demi78] DeMillo, R.A.; Lipton, R.J.; Sayward, F.G. *Hints on Test Data Selection: Help for the Practicing Programmer*, Computer, Vol. 11(4), pp.34-41, 1978.

[Demi80] DeMillo, R.A. *Mutation Analisys as a Tool for Software Quality Assurance*, Proc. of COMPSAC 80, Chicago-IL, outubro, 1980.

[Demi91] DeMillo, R.A.; Offut, A.J. *Constraint-Based Automatic Test Data Generation*, IEEE, Transactions on Software Engineering SE(17(9)), 1991.

[Fabb93] Fabbri, S.C.P.F.; Maldonado, J.C.; Masiero, P.C.; Delamaro, M.E.; Nakazato, K.K. *Considerações sobre o Projeto de Operadores de Mutação Baseado em Máquinas de Estado Finito*, trabalho submetido à Primeira Jornada USP-SUCESU-SP de Informática e Telecomunicações, 1993.

[Fran87] Frankl, F.G. *The Use of Data Flow Information for the Selection and Evaluation of Software Test Data*, PhD Dissertation, N.Y. Univ., N.Y., outubro de 1987.

[Fuji91] Fijiwara, S. et al *Test Selection Based on Finite State Models*, IEEE, Transactions on Software Engineering, Vol. 17, N. 6, June, 1991.

[Gabo90] Gabos, D.; Stiubiener, S. *Aspectos de Metodologia de Geração de Seqüências de Teste para Protocolos de Comunicação de Dados*, in Anais 8º Simpósio de Redes de Computadores, 1990.

[Gill62] Gill, A. *Introduction to the Theory of Finite-State Machines.* New York, McGraw-Hill, 1962.

[Gone70] Gonenc, G. *A Method for the Design of Fault-Detection Experiments*, IEEE Trans. Comput., vol C-19, pp 551-558, june 1970.

[Good75] Goodenough, J.B.; Gerhart, S.L. *Toward a Theory of Test Data Selection*, in Int. Conf. Reliable Software, Los Angeles, 1975.

[Hall91] Hall, P.A.V. *Relationship between Specifications and Testing*, Information and Software Technology, Vol. 33, N. 1, 1991, pp. 47-52.

[Hare88] Harel, D. *On Visual Formalisms.* CACM, 31(5), pp. 514-530, 1988.

[Horg92] Horgan, J. R.; Mathur, A.P. *Assessing Testing Tools in Research and Education*, IEEE Software, Vol. 9, N. 3, Maio, 1992.

[Howd87] Howden, W.E. *Functional Program Testing and Analysis*, McGraw-Hill, USA, 1987.

[Jone90] Jones, G.W. *Software Engineering*, John Wiley & Sons, USA, 1990.

[Krau88] Krauser, E.W.; Mathur, A.P.; Rego, V. *High Performance Testing on SIMD*, Proc. of the 2nd Workshop on Software Testing, Verification and Analisys, Banff, Canada, 1988.

[Leve87] Levesson, N.G. ; Stolzy, J.L. *Safety Analisys using Petri Nets*. IEEE Transactions on Software Engineering, SE(13(3)), pp. 386-397, 1987.

[Linn90] Linnenkugel, V.; Müllerburg, M. *Test Data Selection Criteria for (Software) Integration Testing*, in Proceedings of the First International Conference on Systems Integration, Morristown, New Jersey, April 1990.

[Masi91] Masiero, P.C.; Fortes, R.P.M.; Batista Neto, J.E.S. *Edição e Simulação do Aspecto Comportamental de Sistemas de Tempo Real*, Anais do XVIII Seminário Integrado de Hardware e Software, SBC, Santos-SP, pp. 45-61, 1991.

[Math88] Mathur, A.P.; Krauser, E.W. *Modeling Mutation on Vector Processor*, Proc. of the 2nd Workshop on Software Testing Testing, Verification and Analisys, Banff, Canada, 1988.

[Myer79] Myers,G.J. *The Art of Software Testing*, Willey, N.Y., 1979.

[Nait81] Naito, S.; Tsunoyama, M. *Fault Detection for Sequential Machines by Transition-Tours*, in Proceedings FTCS (Fault Tolerant Comput. Systems), pp 238-243, 1981.

[Pete81] Peterson, J.L. *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

[Pres92] Pressman, R.S. *Software Engineering - A Practitioner's Approach*, (3rd edition), McGraw-Hill, 1992.

[Sabn88] Sabnani, K.K.; Dahbura, A.T. *A Protocol Testing Procedure*, Comput. Networks and ISDN Syst., Vol. 15, N. 4, pp. 285-297, 1988.

[Somm89] Sommerville, I. *Software Engineering*, terceira edição, Addison-Wesley Publ. Company, 1989.