APLICAÇÃO DA ANÁLISE DE MUTANTES NA VALIDAÇÃO DE ESPECIFICAÇÕES BASEADAS EM REDES DE PETRI

Sandra C.P.F. Fabbri[1]
UFSCar[2]/IFSC-USP - e.mail sfabbri@icmsc.sc.usp.br
José Carlos Maldonado
ICMSC-USP[3] - e.mail jcmaldon@icmsc.sc.usp.br
Paulo Cesar Masiero
ICMSC-USP[3] - e.mail masiero@icmsc.sc.usp.br
Márcio Eduardo Delamaro[1]
IFSC-USP - e.mail med@icmsc.sc.usp.br

RESUMO

A atividade de Teste é uma das atividades fundamentais do ciclo de desenvolvimento de software. Para Sistemas Reativos, a atividade de validação de seu aspecto comportamental é ainda mais relevante, uma vez que falhas nesses sistemas, em geral, provocam grandes perdas econômicas e sociais. O objetivo deste artigo é explorar a aplicação do critério de teste Análise de Mutantes para validar especificações baseadas em Redes de Petri. Apresentam-se o projeto dos operadores de mutação para Redes de Petri, ponto chave para a aplicação do critério Análise de Mutantes, e os resultados obtidos da aplicação manual desse critério em uma especificação de um protocolo (extraída de [TAN89]). A viabilidade de automatização da aplicação do critério Análise de Mutantes para Redes de Petri é brevemente discutida , com base na ferramenta Proteum/FSM [FAB94a], especificada para apoiar o uso desse critério na validação de especificações baseadas em MEF.

ABSTRACT

Testing is one of the fundamental software development life cycle acitivities. Considering Reactive Systems, this acitivity becomes more relevant as errors in these systems can promote severe economical and social losses. The objective of this work is to evaluate the adequacy of applying the Mutation Analysis criterion to validate Petri Net based specifications. A set of mutation operators for Petri Nets, a key point for using Mutation Analysis, as well as the results of applying manually these operators to a Petri Net modeling a level 3 protocol (extracted from [TAN89]) are presented. Taking as reference the software tool PROTEUM/FSM [FAB94a], that has been developed to support Mutation Analysis use for validating Finite State Machine based specification, it is briefly discussed that implementing a tool to support the validation of a Petri Net based specification constitutes a feasible task.

Palavras-chaves: Redes de Petri, Análise de Mutantes, Teste eValidação.

---

1 Doutorandos do Instituto de Física de São Carlos - USP

2 Universidade Federal de São Carlos - Departamento de Computação - Cx.Postal 676, 13565-905 - São Carlos, SP

3 Instituto de Ciências Matemáticas de São Carlos - USP - Cx. Postal 668, 13560-970 - São Carlos, SP

1. INTRODUÇÃO

Os Sistemas Reativos são uma classe de sistemas cuja característica principal é interagir com o meio ambiente reagindo a estímulos. Como exemplo desses sistemas pode-se citar: controle metroviário, monitoramento hospitalar de pacientes, protocolos de comunicação, etc.. Em geral, falhas nesses sistemas podem provocar tanto perdas econômicas como também de vidas humanas, o que torna a atividade de teste dos mesmos ainda mais relevante.

O objetivo principal da atividade de teste é revelar a presença de erros e, quanto mais cedo no ciclo de vida os erros forem detectados, menos oneroso é o processo para removê-los. O fato de não serem encontrados erros, no entanto, não significa, necessariamente, que o software não os possua; pode significar apenas que o conjunto de casos de teste utilizado não é adequado o suficiente para revelá-los.

Existem, basicamente, três técnicas de teste: a funcional, onde os requisistos de teste são extraídos da especificação do software; a estrutural, onde os requisitos são extraídos de uma implementação em particular, e a baseada em erros, onde os requisitos de teste são extraídos dos erros típicos e comuns cometidos durante a fase de implementação, e da qual um dos critérios é o critério Análise de Mutantes, alvo de estudo neste trabalho. No nível de programas, vários estudos empíricos mostram o aspecto complementar dessas técnicas [PRE92].

Considerando-se os Sistemas Reativos, as técnicas gráficas mais utilizadas para sua especificação são: Máquinas de Estado Finito (MEF) [GIL62], Statecharts [HAR87] e Redes de Petri [PET81] e, para a validação do aspecto comportamental de sistemas descritos através dessas técnicas, utilizam-se, em geral, a análise de alcançabilidade, vários tipos de simulação e, para as MEFs, alguns métodos de geração de seqüências de teste [NAI81, CHO78, FUJ91, SAB88].

Uma linha de pesquisa que vem sendo conduzida no Instituto de Ciências Matemáticas de São Carlos ICMSC - USP explora qual a adequação, o custo e como as técnicas de teste se aplicam e se complementam para a validação de especificações de sistemas. Neste trabalho, o objetivo é explorar esse aspecto, dando ênfase à aplicação do critério Análise de Mutantes na validação do aspecto comportamental de sistemas reativos, especificados através de Redes de Petri.

No ICMSC-USP vem sendo desenvolvido o ambiente StatSim [MAS91] que apoia a especificação e simulação de sistemas através da técnica Statecharts e uma das diretrizes de pesquisa tem por objetivo introduzir nesse ambiente métodos e critérios de teste e de validação dessas especificações. Em um trabalho anterior [FAB93], aplicou-se, num primeiro passo, o critério Análise de Mutantes na validação de MEF, dado que a técnica Statecharts é uma extensão das MEFs. Embora existam vários métodos para geração de seqüências de teste para MEFs, eles exigem que a MEF satisfaça uma série de requisitos iniciais para sua aplicação, o que, na prática, em geral, não acontece, tornando as seqüências de teste menos eficazes. Devido a esse fato, a aplicação do critério Análise de Mutantes no contexto de MEFs forneceu evidências de que este critério é complementar em relação aos métodos de geração de seqüências de teste para MEFs. Esse estudo foi realizado manualmente, tomando-se como exemplo a especificação de um Protocolo de Transporte e duas seqüências de teste: uma

gerada pelo método TT [NAI81] e outra gerada pelo método W [CHO78]. Mas, pelo fato da especificação não satisfazer os requisitos iniciais, as seqüências não foram tão eficazes e o comportamento de uma série de mutantes não foi distingüido do comportamento da MEF original, ao aplicar-se essas seqüências de teste.

Atualmente, desenvolve-se uma ferramenta, denominada PROTEUM/FSM [FAB94a], cujo objetivo é apoiar a aplicação do critério Análise de Mutantes na validação de especificações baseadas em MEF. Ainda, está sendo conduzida a extensão desses conceitos para a técnica statecharts. Saliente-se que a extensão da PROTEUM/FSM para Statecharts é um passo relativamente simples, pois o simulador MEF é simplesmente uma versão funcional restrita do simulador de statecharts disponível no ambiente Statsim.

Motivados pelo resultado mencionado anteriormente, o mesmo estudo foi conduzido para a técnica de especificação Redes de Petri, procurando, desta forma, fornecer subsídios para eventuais estudos teóricos e empíricos comparativos referentes a atividades de validação do aspecto comportamental de sistemas, especificados através de diferentes técnicas. Neste caso, o critério Análise de Mutantes foi aplicado, também de forma manual, à especificação de um Protocolo nível 3, extraído de [TAN89], e os resultados obtidos são apresentados neste trabalho. Evidências são apresentadas que motivam o refinamento deste estudo e o desenvolvimento de uma ferramenta de suporte, a exemplo do que vem sendo conduzido para MEF.

Nas Seções 2 e 3 apresentam-se os conceitos do critério Análise de Mutantes e da técnica de especificação Redes de Petri, respectivamente; na Seção 4 apresentam-se os resultados obtidos da aplicação da Análise de Mutantes sobre Redes de Petri, dando ênfase ao projeto dos operadores de mutação e, na Seção 5 as conclusões e desdobramentos deste trabalho.

2. ANÁLISE DE MUTANTES

Basicamente, a idéia do critério Análise de Mutantes [DEM78] 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 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 fim, recorre-se da chamada hipótese do programador competente, que afirma que um programa produzido por um programador competente ou está correto ou está próximo do correto e da hipótese do efeito de acoplamento que considera que dados de teste que distinguem os programas que diferem de

um programa correto somente em erros simples são tão sensíveis que também distinguem, implicitamente, programas com erros mais complexos; 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.

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 distinguí-los, pois seus resultados são sempre iguais aos de P. O objetivo é achar um conjunto de casos de teste que consigam matar todos os mutantes não equivalentes.

Mutantes gerados a partir de k alterações simultâneas no programa P sendo testado são chamados de mutantes de ordem k. Experiências passadas [BUD80] 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 [DEM80] 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

Note-se que apenas DM(P,T) depende do conjunto de casos de teste utilizado. Apesar disto, 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.

O critério Análise de Mutantes é, originalmente, um critério utilizado para o teste de programas; no contexto desta pesquisa, esse critério foi "trazido" para outro nível de

aplicação, isto é, validação de especificação do aspecto comportamental de sistemas reativos. Para tanto, como é apresentado em [FAB93, FAB94b], traçou-se um paralelo entre as hipóteses básicas do critério, no nível de programas, para o nível de especificações. Para o projeto de operadores considera-se a hipótese do especificador competente e para aplicação desses considerar-se-ia o efeito de acoplamento, que na realidade deve ser "validado neste novo contexto". Então, dada uma especificação S, gera-se um conjunto de mutantes de S, $\phi(S)$ e diz-se que um conjunto de teste T é adequado para S em relação a ϕ se para cada especificação Z pertencente a ϕ, ou Z é equivalente a S ou Z difere de S em pelo menos um ponto de teste.

3. REDES DE PETRI

Rede de Petri (RP) [PET81] é uma técnica usada para a modelagem de sistemas, constituída de dois componentes básicos: um conjunto de *lugares*, P, e um conjunto de *transições*, T. O relacionamento entre esses dois componentes é dado através de duas funções que conectam as transições aos lugares: a função de *input* *I* e a função de *output* *O*. *I* define, para cada transição t_j, o conjunto $I(t_j)$, de *lugares de entrada*. *O* define, para cada transição t_j, o conjunto $O(t_j)$, de *lugares de saída* para a transição.

Esses quatro itens definem a estrutura de uma Rede de Petri. Formalmente, uma Rede de Petri é definida como uma quádrupla C, onde:

$C = (P,T,I,O)$
$P = \{ p_1, ..., p_n \}$ $n \geq 0$ (finito)
$T = \{ t_1, ..., t_m \}$ $m \geq 0$ (finito) $\qquad P \cap T = \phi$
$I = T \rightarrow P^{\infty}$
$O = T \rightarrow P^{\infty}$

A Rede de Petri pode ser representada graficamente, onde um *círculo* representa um lugar e uma *barra* representa uma transição. As funções de entrada e saída são representadas por arcos direcionados dos lugares para as transições e das transições para os lugares, respectivamente.

A execução de uma Rede de Petri é controlada pela posição e movimentação de marcas, chamadas *tokens*, na Rede. Os tokens são representados por pequenos círculos pretos que "residem" nos círculos correspondentes aos lugares da Rede; os tokens são movidos pelo *disparo* das transições na Rede. A transição deve estar *habilitada* para que possa disparar e isso acontece quando todos os seus lugares de entrada possuem tokens. A transição dispara removendo os tokens de seus lugares de entrada, gerando novos tokens que são depositados em todos os seus lugares de saída. A distribuição dos tokens na Rede de Petri define o estado da Rede e é chamada *marcação*, a qual é denotada pelo vetor $\mu = (\mu_1, \mu_2, ..., \mu_n)$, que fornece, para cada lugar da rede, o número de tokens nesse lugar, ou seja, $\mu(p_i) = \mu_i$.

A seguir, na Figura 3.1, apresenta-se uma Rede de Petri que especifica um protocolo nível 3 [TAN89], a qual é utilizada como exemplo, para explorar as idéias apresentadas neste trabalho.

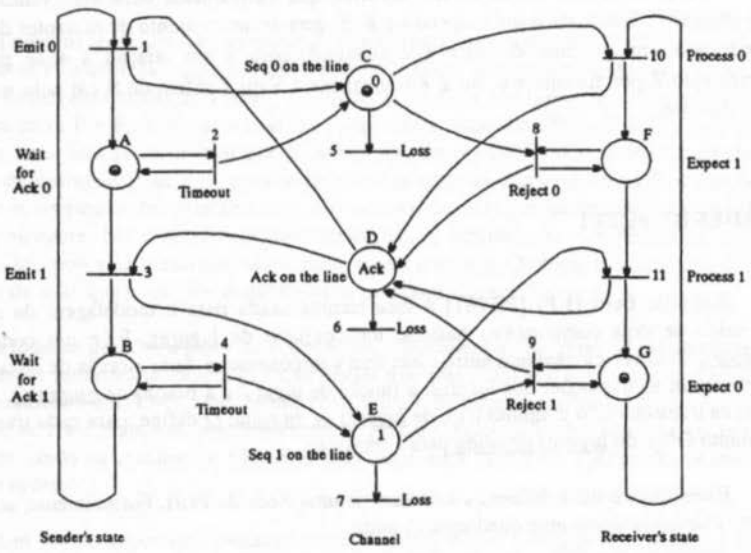


Figura 3.1 - Modelagem do Protocolo Nível 3 atavés de uma Rede de Petri [TAN89]

4. APLICAÇÃO DO CRITÉRIO ANÁLISE DE MUTANTES EM REDES DE PETRI

Nesta seção apresentam-se a definição do conjunto de operadores de mutação para Redes de Petri e uma análise dos resultados obtidos da aplicação manual do critério Análise de Mutantes no exemplo da Figura 3.1.

4.1. PROJETO DOS OPERADORES DE MUTAÇÃO PARA REDES DE PETRI

No projeto dos operadores de mutação para MEF tomou-se por base a classificação dos erros, sugerida por Chow [CHO78]. Essa classificação considera M e M' Máquinas de Estado Finito, onde M é o modelo correto e M' uma possível implementação, e define os erros da seguinte forma: 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).

No projeto dos operadores de mutação para Redes de Petri procurou-se tomar por base a classificação sugerida por Chow [CHO78] para MEF, apesar de não existir uma correspondência direta. Por exemplo, um erro de transferência numa Rede de Petri poderia estar associado à função de saída da mesma; assim, poderiam ser estabalecidos os operadores input/output-faltando, input/output-extra, input/output-trocado, input/output-deslocado, em relação a uma dada transição. Por exemplo, considerando o protocolo da Figura 3.1, obter-se-ia o mutante da Figura 4.1a, que reflete um erro de especificação sintetizado pelo operador *input-faltando*, que do ponto de vista semântico da aplicação corresponde a possibilitar a emissão de um novo quadro de mensagens sem o recebimento prévio do acknowledgement relativo à transmissão de um quadro anterior.

Assim, com base nessas diretrizes, estabeleceu-se um conjunto preliminar de operadores de mutação para Redes de Petri; esses operadores são definidos formalmente a seguir. Apesar de ser um conjunto preliminar, evidências são fornecidas de que a atividade de validação seria aprimorada com o uso do critério Análise de Mutantes neste contexto. Deve-se ter em mente, no entanto, que um estudo mais criterioso dos erros típicos cometidos por um especificador que utilize a técnica Redes de Petri deve ser realizado, almejando refinar e complementar o conjunto de operadores ora proposto, visto que este é um ponto relevante para a aplicação bem sucedida do critério Análise de Mutantes:

Op.1 - Input-faltando define-se por:

para toda t_j, $0 \leq j \leq m$, definem-se M_r mutantes, $0 \leq r \leq |\, I(t_j) \,|$ tal que

$|\, I(t_j) \,|_r = (\, |\, I(t_j) \,| - 1 \,)$ onde

$I(t_j)_r = I(t_j) - \{p_i\}$, para cada $p_i \in I(t_j)$

anotação: os demais operadores devem ser interpretados de forma similar a esta interpretação dada para o operador 1: "para toda transição t_j da RP, definir r mutantes, com r variando entre zero e a cardinalidade do conjunto I de lugares de entrada da transição, tal que a cardinalidade de I da transição t_j no mutante M_r é igual à cardinalidade de I desta transição na RP original, decrescida de um; ou seja, cada mutante é gerado pela exclusão de um lugar de entrada da transição t_j."

Considere a transição 3 do exemplo da Figura 3.1. $I(3) = \{A, D\}$ e $|\, I(3) \,| = 2$. Assim, dois mutantes seriam gerados pela aplicação do operador de mutação input-faltando. Um desses mutantes é ilustrado na Figura 4.1a, onde foi eliminado o lugar D de I(3), sendo agora a cardinalidade da transição 3 no mutante, denotada por $I(3)_m$, igual a $|\, I(3) \,| - 1$, ou seja $I(3)_m = 1$.

Op.2 - Input-extra define-se por

para toda t_j, $0 \le j \le m$, definem-se M_r mutantes, $0 \le r \le (n - | I(t_j) |)$ tal que

$| I(t_j) |_r = (| I(t_j) | + 1)$ onde

$I(t_j)_r = I(t_j) + \{p_i\}$, para cada p_i tal que $p_i \in P$ e $p_i \notin I(t_j)$

Op.3 - Input-deslocado define-se por

para toda t_j, $0 \le j \le m$, definem-se M_r mutantes, $0 \le r \le (| I(t_j) | * (m - 1))$ tal que

$| I(t_j) |_r = | I(t_j) - 1 |$, onde $I(t_j)_r = I(t_j) - \{p_k\}$ e

$| I(t_i) |_r = | I(t_i) + 1 |$, com

$I(t_i)_r = I(t_i) + \{p_k\}$, para todo $i \neq j$ tal que $p_k \notin I(t_i)$

Op.4 - Input-trocado define-se por

para toda t_j, $0 \le j \le m$, definem-se M_r mutantes, $0 \le r \le | I(t_j) | * (n - | I(t_j) |)$ tal que

$| I(t_j) |_r = | I(t_j) |$

$I(t_j)_r = I(t_j) - \{p_i\} + \{p_k\}$, para cada $p_i \in I(t_j)$ e
para cada $p_k \in P$ e $p_k \notin I(t_j)$

Op.5 - Output-faltando define-se por

para toda t_j, $0 \le j \le m$, definem-se M_r mutantes, $0 \le r \le | O(t_j) |$ tal que

$| O(t_j) |_r = (| O(t_j) | - 1)$ onde

$O(t_j)_r = O(t_j) - \{p_i\}$, para cada $p_i \in O(t_j)$

Op.6 - Output-extra define-se por

para toda t_j, $0 \le j \le m$, definem-se M_r mutantes, $0 \le r \le (n - | O(t_j) |)$ tal que

$| O(t_j) |_r = (| O(t_j) | + 1)$ onde

$O(t_j)_r = O(t_j) + \{p_i\}$, para cada p_i tal que $p_i \in P$ e $p_i \notin O(t_j)$

Op.7 - Output-deslocado define-se por

para toda t_j, $0 \le j \le m$, definem-se M_r mutantes, $0 \le r \le (|O(t_j)| * (m - 1))$ tal que

$|O(t_j)|_r = |O(t_j) - 1|$, onde $O(t_j)_r = O(t_j) - \{p_k\}$ e

$|O(t_i)|_r = |O(t_i) + 1|$, com

$O(t_i)_r = O(t_i) + \{p_k\}$, para todo $i \neq j$ tal que $p_k \notin O(t_i)$

Op.8 - Output-trocado define-se por

para toda t_j, $0 \le j \le m$, definem-se M_r mutantes, $0 \le r \le |O(t_j)| * (n - |O(t_j)|)$ tal que

$|O(t_j)|_r = |O(t_j)|$

$O(t_j)_r = O(t_j) - \{p_i\} + \{p_k\}$, para cada $p_i \in O(t_j)$ e
para cada $p_k \in P$ e $p_k \notin O(t_j)$

Op.9 - Alteração-da-marcação-inicial definido por

Seja a marcação inicial $\mu^0 = (\mu_1, \mu_2, ..., \mu_n)$ onde $\mu_i = \mu(p_i)$

Definem-se os mutantes das seguintes maneiras:

a) Seja $P' = \{ p_j \in P$ tal que $\mu(p_j) = 0\}$ e $|P'| = k$

definem-se M_r mutantes, $0 \le r \le k$ onde

Para cada $p_j \in P'$ gera-se um mutante tal que

$\mu^0_r = (\mu_{1r}, \mu_{2r}, ..., \mu_{nr})$ onde $\mu_{jr} = \mu(p_j)$, para $j \neq i$ e $\mu_{ir} = 1$

b) Seja $P'' = \{ p_j \in P$ tal que $\mu(p_j) = 1\}$ e $|P''| = s$

definem-se M_r mutantes, $0 \le r \le s$ onde

Para cada $p_j \in P''$ gera-se um mutante tal que

$\mu^0_r = (\mu_{1r}, \mu_{2r}, ..., \mu_{nr})$, onde $\mu_{jr} = \mu(p_j)$, para $j \neq i$ e $\mu_{ir} = 0$

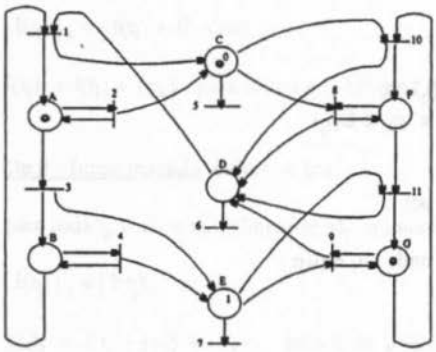
c) Definem-se M_r mutantes, $0 \leq r \leq (k*s)$ tal que

Para cada (p_i, p_j), $p_j \in P^n$ e $p_j \in P^r$, gera-se um mutante tal que

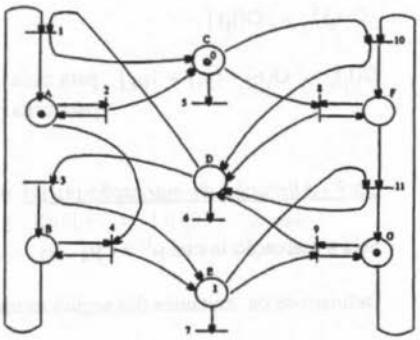
$\mu^0_r = (\mu_{1r}, \mu_{2r}, ..., \mu_{nr})$, onde $\mu_{kr} = \mu(p_k)$, para $k \neq i$ e $k \neq j$ e

$$\mu_{ir} = \mu(p_j) = 0 \quad e \quad \mu_{jr} = \mu(p_i) = 1$$

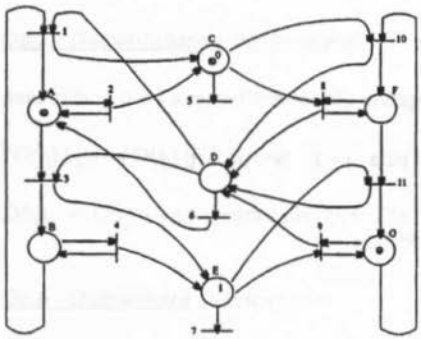
Na Figura 4.1 apresentam-se exemplos da aplicação desses operadores sobre a Rede de Petri apresentada na Figura 3.1. Observa-se na Figura 4.1 que: em (a) excluiu-se D de I(3); em (b) excluiu-se A de I(3) e incluiu-se A em I(4); em (c) incluiu-se A em O(6); e em (d) excluiu-se D e incluiu-se B em O(9).



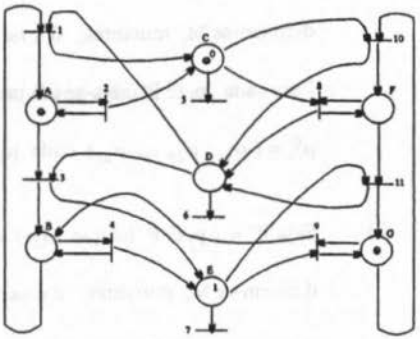
(a) op. input-faltando



(b) op. input-deslocado



(c) op. output-extra



(d) op. output-trocado

Figura 4.1 - Exemplos da Aplicação dos Operadores de Mutação para a Rede de Petri da Figura 3.1

4.2. APLICAÇÃO DA ANÁLISE DE MUTANTES E ANÁLISE DOS RESULTADOS

Com base nos operadores definidos na seção anterior, a partir do protocolo Nível 3 da Figura 3.1 foram geradas 643 Redes de Petri mutantes, conforme pode ser observado na Tabela 4.1. Não foi aplicado o operador input-deslocado às transições 5, 6 e 7, pois geravam mutantes com transição sem lugares de entrada. Por outro lado, não foram desconsiderados mutantes que pertencem a classes de Redes de Petri estendidas como "generalized Petri Nets" [PET81], o que poderia contribuir para reduzir o número de mutantes gerados e conseqüente redução do custo de aplicação do critério Análise de Mutantes. Alguns dos mutantes gerados foram ilustrados na Figura 4.1. Deve-se retomar que o 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 uma tentativa preliminar nesta direção. Apesar deste fato, os operadores utilizados neste trabalho estariam incluídos, possivelmente com pequenas reformulações, em um conjunto mais abrangente de operadores de mutação para Redes de Petri.

Considere inicialmente o conjunto T-inic ={ (10, 3, 11, 1), (10, 3, 11, 1, 2 ,5, 10, 2, 6, 8, 3, 7, 4, 11, 4, 9)} de seqüências de teste caracterizado por uma seqüência normal de operação, sem erros de transmissão e por uma seqüência que além de refletir problemas de comunicação, provocando a perda e retransmissão de quadros de mensagens e de ack, garante também o disparo de todas as transições da Rede de Petri considerada.

Para essas seqüências o modelo comportou-se de acordo com as expectativas, ou seja, refletiu o comportamento esperado do sistema a ser implementado. A adequação de T-inic em relação à Análise de Mutantes, considerando o conjunto de operadores de mutação definido na seção anterior, pode ser extraída da Tabela 4.1. Desta tabela observa-se que 13.84% dos mutantes permanecem vivos (um escore de mutação de 86.16%), ou porque as seqüências utilizadas não são adequadas para distingüir o comportamento da rede inicial em relação à rede mutante, ou porque a rede e o mutante em questão são equivalentes, ou seja, têm o mesmo comportamento para qualquer seqüência possível de execução da rede. No primeiro caso equivaleria dizer que se fosse tomado o mutante como a especificação "correta", as seqüências utilizadas não seriam sensíveis o suficiente para revelar esse tipo de erro. Assim, com base no conjunto de mutantes vivos, a massa de teste pode ser aprimorada levando a uma atividade de validação de melhor qualidade. Para exemplificar, considere o mutante da Figura 4.1b gerado pela aplicação do operador *input-deslocado*. Para "matar" este mutante seria necessário inserir, por exemplo, a seqüência de teste (2, 5, 2, 10, 6, 2, 8, 3, 7, 4, 11, 6, 4, 9, 1), que simula seqüências de eventos que caracterizam canais de comunicação de péssima qualidade, o que implicaria na transmissão e retransmissão de quadros de mensagens e de acknowledgments.

Para identificar se um determinado mutante apresenta um comportamento distinto do comportamento da Rede de Petri original verificou-se a marcação apresentada pelo mutante e pela Rede de Petri após a execução da seqüência de teste utilizada, sem diferenciar entre um ou mais tokens presentes nos lugares, ou seja, multiplos tokens em um determinado lugar é equivalente à ocorrência de um único token. Assim, para que uma marcação μ seja distingüida de uma marcação μ' é necessário que para algum lugar p_i, $\mu(p_i) = 0$ e $\mu'(p_i) \neq 0$, ou vice-versa. Esta abordagem poderia ser reformulada se fosse requerido que a Rede de Petri em

questão apresentasse uma propriedade específica ou pertencesse a uma dada classe de redes, como por exemplo, "safe nets" ou "k-bounded nets". Assim, os mutantes que apresentassem uma marcação final ou mesmo intermediária que não preenchessem tais características poderiam ser considerados mortos.

A comparação das marcações somente após a execução de toda a seqüência de teste, sem diferenciar um ou mais tokens e sem utilizar marcações intermediárias, afeta o escore de mutação e dificulta a eliminação dos mutantes, o que pode favorecer o aprimoramento da massa de teste e, conseqüentemente, o aumento de qualidade dessa atividade. Por exemplo, considere a seqüência s = (10, 3, 11, 1) de T-inic que ocorre no início da outra seqüência definida em T-inic; à primeira vista o testador poderia pensar em excluir s de T-inic. Neste caso, teria obtido um score de mutação inferior, de 78.7 % (21.3% dos mutantes permaneceriam vivos), o que implicaria em uma maior possibilidade de se aceitar como certa uma especificação errônea, pois o número de mutantes que teriam permanecido vivos teria sido maior. A massa de teste não teria sido sensível o suficiente para diferenciar esses mutantes, ou melhor dizendo, revelar os tipos de erros sintetizados por eles.

Tabela 4.1 - Síntese dos Resultados Obtidos na Aplicação da Análise de Mutantes no Exemplo da Figura 3.1, com o Conjunto de Teste T-inic

Operador	Total	Mortos	Vivos
Op.1 Input-faltando	17	13	4
Op.2 Input-extra	60	59	1
Op.3 Input-deslocado	140	113	27
Op.4 Input-trocado	90	89	1
Op.5 Output-faltando	16	15	1
Op.6 Output-extra	61	41	20
Op.7 Output-deslocado	160	126	34
Op.8 Output-trocado	80	79	1
Op.9 Alteração-da-marcação-inicial	19	19	0
Total	643	554	89

O conjunto T-inic de seqüências de teste utilizado para a aplicação da Análise de Mutantes foi aprimorado procurando refletir aspectos semânticos da aplicação considerada, identificando-se seqüências e situações típicas: seqüências de transmissão com retransmissão do pacote número 0; seqüências de transmissão com retransmissão do pacote número 1; e seqüências de transmissão com problemas nos canais de transmissão e recepção, levando a retransmissões dos pacotes e respectivos quadros de acknowledgment. Assim, a partir de T-inic obtém-se o conjunto T-fim constituído pelas seguintes seqüências típicas:

T-fim = { (10, 3 11, 1), (10, 3, 11, 1, 2, 5, 10, 2, 6, 8, 3, 7, 4, 11, 4, 9), (2, 5, 10, 3, 11, 1), (10, 6, 2, 8, 3, 11, 1), (10, 3, 4, 7, 11, 1), (10, 3, 11, 6, 4, 9, 1), (2, 5, 10, 3, 4, 7, 11, 1), (10, 6, 2, 8, 3, 11, 6, 4, 9, 1), (2, 5, 2, 10, 6, 2, 8, 3, 7, 4, 11, 6, 4, 9, 1)}

Com o conjunto de seqüências T-fim, mais completo e refletindo ocorrências típicas da natureza da aplicação considerada, obtém-se, conforme ilustrado na Tabela 4.2, um escore de mutação perto de 95%, ou seja, apenas 5.44% dos mutantes (35) permanecem vivos. Assim, em princípio, dever-se-ia acrescentar ao conjunto T-fim outras seqüências com o propósito de "matar" os mutantes que permaneceram vivos, se possível, pois, como estabelecido anteriormente, entre eles pode haver mutantes que sejam equivalentes à Rede de Petri original.

A questão de equivalência é um ponto relevante para a aplicação mais efetiva e automatização do critério Análise de Mutantes. No escopo deste trabalho este aspecto não foi considerado e deverá ser abordado em trabalhos futuros.

Tabela 4.2 - Síntese dos Resultados Obtidos na Aplicação da Análise de Mutantes no Exemplo Considerado com o Conjunto de Teste T-fim

Operador	Total	Mortos	Vivos
Op.1 Input-faltando	17	15	2
Op.2 Input-extra	60	60	0
Op.3 Input-deslocado	140	127	13
Op.4 Input-trocado	90	90	0
Op.5 Output-faltando	16	16	0
Op.6 Output-extra	61	52	9
Op.7 Output-deslocado	160	149	11
Op.8 Output-trocado	80	80	0
Op.9 Alteração-da-marcação-inicial	19	19	0
Total	643	608	35

Fica assim evidente que o uso da Análise de Mutantes no contexto de Redes de Petri, proposto neste trabalho, fornece informações adicionais à validação de Redes de Petri, que complementam as atividades realizadas e informações obtidas decorrentes do uso de técnicas como simulação e análise de alcançabilidade, entre outras.

5. CONCLUSÕES

Os resultados apresentados neste trabalho fornecem evidências, em concordância com os resultados obtidos para MEF [FAB93], de que o uso do critério Análise de Mutantes no contexto de Redes de Petri contribui para o aprimoramento da atividade de validação, além de fornecer um mecanismo para a quantificação da qualidade desta atividade, em sintonia com a constante preocupação neste sentido reinante na área de Engenharia de Software [PRE92]; motiva ainda a continuidade destes estudos, que deverá culminar na especificação e

implementação de uma ferramenta de suporte, com características funcionais semelhantes à ferramenta PROTEUM/FSM [FAB94a].

Com base nos trabalhos já conduzidos pertinentes à especificação e implementação da PROTEUM/FSM é simples visualizar uma ferramenta de suporte à aplicação da Análise de Mutantes no contexto de Redes de Petri, que poderia ser denominada desde já de PROTEUM/PN. Essencialmente, seria necessário estender a LES — Linguagem de Especificação de Statecharts — [MAS91] para apoiar a especificação de Redes de Petri. A partir da LES estendida, a exemplo do que ocorre na PROTEUM/FSM, seriam gerados os mutantes. Feita essa tarefa, bastaria incluir no ambiente um simulador de Redes de Petri, pois todos os demais recursos, eventualmente disponíveis na PROTEUM/FSM, seriam plenamente reutilizados com pequeno esforço de adequação, como por exemplo o módulo de visualização de mutantes, em desenvolvimento. A implementação de uma ferramenta viabilizaria que um estudo mais abrangente (benchmark) fosse conduzido para validar as conjecturas estabelecidas no escopo deste trabalho.

A continuidade deste trabalho do ponto de vista teórico consiste, em uma primeira etapa, revisar e complementar o conjunto de operadores de mutação definido e utilizado neste trabalho. Por exemplo, os operadores transição-faltando, lugar-faltando e lugar-extra deverão ser definidos e incluídos no atual conjunto de operadores. Classes específicas de Redes de Petri, ou extensões destas, podem levar à caracterização de outros operadores, ou eventualmente à eliminação de alguns. No mesmo sentido se aplicam considerações deste gênero relativas a propriedades de Redes de Petri. Dentro deste mesmo espírito, a análise dos mutantes que caracteriza um mutante como vivo, morto, anômalo ou equivalente pode ser revisitada. Ainda, a hipótese de acoplamento deverá ser explorada e validada no contexto de Redes de Petri.

Para concluir, dois outros pontos deverão ser priorizados: o estabelecimento de heurísticas para determinação de Redes de Petri equivalentes e o estudo de estratégias alternativas para a aplicação do critério Análise de Mutantes, como por exemplo, a minimização do número de mutantes gerados, pela caracterização de operadores mais efetivos ou pela aplicação de somente um percentual dos mutantes gerados, objetivando a minimização do custo de utilização deste critério, sem no entanto, comprometer a qualidade da atividade de validação.

REFERÊNCIAS BIBLIOGRÁFICAS

[BUD80] Budd, T.A.; DeMillo, R.A.; Lipton, R.J.; Sayward, F.G. *Theoretical and Empirical Studies on Using Prog Mutation to Test the Functional Correctness of Prog.*, 7th ACM Symposium on Principles of Programming Languages, jan., 1980.

[CHO78] Chow, T.S. *Testing Software Design Modeled by Finite-State Machines.* IEEE Transactions on Software Engineering, SE(4(3)), pp. 178-187, 1978.

[DEM78] 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.

[DEM80] DeMillo, R.A. *Mutation Analisys as a Tool for Software Quality Assurance*, Proc. of COMPSAC 80, Chicago-IL, outubro, 1980.

[FAB93] Fabbri, S.C.P.F.; Maldonado, J.C.; Masiero, P.C.; Delamaro, M.E. *Análise de Mutantes Baseada em Máquinas de Estado Finito*, in Anais do 11º Simpósio Brasileiro de Redes de Computadores, Campinas 1993.

[FAB94a] Fabbri, S.C.P.F.; Delamaro, M.E.; Maldonado, J.C.; Masiero, P.C. *Proteum/FSM - Uma Ferramenta para Apoiar a Validação de Máquinas de Estado Finito pelo Critério Análise de Mutantes*, in Anais do 12º Simpósio Brasileiro de Redes de Computadores, Curitiba, 1994.

[FAB94b] Fabbri, S.C.P.F.; Maldonado, J.C.; Masiero, P.C.; Delamaro, M.E. *Mutation Analysis Testing for Finite State Machines*, trabalho aceito no Fifth International Symposium on Software Reliability Engineering, California, novembro 1994.

[FUJ91] Fujiwara,S.; Bochmann,G.V.; Khendek,F.; Amalou,M.; Ghedamsi,A., *Test Selection Based on Finite State Models*, IEEE Trans. on Software Eng., Vol. 17, N. 6, June 1991.

[GIL62] Gill, A. *Introduction to the Theory of Finite-State Machines*. New York, McGraw-Hill, 1962.

[HAR87] Harel, D., *Statecharts: A Visual Formalism for Complex Systems*, Science of Computer Programming, 1987.

[MAS91] 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 XI Congresso Nacional da SBC, XVIII SEMISH, Santos, pp. 45-61,5-9 Agosto 1991.

[NAI81] Naito, S.; Tsunoyama, M. *Fault Detection for Sequential Machines by Transition-Tours*, in Proceedings FTCS (Fault Tolerant Comput. Systems), pp 238-243, 1981.

[PET81] Peterson, J.L. *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

[PRE92] Pressman, R.S. *Software Engineering - A Practitioner's Approach*, (3rd. edition), McGraw-Hill, 1992.

[SAB88] Sabnani, K.K.; Dahbura, A.T. *A Protocol Testing Procedure*, Comput. Networks and ISDN Syst., Vol. 15, N. 4, pp. 285-297, 1988.

[TAN89] Tanenbaum, A.S. *Computer Networks*, (2nd. edition), Prentice Hall, 1989.