APLICAÇÃO DO CRITÉRIO ANÁLISE DE MUTANTES NA VALIDAÇÃO DE ESPECIFICAÇÕES BASEADAS EM STATECHARTS

Sandra C.P.F. Fabbri
UFSCar[1] - e.mail sandraf@dc.ufscar.br
José Carlos Maldonado
ICMSC-USP[2] - e.mail jcmaldon@icmsc.sc.usp.br
Paulo Cesar Masiero
ICMSC-USP[2] - e.mail masiero@icmsc.sc.usp.br

RESUMO

A qualidade da atividade de VV&T – Verificação, Validação e Teste – de especificações é fundamental no processo de desenvolvimento de software. Técnicas e critérios de teste têm sido investigados no contexto de VV&T de especificações do aspecto comportamental de Sistemas Reativos fornecendo mecanismos para a avaliação da qualidade dessa atividade.Este artigo apresenta a aplicação do critério de teste Análise de Mutantes, um critério desenvolvido originalmente para o teste de programas, para a atividade de teste de especificações que utilizam a técnica Statecharts. A aplicação desse critério é baseada em um conjunto de operadores de mutação. Assim, apresentam-se os operadores de mutação definidos para Statecharts, que estão divididos em três classes, bem como estratégias de abstração de componentes de Statecharts que permitem conduzir a atividade de validação de forma incremental, explorando aspectos particulares dessa técnica, como hierarquia, ortogonalidade e história.

Palavras Chaves: Análise de Mutantes, Statecharts, Validação, Sistemas Reativos, Teste de Software.

ABSTRACT

The quality of VV&T – Verification, Validation and Testing – activity is extremely relevant to the software development process. Testing techniques and criteria have been investigated in the context of VV&T of Reactive System behavioral aspect specifications providing mechanisms to the VV&T activity quality assessment. This paper presents the application of the Mutation Analysis criterion, which was originaly developed for program testing, to the testing and validation of Statechart based specifications. The application of this criterion is dependent on the mutation operators set. Therefore it is defined a mutation operator set for this technique, that is divided in three classes, as well as Statechart component abstraction strategies which enable to apply incrementally the testing and validation activity exploring specifics aspects of the Statecharts like hierarchy, orthogonality and history.

Keywords: Mutation Analysis, Statecharts, Validation, Reactive System, Software Testing.

1 Universidade Federal de São Carlos - Departamento de Computação - Cx.Postal 676, 13565-905 - São Carlos, SP
2 Instituto de Ciências Matemáticas de São Carlos - USP - Cx. Postal 668, 13560-970 - São Carlos, SP

1. INTRODUÇÃO

Software é atualmente um componente básico e necessário em várias atividades humanas. Muitos dos sistemas desenvolvidos são bastante complexos e podem envolver riscos à vida humana, exigindo, portanto, um desenvolvimento criterioso e de alta qualidade, para que esses riscos sejam minimizados (Pressman, 1992). Dentre os vários tipos em que o software pode ser classificado, os Sistemas Reativos, cuja característica principal é reagir continuamente a estímulos, possuem um aspecto bastante crítico pois usualmente controlam atividades humanas essenciais, como por exemplo, controle metroviário, controle de monitoramento hospitalar, controle de tráfego aéreo ou protocolos de comunicação. Falhas nesses sistemas podem envolver grandes riscos à vida ou ao patrimônio. Especialmente nos casos em que o risco à vida humana é bastante alto, torna-se imprescindível um maior rigor no processo de desenvolvimento, conseguido através do uso disciplinado de técnicas especializadas, seguras e eficientes na produção de software (Harel, 1997a).

Acompanhando todo o processo de produção de software, é realizado um conjunto de atividades de apoio, denominado *Garantia de Qualidade de Software*, para assegurar que a qualidade seja mantida em cada passo desse processo. Muitas dessas atividades são atividades de verificação e validação, que têm a atividade de teste como elemento crítico para a garantia da qualidade do software. Em geral, o custo da atividade de teste é bastante alto em relação ao custo total de desenvolvimento. Assim, para se conseguir maior qualidade e produtividade na atividade de teste, têm sido definidos vários *critérios de teste*, uma grande parte utilizada basicamente em nível de programas. Um critério estabelece requisitos que devem ser observados na atividade de teste. Através da análise do quanto esses requisitos são satisfeitos – análise de cobertura – obtém-se uma maneira de quantificar essa atividade.

Os critérios de teste estão agrupados em três técnicas: a *funcional*, a *estrutural* e a *baseada em erros*. A principal diferença entre essas técnicas é a fonte de informação de onde são extraídos os requisitos de teste. Na funcional, os requisitos são extraídos da própria especificação do software; na estrutural, eles são extraídos de uma particular implementação e na baseada em erros, os requisitos são construídos com base nos erros típicos e comuns cometidos no processo de desenvolvimento (Maldonado, 1991). Um dos principais critérios da técnica de teste baseada em erros é o critério *Análise de Mutantes*. Salienta-se que para programas, os critérios de teste existentes são complementares e devem ser aplicados de maneira conjunta, para aumentar a qualidade da atividade de teste (Pressman, 1992).

Também para as outras fases do processo de desenvolvimento de software, a Engenharia de Software recomenda que o projetista utilize métodos para apoiar seu trabalho, visando a aumentar a qualidade e a minimizar a inserção de erros. Esses métodos são, em geral, classificados como métodos formais e informais. Os métodos informais não possuem rigor matemático e geram, normalmente, grande ambigüidade. Os métodos formais são aqueles que possuem uma base matemática dada, em geral, por uma linguagem formal de especificação. Eles permitem que o projetista possa, de forma sistemática, especificar, desenvolver e verificar o software, ou parte dele. Em geral, os métodos formais são utilizados para especificar o aspecto comportamental e propriedades estruturais do software. Quando utilizados em fases iniciais do desenvolvimento do software, eles podem detectar falhas que só seriam reveladas nas fases de teste e depuração e, quando usados em fases mais adiantadas do desenvolvimento, ajudam a determinar a corretitude de uma implementação e a equivalência de diferentes implementações (Wing, 1990).

As técnicas formais Máquina de Estados Finitos (MEF) (Gill, 1962), Redes de Petri (Peterson, 1977) e Statecharts (Harel, 1987a, 1987b) são intensamente utilizadas na

especificação do aspecto comportamental de Sistemas Reativos. Para apoiar as atividades de teste de tais especificações, que devem ser conduzidas com bastante rigor e critério, vários métodos e abordagens são propostos na literatura. No caso da técnica Statecharts, são propostas várias abordagens de simulação, que são consideradas formas de validação de Sistemas Reativos em geral e que necessitam de ferramentas para que possam ser utilizadas (Harel, 1992). Uma questão que não é considerada por essas formas de validação, mas que é relevante para a atividade de teste, é o aspecto de cobertura. Petrenko e Bochmann (1996) abordam esse aspecto em relação às especificações baseadas em Máquinas de Estados Finitos, salientando a relevância de pesquisas nessa direção, uma vez que com a análise de cobertura pode-se quantificar a qualidade da atividade de teste.

Assim, com o objetivo de contribuir com novos mecanismos para facilitar o desenvolvimento de sistemas e considerando que, para programas, os critérios de teste são complementares, explorou-se a aplicação do critério Análise de Mutantes na validação em nível de especificações. Já foram conduzidos dois experimentos referentes à aplicação do critério Análise de Mutantes para especificações: um deles em Máquinas de Estados Finitos (Fabbri et al, 1993; Fabbri et al, 1994b) e o outro em Redes de Petri (Fabbri et al, 1994c; Fabbri et al, 1995a). Os resultados desses experimentos evidenciam o aspecto complementar da aplicação da Análise de Mutantes em relação aos métodos de validação propostos para essas técnicas. Assim, pelo fato da técnica Statecharts ser uma extensão das MEFs, por ser uma técnica mais completa, o que diminui a capacidade de analisá-la e de verificar suas propriedades, e por ser também bastante utilizada para a especificação do aspecto comportamental de Sistemas Reativos, apresentam-se neste artigo os mecanismos necessários para a aplicação do critério Análise de Mutantes em especificações baseadas em Statecharts .

O artigo está organizado da seguinte maneira: na Seção 2 apresentam-se os principais conceitos do critério Análise de Mutantes; na Seção 3, a técnica Statecharts e as definições necessárias para entendimento dos mecanismos propostos; na Seção 4 apresenta-se a aplicação do critério Análise de Mutantes no contexto de Statecharts e, na Seção 5, apresentam-se as conclusões.

2. ANÁLISE DE MUTANTES

A Análise de Mutantes surgiu na década de 70 na Yale University e Georgia Institute of Tecknology (DeMillo et al, 1978); tem-se mostrado, através de trabalhos empíricos e teóricos, ser um critério atrativo para o teste de programas (DeMillo, 1980; Budd et al, 1980; Horgan e Mathur, 1992). 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 no contexto de teste e validação de programas (Choi et al, 1989, Mathur; Krauser, 1988), minimizando o custo de sua aplicação.

Para a aplicação desse critério na validação do aspecto comportamental de Sistemas Reativos foi necessário estabelecer um paralelo entre o nível de programas e o nível de especificação, no que diz respeito às hipóteses básicas do critério, que são: a hipótese do programador competente e o efeito de acoplamento. Dessa forma, estabeleceu-se, no nível de especificação, a *hipótese do especificador competente* que considera que uma especificação produzida por um especificador competente ou está correta ou está próxima do correto. Assume-se a validade do *efeito de acoplamento* que considera que um caso de teste capaz de detectar um erro simples é também capaz de detectar erros mais complexos. Em nível de programas, essa hipótese é verificada através de alguns trabalhos, como por exemplo, (Offutt, 1992), o que justifica a geração de mutantes de primeira ordem, isto é, que possuem apenas

uma alteração em relação ao programa em teste. Essas hipóteses servem para delimitar a vizinhança das especificações mutantes que serão geradas durante a aplicação do critério.

Basicamente, a idéia da Análise de Mutantes é criar a confiança de que uma especificação S está correta produzindo-se, através de pequenas alterações, um conjunto de especificações, chamadas de *mutantes*, semelhantes a S, e construindo-se casos de teste capazes de provocar diferenças de comportamento entre S 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 na especificação. A escolha e definição dos operadores de mutação é um ponto importante para a aplicação da Análise de Mutantes e a delimitação do conjunto de operadores baseia-se nas hipóteses apresentadas anteriormente.

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

Todos os mutantes são executados usando-se os casos de teste T como entradas. Se um mutante S_i apresenta resultados diferentes de S diz-se que esse mutante está *morto*; nesse caso, T conseguiu identificar o "erro" representado pelo mutante. Por outro lado, se S_i apresenta respostas idênticas a S, diz-se que ele continua *vivo*. Isto pode ocorrer por dois motivos: ou porque T não contém casos de teste capazes de distinguir S_i de S ou porque ambas as especificações 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 S. A questão de equivalência é, em geral, indecidível e a definição de heurísticas que auxiliem a análise do testador nesse sentido são de extrema relevância.

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 S, no sentido de que ou S está correta ou contém um erro sutil e inesperado, o que deve ser raro se as modificações usadas para criar os mutantes forem cuidadosamente escolhidas.

Um ponto importante destacado em (DeMillo, 1980) 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*, que relaciona o número de mutantes mortos com o número de mutantes gerados, pode-se avaliar a adequação dos casos de testes usados e, como conseqüência, a confiabilidade do programa testado. Assim, com o escore de mutação pode-se quantificar e avaliar a qualidade da atividade de teste.

3. STATECHARTS

A técnica Statecharts, proposta por Harel (1987a), é uma extensão das Máquinas de Estados Finitos, que procura justamente solucionar vários aspectos desta técnica que dificultam sua aplicação em sistemas complexos. Essa extensão corresponde à hierarquia dos estados, à capacidade de especificar paralelismo, ao mecanismo de comunicação através de "broadcasting", além de um conjunto de notações especiais que fornecem um maior poder de representação em relação às MEFs, possibilitando que sistemas complexos sejam descritos de forma bastante concisa. Por outro lado, como comentam Masiero et al (1994), esse poder de representação proporcionado pela técnica Statecharts torna mais difícil a atividade de validação das especificações em relação ao comportamento esperado. Essa técnica é resumida por Harel (1987a, 1987b) da seguinte maneira:

Statecharts = diagramas de estado + decomposição + ortogonalidade + broadcasting

Salienta-se nessa expressão que os Statecharts são, basicamente, diagramas de estados acrescidos de outras características, ou seja, que os componentes básicos de um Statecharts correspondem a Máquinas de Estados Finitos.

A idéia de *decomposição* permite especificar o sistema de forma modularizada, sendo que esta pode ocorrer através de dois tipos de estados: estados tipo OR e estados tipo AND.

A *ortogonalidade* corresponde à especificação de estados do tipo AND, isto é, estados que são compostos de subestados paralelos, que ficam ativos simultaneamente. Ao contrário, em estados tipo OR, apenas um pode estar ativo em cada instante, como é o caso das MEFs. Quando existem estados do tipo AND, os aspectos de sincronização entre eles são representados por algumas condições e eventos, que são avaliados no momento em que uma transição esteja habilitada, isto é, apta a ocorrer.

Segundo Harel (1987b), a forma de se modelar sincronização nos Statecharts é através de seus eventos de saída, um tipo de *ação*, que podem provocar o disparo de uma transição. No entanto, em contraste com as máquinas de Mealy, nos Statecharts as ações não são meramente enviadas, como simples saídas, para o exterior do sistema que está representado. Normalmente, as ações afetam o comportamento do próprio Statechart, nos componentes ortogonais, através de um mecanismo denominado *broadcasting* (reação em cadeia). A ação é considerada então como um *evento interno* que pode, possivelmente, causar outras transições em outros componentes. A *configuração* de estados do Statecharts corresponde ao conjunto de estados que estão ativos num determinado instante.

Outro aspecto da técnica Statecharts é sua capacidade de "lembrar" alguns estados que foram visitados previamente, à qual se dá o nome de *história*. Quando um estado é ativado, o subestado dele que se torna ativo é, normalmente, o subestado default, a menos que um símbolo de história *(H)* esteja associado à transição que leva até esse estado. Esse caso é denominado por Harel (1987a) de "*entrada por história*". Assim, em vez de ser ativado o subestado default, ficará ativado o último subestado que foi visitado anteriormente. Se um estado é decomposto em vários níveis e o aspecto de história for válido recursivamente para todos esses níveis da hierarquia, essa história é denotada por H^*.

Somando-se ao poder de representação, os Statecharts também possuem sintaxe e semântica bem definidas, permitindo a análise dos aspectos dinâmicos do modelo. De acordo com Harel (1992), verificar se o modelo está consistente e completo não previne a ocorrência de erros lógicos. É necessária a execução do modelo, cujo pré-requisito se resume na disponibilidade de uma semântica formal que contenha informação suficiente para definir de modo preciso o estado do sistema a cada passo da execução. Assim, os sistemas modelados através de técnicas que possuam esse formalismo, como é o caso dos Statecharts, são freqüentemente validados pelas seguintes formas de execução: interativa, em batch, programada e exaustiva. Ressalta-se que, através dessas formas de execução, a quantificação da atividade de teste é bastante deficiente e não é possível estabelecer precisamente um valor que retrate a qualidade dessa atividade e, conseqüentemente, do produto que está sendo avaliado.

A seguir, apresenta-se a sintaxe dos Statecharts necessária ao entendimento dos mecanismos propostos na seção seguinte:

Estados: o conjunto de estados, S, é definido junto com uma função hierárquica, ρ, uma função tipo, ψ, um conjunto de símbolos de história H e uma função default δ.

A ***Função Hierárquica*** $\rho : S \rightarrow 2^S$ define para cada estado os seus subestados. Tem-se que se $\rho(x) = \rho(y)$ então $x = y$.
Existe um único estado $r \in S$ tal que $\forall s \in S \; r \notin \rho(s)$; r é a raiz do Statecharts.
ρ^* e ρ^+ são extensões de ρ definidas por: $\rho^*(s) = \cup \rho^i(s)$, $i \geq 0$
$\rho^+(s) = \cup \rho^i(s)$, $i \geq 1$.

A ***Função Tipo*** $\psi : S \rightarrow \{AND, OR\}$ define para cada estado o seu tipo.
Se $\rho(s) \neq \phi$ e $\psi(s) = OR$ então $\rho(s)$ é uma decomposição *xor* de s. Isso significa que quando um sistema está no estado s ele está em um e somente um dos subestados de s.
Se $\rho(s) \neq \phi$ e $\psi(s) = AND$ então $\rho(s)$ é uma decomposição ***and*** de s. Isso significa que quando um sistema está no estado s ele está simultaneamente em todos os subestados de s.

O conjunto de ***Símbolos de História***, H, está relacionado ao conjunto de estados pela função $\gamma : H \rightarrow S$ tal que:
$\gamma(h1) = \gamma(h2)$ implica $h1 = h2$ e $\gamma(H)$ é um subconjunto do conjunto de estados OR, ou seja, somente um estado OR pode ter um símbolo de história associado a ele.
Define-se $\omega : S \cup H \rightarrow S$ por $\omega(z)$ é z se $z \in S$ e $\omega(z)$ é $\gamma(z)$ se $z \in H$.

A ***Função Default*** $\delta : S \rightarrow 2^{S \cup H}$, define para um estado s, um conjunto de estados e símbolos de história que estão contidos no estado.
Se $x \in \delta(s)$ então: se $x \in S$, $x \in \rho^+(s)$,
se $x \in H$, $\gamma(x) \in \rho^*(s)$.
$\delta(s)$ é o conjunto default para s.

Em seguida, são definidos alguns termos e conceitos usados na definição da semântica (Harel, 1987b), a qual não é apresentada neste trabalho. Os termos e conceitos apresentados a seguir são suficientes para o entendimento das estratégias apresentadas na próxima seção.

a) Um estado s é ***básico*** se $\rho(s) = \phi$

b) Para um conjunto de estados X, o ***Menor Ancestral Comum de X***, denotado por $LCA(X)$ é o estado x definido como segue: $LCA(X) = x$ se e somente se:

1. $X \subseteq \rho^*(x)$
2. $\forall s \in S$, $X \subseteq \rho^*(s) \Rightarrow x \in \rho^*(s)$

Para um conjunto de estados X, o ***Menor Ancestral Estrito Comum OR de X***, denotado por $LCA^*(X)$ é o estado x definido como segue: $LCA^*(X) = x$ se e somente se:

1. $X \subseteq \rho^+(x)$
2. $\psi(x) = OR$
3. $\forall s \in S$ se $\psi(s) = OR$ então $X \subseteq \rho^*(s) \Rightarrow x \in \rho^*(s)$

4. ANÁLISE DE MUTANTES NO CONTEXTO DE STATECHARTS

Como foi comentado na seção anterior, a validação de especificações Statecharts está baseada em diferentes tipos de simulação, que podem ser vistos como complementares: 1) *execução interativa*, que com o apoio de uma ferramenta automatizada, possibilita que o usuário represente o meio ambiente do sistema gerando, passo a passo, eventos que provocam alterações no estado do mesmo; 2) *execução em batch*, onde a execução é feita de forma iterativa e não mais interativa, com base em um conjunto de eventos pré-determinados; 3) *execução programada*, que permite analisar o modelo sob condições geradas randomicamente, e onde podem ser definidos pontos de parada a fim de que a ferramenta execute determinadas ações quando ocorrem situações específicas; 4) *execução exaustiva*, que consiste, teoricamente, em executar o modelo gerando-se todos os possíveis seqüências de eventos externos e todas as alterações nos valores das condições e variáveis; em geral, esse tipo de simulação é impraticável e deve ser considerada como uma alternativa no caso de partes pequenas, críticas e bem isoladas do modelo.

Analisando-se esses tipos de simulação observa-se que a execução exaustiva, em geral, é impraticável. As execuções interativa e em batch dependem da intervenção do usuário, que é responsável pela definição de uma seqüência de eventos que corresponda a um conjunto de teste, sendo que essa definição é feita exclusivamente com base no sentimento do usuário, em relação àquilo que deve ser testado. Na execução programada, os eventos são gerados segundo uma distribuição probabilística, o que não garante que uma determinada seqüência de eventos seja gerada em uma determinada execução.

Considerando-se que os Statecharts podem fazer uso de variáveis, certamente, para um sistema complexo que utilize um grande número delas, a identificação de um erro não seria uma tarefa simples em qualquer um dos tipos de simulação mencionados, pois aspectos de indecidibilidade e erros sensíveis a dados emergem nesse contexto.

Portanto, testar uma especificação com o objetivo de garantir que ela não possua determinados tipos de erros, aqueles mais relevantes e críticos para a aplicação que esteja sendo modelada, não é uma tarefa simples. Assim, a aplicação do critério Análise de Mutantes, que permite que os operadores de mutação sejam definidos de modo a retratar os erros que se desejam detectar na especificação em teste, atende a esse objetivo de forma mais eficiente, como é apresentado em (Fabbri, 1996). Nesse trabalho apresenta-se um exemplo que mostra que mesmo utlizando-se a execução programada para gerar uma seqüência de eventos para testar especificações mutantes, ainda assim, alguns mutantes podem permanecer vivos. Dessa forma, com o intuito de matar esses mutantes, o testador seria forçado a elaborar seqüências de teste específicas, melhorando a qualidade da massa de teste e, conseqüentemente, aumentando a confiança na especificação que está sendo testada, fazendo do critério Análise de Mutantes uma forma complementar de teste nesse contexto.

O critério Análise de Mutantes já foi aplicado pelos autores no contexto de Máquinas de Estados Finitos e Redes de Petri, através de experimentos conduzidos manualmente e, em ambos os casos, foram obtidas evidências do aspecto complementar do mesmo (Fabbri et al, 1993; 1994b; 1994c; 1995a). Outros autores como Probert e Guo (1991), Wang e Liu (1993), também utilizam mutações para o teste de MEFs. No entanto, não especificam formalmente um conjunto de operadores que gerem tais mutações, como é apresentado em Fabbri (1996). Mais especificamente, considerando o fato de que os Statecharts são uma extensão das MEFs, apresenta-se na Tabela 1 a síntese da aplicação do critério na validação de uma MEF que especifica um Protocolo de Transporte Classe 0 da ISO (Gabos e Stiubiener, 1990), ressaltando-se que, mesmo para essa técnica que possui vários métodos de geração de

seqüências de teste já consolidados, ainda assim o critério Análise de Mutantes pode atuar de forma complementar pois vários mutantes permanecem vivos após a aplicação de tais seqüências.

Tabela 1 - Resultados da Aplicação da Análise de Mutantes no contexto de MEF

			método W		método TT	
operador	total	equiv.	vivos	mortos	vivos	mortos
1. alteração-do-estado-inicial	3	0	0	3	0	3
2. arco-faltando	9	1	0	8	0	8
3. evento-faltando	21	3	0	18	0	18
4. evento-extra	43	5	38	0	38	0
5. evento-trocado	91	15	0	76	0	76
6. destino-trocado	63	0	0	63	5	58
7. saída-trocada	202	0	0	202	0	202
8. saída-faltando	18	0	0	18	0	18
9. estado-faltando	1	0	0	1	0	1
Total	451	24	38	389	43	384
		Escore	0.9110		0.8992	

Pelos resultados, observa-se que a grande maioria dos mutantes que permanecem vivos relaciona-se à exploração do fato de que a MEF utilizada nesse experimento não atendia ao requisito inicial do método W (Chow, 1978) de ser completamente especificada, ou seja, possuir tratamento para todas as possíveis entradas, em todos os estados. Esse fato é muito comum e é o que ocorre geralmente na prática, justificando ainda mais a aplicação do critério Análise de Mutantes. Salienta-se que com esses resultados, os benefícios da aplicação da Análise de Mutantes no contexto de Statecharts se tornam evidentes, uma vez que os componentes básicos de uma especificação Statecharts são Máquinas de Estados Finitos. Ou seja, pelo menos no nível dos componentes básicos, esse critério já é uma forma complementar de teste. Além disso, considerando-se que os Statecharts não possuem métodos de geração de seqüências de teste como as MEFs e também que eles permitem introduzir uma maior complexidade à especificação, os benefícios da aplicação do critério nesse contexto serão ainda maiores.

Como foi mencionado na Seção 2, o ponto chave para a aplicação do critério Análise de Mutantes é a definição dos operadores de mutação. Os operadores definidos para a técnica Statecharts, apresentados na Tabela 2, foram divididos em três classes, de acordo com as características que são abordadas por eles. A seguir, comenta-se cada uma dessas classes e ilustra-se a definição de um operador para cada uma delas. A especificação formal dos demais operadores pode ser encontrada em (Fabbri, 1996):

1) Operadores que abordam aspectos de Máquinas de Estados Finitos

Esses operadores foram definidos com o objetivo de explorar o fato de que cada componente de um Statecharts é, essencialmente, uma Máquina de Estados Finitos. Sendo assim, a definição dos operadores dessa classe está baseada na classificação de erros proposta por Chow (1978), para MEFs e que consiste de erros de transferência, erros de operação e erros de estados extras/ausentes.

Definição do Operador: *arco faltando*:

Este operador exclui um arco (transição) do componente s.

Mutantes M_r, $0 \leq r \leq ((m.k) + nt_{ext})$, são gerados da seguinte maneira:

Para cada par de estados (s_i , s_j) onde $s_i \in \rho(s)$ e $s_j \in \rho(s) \vee s_j \in S$ tal que $T|_{si,sj} \neq \phi$ geram-se mutantes onde:

$(E|_s)_r \subseteq E|_s$; $(A|_s)_r \subseteq A|_s$; $(\rho(s))_r = \rho(s)$; $(T|_s)_r \neq T|_s$;

$(T|_s)_r = T|_s - \{ t \}$ para cada $t \in T|_{si,sj}$

sendo: m o número de eventos e k o número de estados do componente s; nt_{ext} o número de transições que saem do componente s e vão para outro componente; $E|_s$ e $A|_s$ os eventos e as ações restritas ao componente s; $T|_s$ as transições que têm por origem um subestado de s e $T|_{si,sj}$ as transições existentes do estado s_i para o estado s_j.

2) Operadores que abordam aspectos de Máquinas de Estados Finitos Estendidas

Esses operadores foram definidos com o objetivo de explorar o fato de que cada componente de um Statecharts pode ser uma Máquina de Estados Finitos Estendida, caso ele faça uso de variáveis e condições. Nesse caso, a definição dos operadores foi inspirada em dois trabalhos existentes na literatura: o trabalho de Agrawal et al (1989) que define operadores para a linguagem de programação C e no trabalho de Weyuker et al (1994) que define alguns operadores para apoiar a seleção de casos de teste para expressões booleanas.

Definição do Operador: *troca operador relacional por operador relacional*:

Para cada $t \in T_{ext}$, com l = e/a , em cada ocorrência de um operador relacional, substitui-se o operador relacional por outro operador relacional.

sendo: T_{ext} o conjunto de transições do componente s que fazem uso de variáveis e ações e l = e/a o rótulo da transição com e o evento e a a ação.

3) Operadores que abordam aspectos intrínsecos à técnica Statecharts

Esses operadores foram definidos com o objetivo de explorar aspectos próprios da técnica Statecharts relativos às características de história, broadcasting e paralelismo. Nesse caso, a definição dos operadores foi baseada na sintaxe dos Statecharts que descreve essas características.

Definição do Operador: *troca h por h**:

Este operador troca o tipo de história associada a um estado, fazendo com que a história passe a atingir todos os subestados na hierarquia sempre que o estado possuir mais de um nível de decomposição.

Mutantes M_r , $0 \leq r \leq |S|$, são definidos da seguinte maneira:

Para cada $s \in S$ tal que $\gamma^{-1}(s) = h$ e $\exists\ s' \in \rho(s)$ com $\rho(s') \neq \phi$

faz-se $\gamma^{-1}(s) = h^*$

(neste caso, a notação utilizada foi apresentada anteriormente, com a sintaxe)

Para a aplicação da Análise de Mutantes no contexto de Statecharts, além dos operadores de mutação foram definidas três estratégias de abstração que estão baseadas no aspecto de decomposição dessa técnica e que selecionam, por nível de decomposição, os componentes, ou seja, as Máquinas de Estados Finitos pertencentes a um determinado nível. Essas estratégias possibilitam que a condução do teste seja realizada através das abordagens "top-down" ou "bottom-up". Identificados os componentes, o teste da especificação pode partir do nível mais alto de abstração, que corresponde ao próprio Statecharts, ou então, pode partir do nível mais baixo de abstração, o qual é composto pelos componentes cujos subestados são todos básicos (Fabbri, 1996; Fabbri et al, 1997).

Tabela 2 - Operadores de Mutação para a Técnica Statecharts

Operadores que abordam aspectos de Máquinas de Estados Finitos	
1. alteração do estado default	6. destino trocado
2. arco faltando	7. ação faltando
3. evento faltando	8. ação trocada
4. evento extra	9. estado faltando
5. evento trocado	
Operadores que abordam aspectos de Máquinas de Estados Finitos Estendidas	
1. exclusão de expressão	7. negação Lógica
2. negação de expressão booleana	8. troca variável por variável
3. troca associatividade dos termos	9. troca variável por constante
4. troca operador aritmético por operador aritmético	10. troca de constantes por constantes requeridas
5. troca operador relacional por operador relacional	11. troca constante por variável escalar
6. troca operador lógico por operador lógico	
Operadores que abordam aspectos específicos da técnica Statecharts	
1. desassocia história da transição	10. exclui condição not-yet(e)
2. troca transição associada ao símbolo de história	11. troca evento da condição not-yet(e)
3. exclui história do estado	12. exclui evento exit(s)
4. troca h por h*	13. troca estado do evento exit(s)
5. troca h* por h	14. exclui evento entered(s)
6. associa h ao estado	15. troca estado do evento entered(s)
7. associa h* ao estado	16. altera transição origem do broadcasting
8. exclui condição in(s)	17. altera transição destino do broadcasting
9. troca estado da condição in(s)	

A análise dos mutantes que são gerados, um ponto importante do critério, pode ser feita avaliando-se o comportamento dos mesmos em relação a duas abordagens: 1) isoladamente, considerando-se o comportamento do componente original e do componente mutante, comparando-se a configuração alcançada por eles; 2) dentro do contexto do próprio Statecharts, comparando-se a configuração alcançada pelo Statecharts como um todo.

As estratégias, denominadas *Básica*, *Baseada em Ortogonalidade* e *Baseada em História*, diferenciam, essencialmente, no aspecto que se deseja explorar na atividade de teste. No entanto, todas elas aplicam procedimentos similares para abstrair os componentes de cada nível do Statechart. Em seguida, apresenta-se a Estratégia Básica com seu algoritmo e discutem-se as outras duas:

- **Estratégia Básica**

A Estratégia Básica, cujo algoritmo está apresentado na Figura 1, abstrai os componentes do tipo OR que correspondem essencialmente à estrutura básica de composição dos Statecharts que é a Máquina de Estados Finitos. Esses componentes podem ser vistos como Máquinas de Estados Finitos Estendidas. O objetivo principal dessa estratégia é testar os componentes OR, abordando também aspectos de integração entre eles. Ela pode ser considerada como uma estratégia mais geral, onde não se exploram aspectos intrínsecos à técnica Statecharts.

Início Estratégia Básica

** *B_1 é o conjunto de estados básicos* **

$B_1 = \{ s \in S \mid \rho(s) = \phi \}$

** *$CLCA^+_1$ é o conjunto de estados candidatos a MEFs do nível de abstração 1* **

$CLCA^+_1 = \{ s \in S \mid s = LCA^+(b_i), b_i \in B_1 \}$

** *$CMEF_1$ é o conjunto das MEFs do nível de abstração 1* **

$CMEF_1 = \{ s \in CLCA^+_1 \mid \forall s_i \in \rho(s), \rho(s_i) = \phi \}$

$i = 1$

Enquanto $\mid CLCA^+_i \mid > 1$ *faça*

$i = i + 1$

** *B_i é o conjunto dos estados pertencentes às MEFs que ainda não foram abstraídas* **

$B_i = B_{i-1} - \{ \cup \rho(s), \text{para todo } s \in CMEF_{i-1} \}$
$- \{ \cup \rho(s_i) \mid \psi(s_i) = \text{AND} \wedge s_i \in \rho(s), s \in CMEF_{i-1} \}$
$\cup CMEF_{i-1}$

** *$CLCA^+_i$ é o conjunto dos estados candidatos a MEFs do nível de abstração i* **

$CLCA^+_i = \{ s \in S \mid s = LCA^+(b_i), b_i \in B_i \}$

** *$CMEF_i$ é o conjunto das MEFs do nível de abstração i* **

$CMEF_i = \{ s \in CLCA^+_i \mid \forall s_i \in \rho(s),$
$\text{se } \psi(s_i) = \text{OR então } s_i \in B_i$
$\text{c.c. } \forall s_j \in \rho(s_i), s_j \in B_i \}$

fim-enquanto

Fim Estratégia Básica

Figura 1 - Estratégia Básica

Para exemplificar a aplicação dessa estratégia, considere o statecharts extraído de Harel (1987a), apresentado na Figura 2. Nessa Figura, as MEFs que compõem o primeiro nível de abstração apresentam-se hachuradas.

Os conjuntos selecionados de acordo com a aplicação dessa estratégia são:

$B_1 = \{ s \in S \mid \rho(s) = \phi \}$
$B_1 = \{$ dead, light.off, light.on, power.ok, power.blink, alarm1_status.disabled, alarm1_status.enabled, alarm2_status.disabled, alarm2_status.enabled, enabled.quiet, enabled.beep, chime_status.disabled, alarms_beep.alarm1beeps, alarms_beep.alarm2beeps, alarms_beep.bothbeep, update1.hr, update1.10min, update1.1min, alarm1.off, alarm1.on, update2.hr, update2.10min, update2.1min, alarm2.off, alarm2.on, chime.off, chime.on, run.on, run.off, display.reg, display.lap, zero, beep_test.00, beep_test.01, beep_test.10, beep_test.beep, update.sec, update.min, update.10min, update.hr, update.mon, update.date, update.day, update.year, update.mode, time, date, wait $\}$

$CLCA^+_1 = \{ s \in S \mid s = LCA^+(b_i), b_i \in B_1 \}$
$CLCA^+_1 = \{$ citzen_quartz_multi_alarm, light, power, alarm1_status, alarm2_status, enabled, chime_status, alarms_beep, update1, alarm1, update2, alarm2, chime, run, display, stopwatch, beep_test, update, regular, displays $\}$

$CMEF_1 = \{ s \in CLCA^+_1 \mid \forall s_i \in \rho(s), \rho(s_i) = \phi \}$
$CMEF_1 = \{$ light, power, alarm1_status, alarm2_status, enabled, alarms_beep, update1, alarm1, update2, alarm2, chime, run, display, beep_test, update $\}$

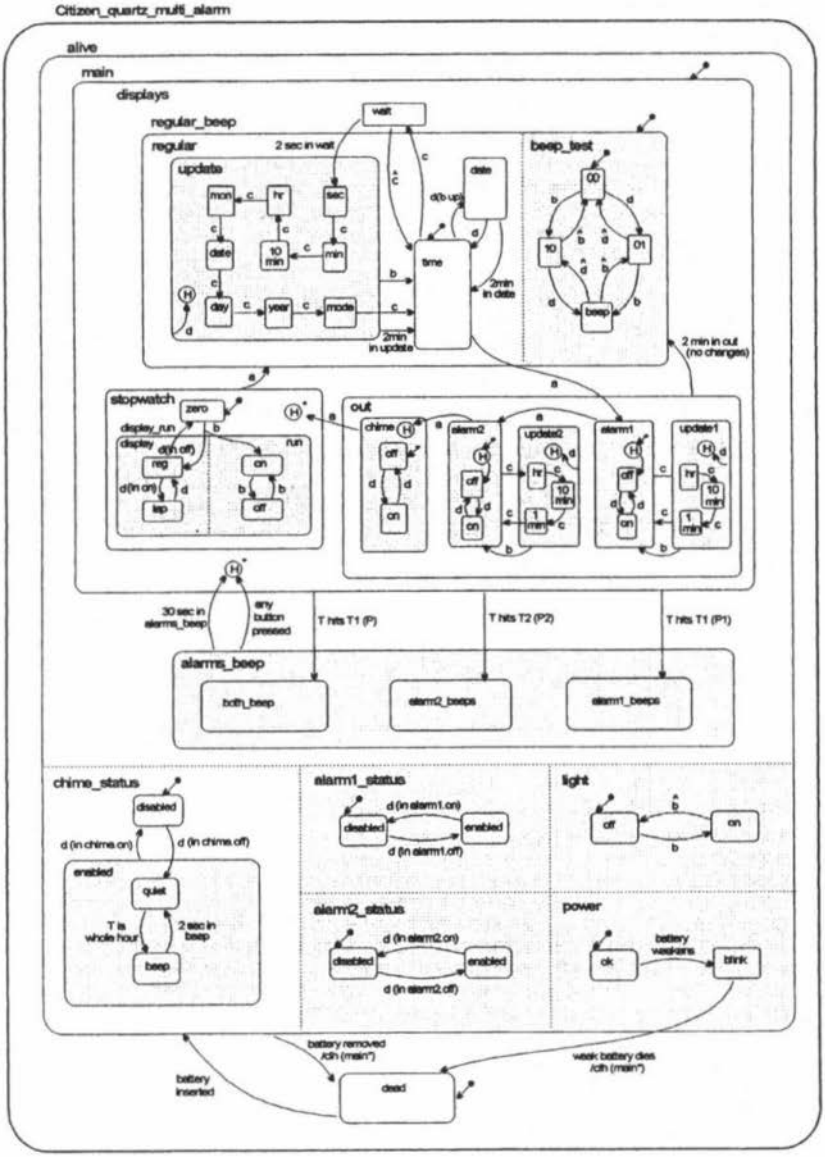

Figura 2 - Especificação de um Relógio Digital através da Técnica Statecharts (Harel, 1987a)

Aplicando-se o procedimento estabelecido na estratégia enquanto a cardinalidade do conjunto $CLCA^{+}_{i}$ é maior que um, isto é, $| CLCA^{+}_{i} | > 1$, obtém-se os seguintes conjuntos de MEFs, para os níveis do statecharts em questão:

$CMEF_2$ = { chime_status, stopwatch, regular, out } $CMEF_4$ = { main }

$CMEF_3$ = { displays } $CMEF_5$ = { citzen_quartz_multi_alarm }

Teorema 1: o algoritmo Estratégia Básica sempre pára.
Prova: a cada iteração, define-se o conjunto dos estados candidatos a MEFs que compõem um determinado nível de abstração e esse conjunto tende a diminuir à medida que o nível de abstração vai se tornando mais geral, ao ponto de corresponder ao próprio Statecharts, caso este seja do tipo OR. Se o Statecharts for do tipo AND, esse conjunto torna-se vazio depois de abstraídos todos os componentes correspondentes às MEFs que compõem o Statecharts. Portanto, a cardinalidade de $CLCA^{+}_{i}$ sempre atinge o valor "um" ou "zero" e o algoritmo pára.

Observa-se então, que existem cinco níveis de abstração, correspondentes aos conjuntos $CMEF_1$ a $CMEF_5$. Em cada um desses níveis identificam-se Máquinas de Estados Finitos Estendidas (MEFEs), com base nas quais os mutantes serão gerados ao serem aplicados os operadores de mutação apresentados anteriormente.

- **Estratégia Baseada em Ortogonalidade**

Essa estratégia explora o aspecto de paralelismo da técnica Statecharts. Nesse caso, são selecionados como alvos de mutação, em cada nível de abstração do Statecharts, apenas os componentes tipo AND que compõem a especificação. A diferença fundamental entre o algoritmo desta estratégia e o da Estratégia Básica consiste em verificar, a cada iteração, se as MEFs abstraídas possuem subestados do tipo "AND". Caso possuam, serão esses os componentes daquele nível.

No caso dessa estratégia, os componentes abstraídos que correspondem aos estados do tipo "AND" que compõem o statecharts são:

$COrt_1 = \phi$ $COrt_4 = \phi$

$COrt_2$ = { display_run } $COrt_5$ = { alive }

$COrt_3$ = { regular_beep }

Teorema 2: o algoritmo da Estratégia Baseada em Ortogonalidade pára.

- **Estratégia Baseada em História**

Essa estratégia explora a característica de história, isto é, a capacidade da técnica Statecharts de memorizar a última configuração atingida por um estado. Uma vez escolhida essa característica, são selecionados como alvos de mutação apenas os componentes (as MEFs Estendidas) que possuem símbolos de história associados a eles. A diferença fundamental entre o algoritmo desta estratégia e o da Estratégia Básica consiste no fato de que a cada iteração é feita a intersecção do conjunto de MEFs daquele nível de abstração com o conjunto $\gamma(H)$ que contém os componentes que possuem símbolo de história associado a eles. Assim, definidos esses componentes, o testador pode dar ênfase ao aspecto de história, para cada nível de abstração do Statecharts.

Nesse caso, os componentes que possuem associado a eles o símbolo de história, em cada um dos cinco níveis do statechart em questão, são:

$CMEF_{H1}$ = { update1, alarm1, update2, alarm2, chime, update } $CMEF_{H4} = \phi$

$CMEF_{H2}$ = { stopwatch } $CMEF_{H5} = \phi$

$CMEF_{H3}$ = { displays }

Teorema 3: o algoritmo da Estratégia Baseada em História pára.

Analogamente à prova do Teorema 1 têm-se as provas dos Teoremas 2 e 3, pois a cada iteração, constróem-se conjuntos de componentes relativos aos níveis mais altos de abstração, até que se chegue ou no componente raiz, caso este seja do tipo OR, ou então não se tenha mais componentes para abstrair, caso o componente raiz seja do tipo AND.

Uma vez selecionados os componentes, aplicam-se a eles os operadores de mutação de acordo com os tipos de erros que se deseja explorar na atividade de teste. Certamente, para as estratégias Baseada em Ortogonalidade e Baseada em História, os operadores mais releventes são aqueles que exploram aspectos de broadcasting e de história, respectivamente. No entanto, qualquer um dos operadores definidos para a técnica Statecharts pode ser aplicado, a critério do testador ou de acordo com experiências prévias.

Salienta-se que assim como os outros critérios de teste existentes, a aplicação da Análise de Mutantes no contexto de Statecharts é inviável sem o apoio de uma ferramenta. Isso ficou comprovado pelos dois experimentos mencionados, que foram conduzidos de forma manual. Portando, dada a necessidade do apoio de uma ferramenta, foi desenvolvida a Proteum-RS que apóia a aplicação do critério no contexto de Sistemas Reativos. Atualmente ela se encontra instanciada, na versão Proteum-RS/FSM, para Máquinas de Estados Finitos (Fabbri et al, 1994a; Fabbri et al, 1995b). Nessa versão, a simulação da MEF em teste e das MEFs mutantes, é feita através do simulador do ambiente StatSim (Masiero et al, 1991), que apóia a simulação de especificações baseadas em Statecharts, uma vez que a técnica Statecharts é uma extensão das Máquinas de Estados Finitos. A versão Proteum-RS/ST para apoiar a aplicação da Análise de Mutantes na validação de especificações baseadas em Statecharts está sendo desenvolvida e corresponde, basicamente, à implementação dos operadores de mutação e das estratégias apresentadas neste artigo, uma vez que o simulador utilizado pela ferramenta, na sua versão FSM, é o próprio simulador de Statecharts.

5. CONCLUSÕES

Apresentaram-se neste trabalho as definições necessárias para a aplicação do critério Análise de Mutantes na validação de especificações baseadas em Statecharts. A técnica Statecharts é uma das técnicas formais para especificação do aspecto comportamental de Sistemas Reativos. Esses sistemas, normalmente, apóiam atividades humanas fundamentais podendo envolver grandes riscos à vida ou ao patrimônio. Assim, a atividade de teste e validação dos mesmos deve ser conduzida de forma ainda mais criteriosa para que esses riscos sejam minimizados. Em geral, a validação de especificações baseadas em Statecharts é feita através de alguns tipos de simulação: batch, interativa, programada e exaustiva (Harel, 1992). No entanto, através de simulações nem todos os erros são detectados e, além disso, não se tem um critério para avaliar a atividade de teste de forma quantitativa.

Com a aplicação do critério Análise de Mutantes, como foi mostrado neste artigo, a atividade de teste de tais especificações pode ser avaliada quantitativamente, através do escore

de mutação. Além disso, a aplicação da Análise de Mutantes permite que o teste seja conduzido de forma incremental, através da seleção dos operadores de mutação, definindo-se um conjunto inicial e aumentando-se esse conjunto de acordo com os recursos disponíveis. Particularmente, no caso de Statecharts, essa forma incremental pode ser considerada também do ponto de vista hierárquico, isto é, selecionando-se para teste determinados componentes da especificação, utilizando-se as estratégias propostas para abstrair os componentes desejados. Outra forma de aplicação do critério é através do uso da Mutação Alternativa – Mutação Aleatória e Mutação Restrita – proposta em Fabbri (1996), que visa à diminuição do custo, um fator restritivo de aplicação da Análise de Mutantes, associado à execução dos mutantes gerados.

Foram apresentadas três estratégias para selecionar os componentes de uma especificação. A Estratégia Básica, que abstrai todos os componentes do tipo OR dos Statecharts, por nível de hierarquia, os quais correspondem às MEFs Estendidas que compõem a especificação. A Estratégia Baseada em História, que abstrai os componentes que possuem essa característica, permitindo que esse aspecto seja explorado separadamente e a Estratégia Baseada em Ortogonalidade, que abstrai os estados do tipo AND, possibilitando que a comunicação através de broadcasting seja tratada com maior detalhe.

Além dessas estratégias, foram apresentados operadores de mutação que enfocam também três contextos diferentes, isto é, características particulares de Máquinas de Estados Finitos, a utilização de variáveis e condições, que caracterizam as Máquinas de Estados Finitos Estendidas e operadores que exploram aspectos específicos da técnica Statecharts, relacionados com história, broadcasting e paralelismo. Além dos operadores propostos, outros operadores de mutação podem ser definidos com a finalidade de explorar outras condições de erro, permitindo uma melhor adequação do teste aos erros mais freqüentemente cometidos pelo especificador.

O experimento conduzido para Máquinas de Estados Finitos evidencia que a Análise de Mutantes para a validação de Statecharts, proposta neste artigo, é complementar às formas de validação existentes para esta técnica, uma vez que os componentes de um statechart são MEFs e, para esta técnica que possui vários métodos de geração de seqüências de teste, ao contrário dos Statecharts que não os possuem, o critério Análise de Mutantes mostrou-se uma forma complementar de teste.

REFERÊNCIAS BIBLIOGRÁFICAS

Agrawal, H.; DeMillo, R.A.; Hathaway, R.; Hsu, Wm.; Hsu, W.; Krauser, E.; Martin, R.J.; Mathur, A.P.; Spafford, E. "Design of Mutant Operators for the C Programming Language", *Technical Report SERC-TR-41-P*, Software Eng. Research Center, Purdue University, Março, 1989.

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.", in *7th ACM Symposium on Principles of Programming Languages*, Janeiro, 1980.

Choi, B.J.; DeMillo, R.A.; Krawser, E.W.; Martin, R.J.; Mathur, A.P.; Offut, A.J.; Pan, H.; Spafford, E.H. "The Mothra Tool Set", in *Proceedings of the 22nd Hawaii International Conference on Systems and Software*, Kona, Hawaii, Janeiro, 1989b.

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

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.

DeMillo, R.A. "Mutation Analisys as a Tool for Software Quality Assurance", in *Proc. of COMPSAC 80*, Chicago-IL, Outubro, 1980.

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, Maio, 1993.

Fabbri, S.C.P.F.; Delamaro, M.E.; Maldonado, J.C.; Masiero,P.C. "Proteum/FSM: Especificação de 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*, pp.284-302, Curitiba, Maio, 1994a.

Fabbri, S.C.P.F.; Maldonado, J.C.; Delamaro, M.E.; Masiero, P.C. "Mutation Analysis Testing for Finite State Machines", in *Proc. ISSRE'94 - Fifth International Symposium on Software Reliability Engineering*, pp.220-229, California, Novembro, 1994b.

Fabbri, S.C.P.F.; Maldonado, J.C.; Masiero, P.C.; Delamaro, M.E. "Aplicação do Critério Análise de Mutantes na Validação de Especificações Baseadas em Redes de Petri", in *Anais do VIII Simpósio Brasileiro de Engenharia de Software*, Curitiba, Outubro, 1994c.

Fabbri, S.C.P.F.; Maldonado, J.C.; Masiero, P.C.; Delamaro, M.E.; Wong, E. "Mutation Testing Applied to Validate Specifications Based on Petri Nets", in *Proc. FORTE'95 - 8th International IFIP Conference on Formal Description Techniques for Distributed Systems and Communications Protocols*, Montreal, Canada, Outubro, 1995a.

Fabbri, S.C.P.F.; Maldonado, J.C.; Delamaro, M.E.; Masiero, P.C. "Proteum/FSM - Uma Ferramenta para Apoiar a Validação de Máquinas de Estados Finitos pelo Critério Análise de Mutantes", in *Anais do IX Simpósio Brasileiro de Engenharia de Software*, pp.475-478, Recife, Pernambuco, Outubro, 1995b.

Fabbri, S.C.P.F. "A Análise de Mutantes no Contexto de Sistemas Reativos: uma Contribuição para o Estabelecimento de Estratégias de Teste e Validação", *Tese de Doutorado*, IFSC/USP, São Carlos, SP, 1996.

Fabbri, S.C.P.F.; Maldonado, J.C.; Masiero, P.C.- "Mutation Analysis in the Context of Reactive System Specification and Validation", in *Anais do SQM'97 - 5th Annual International Conference on Software Quality Management*, Bath, UK, Março, 1997.

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.

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

Harel, D. "Statecharts: A Visual Formalism for Complex Systems", *Science of Computer Programming*, Vol. 8, pp 231-274, 1987a.

Harel, D. "Statecharts: On the Formal Semantics of Statecharts", in *Proc. 2nd IEEE Symposium on Logic in Computer Science*, Ithaca, New York, 1987b.

Harel, D. "Biting the Silver Bullet - Toward a Brighter Future for Systems Development", *Computer IEEE*, pp.8-20, Janeiro, 1992.

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

Maldonado, J.C. "Critérios Potenciais Usos: Uma Contribuição ao Teste Estrutural de Software", *Tese de Doutorado*, FEE/Unicamp, Campinas - SP, 1991.

Masiero,P.C.; Fortes,R.P.M.; Batista Neto,J.E.S. "Edição e Simulação do Aspecto Comportamental de Sistemas de Tempo Real", in *Anais do XI Congresso Nacional da SBC, XVIII SEMISH*, Santos, pp. 45-61, Agosto, 1991.

Masiero, P.C.; Maldonado, J.C.; Boaventura, I.G. "A Reachability Tree for Statecharts and Analysis of Some Properties", *Information and Software Technology*, Vol.36 (10), pp.615-624, 1994.

Mathur, A.P.; Krauser, E.W. "Modeling Mutation on Vector Processor", in *Proceedings of the 2nd Workshop on Software Testing, Verification and Analysis*, Banff, Canada, 1988.

Offutt, A.J. "Investigations of the Software Testing Coupling Effect", *ACM Transactions on Software Engineering Methodology*, 1(1), pp.3-18, Janeiro, 1992.

Peterson, J.L. *Petri Nets*, Computing Surveys, Vol.9, N. 3, Setembro, 1977.

Petrenko, A.; Bochmann, G.v. "On Fault Coverage of Tests for Finite State Specifications", http://www.iro.umontreal.ca/pub/teleinfo/TRs/Petr96b.ps.gz, 1996.

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

Probert, R.L.; Guo, F. "Mutation Testing of Protocols: Principles and Preliminary Experimental Results", in *Proceedings of the IFIP TC6 Third International Workshop on Protocol Test Systems*, North-Holland, pp. 57-76, 1991.

Wang, C.J.; Liu, M.T. "Generating Test Cases for EFSM with Given Fault Model", *IEEE INFOCOM'93 - 12th Annual Joint Conference of the IEEE Computer and Commun. Societies*, Vol.2, pp. 774-781, 1993.

Weyuker, E.; Goradia, T.; Singh, A. "Automatically Generating Test Data from a Boolean Specification", *IEEE Trans. on Software Engineering*, Vol. 20, N. 5, pp.353-363, Maio, 1994.

Wing, J.M. "A Specifier's Introduction to Formal Methods", *IEEE Computer*, pp.8-22, Setembro, 1990.