Um Estudo sobre a Avaliação do Custo de Aplicação da Análise de Mutantes na Validação de Máquinas de Estados Finitos

Renata A. de Carvalho [1 *]
Sandra Camargo P. F. Fabbri [1]
José Carlos Maldonado [2]

1 Departamento de Computação – DC
Universidade Federal de São Carlos – UFSCar
Caixa Postal 676 CEP 13.565-905
São Carlos – SP – Brasil
{renataap, sandraf}@dc.ufscar.br

2 Departamento de Computação e Estatística
Instituto de Ciências Matemáticas e de Computação – ICMC
Universidade de São Paulo – USP
Caixa Postal 668 CEP 13.560-970
São Carlos – SP – Brasil
jcmaldon@icmc.sc.usp.br

Abstract

Finite State Machine (FSM) is one of the most used techniques for the specification of the Reactive Systems behavioral aspect, for instance, communication protocols. The validation of these specifications, in the case of protocols, conformance testing, is a relevant topic and the aim of many researches. A tendency to characterize an error model to conduct the validation activity is identified. Recently, the adequacy of the use of Mutation Analysis Criterion (MA), traditionaly used in program testing, has been studied in the context of FSM validation. This paper presents the results of an experiment that was conducted aiming at contributing to evaluate the application cost of Mutation Analysis and, in this direction, an essential mutant operators set for FSM is invetigated, aiming at the cost reduction without compromissing the quality of testing and validation activities. The results presented contribute for establishement of a knowledge body on the perspective of defining an incremental, low-cost testing and validation strategy in the context of FSM.

Keywords: Finite State Machines, Error Model, Mutation Analysis, Incremental Testing, Essential Mutant Operators Set

1. Introdução

O uso amplo e crescente de software nas mais diversas áreas de aplicação impõe a necessidade de desenvolvimento de software de alta qualidade através do uso de métodos, técnicas e ferramentas apropriadas. O Teste de Software é fundamental no desenvolvimento de software de qualidade, seguro e confiável; no entanto, o custo da atividade de teste é, em geral, bastante alto em relação ao custo total de desenvolvimento. A atividade de teste é essencial para revelar a presença de erros e nos Sistemas Reativos, como por exemplo, protocolos de comunicação, controle metroviário, etc., cuja principal característica é interagir com o meio ambiente reagindo a estímulos, essa atividade deve ser ainda mais criteriosa, pois falhas nesses sistemas podem provocar riscos à vida e/ou econômicos.

Dado o aspecto crítico do desenvolvimento e aplicação dos Sistemas Reativos, métodos e técnicas formais têm sido estabelecidos para especificação de seu aspecto comportamental. Dentre eles estão as Máquinas de Estados Finitos (MEF) [1] que, para sua

[*] Profª da Faculdade de Ciências Exatas e Tecnológicas (FCET) – Universidade de Marília – UNIMAR

validação, vários critérios e métodos de geração de seqüências de teste [2, 3, 4, 5, 6] têm sido propostos e investigados, todos eles objetivando mostrar a ausência de certos tipos de erros categorizados em modelos de erros. Esforços têm sido identificados na definição e caracterização de modelos de erros de especificações baseadas em MEF [7, 8, 9, 10]. Mais recentemente, propôs-se o uso do critério Análise de Mutantes, tradicionalmente utilizado no teste de programas, nesse contexto [8, 11, 12].

Na Análise de Mutantes, os erros são modelados através dos operadores de mutação. Fabbri definiu um conjunto de operadores de mutação para Máquinas de Estados Finitos implementados na ferramenta Proteum-RS/FSM [13]. A aplicação dos operadores de mutação na especificação **S** leva à obtenção de um conjunto de especificações mutantes denotado por $\phi(S)$. Um problema inerente à aplicação da Análise de Mutantes é o seu custo, essencialmente devido ao custo de execução e análise dos mutantes gerados. Por outro lado, o critério Análise de Mutantes, no teste de programas, tem apresentado uma alta eficácia em revelar erros e evidências nessa direção foram obtidas por Fabbri [13], no contexto de especificações.

Várias alternativas para a aplicação da Análise de Mutantes têm sido investigadas. Dentre elas, a Mutação Restrita ("Constrained Mutation") [14], que procura caracterizar um subconjunto de operadores de mutação para aplicação na atividade de teste, reduzindo o custo mas procurando manter a eficácia do critério. Nessa perspectiva inserem-se os trabalhos de Offut *et al.* [15] e Barbosa *et al.* [16] que procuraram determinar um conjunto essencial de operadores de mutação para as linguagens Fortran e C, respectivamente. Espera-se que ao determinar um conjunto **T** de seqüências de teste adequado ao conjunto de operadores essenciais, **T** seja também adequado em relação ao conjunto de operadores definidos para a linguagem alvo. Barbosa *et al.* definiu o Procedimento Essencial para a determinação do conjunto de operadores essenciais para a linguagem C.

O objetivo deste trabalho é contribuir para avaliar o custo de aplicação do critério Análise de Mutantes no contexto de Máquinas de Estados Finitos. Nessa perspectiva, discute-se uma estratégia incremental de aplicação dos operadores e avalia-se a adequação de aplicação do Procedimento Essencial no contexto de MEF. Assim, procura-se contribuir para a definição de uma estratégia de teste que minimize o custo de aplicação desses operadores, visando determinar uma melhor relação custo/benefício de aplicação da Análise de Mutantes no contexto de Máquinas de Estados Finitos.

Na Seção 2 comentam-se os critérios e métodos de teste e validação de Máquinas de Estados Finitos. Na Seção 3 é apresentado o critério Análise de Mutantes no contexto de Máquinas de Estados Finitos. Na Seção 4 apresenta-se o experimento conduzido e discutem-se algumas estratégias de aplicação dos operadores de mutação, com maior ênfase na determinação de um conjunto essencial de operadores, definido com base no procedimento apresentado no Apêndice A. Na Seção 5 são apresentadas as conclusões deste trabalho.

2. Aspectos de Validação de Máquinas de Estados Finitos

Máquina de Estados Finitos é uma técnica de especificação formal, com apoio gráfico, muito utilizada na especificação do aspecto comportamental de Sistemas Reativos, particularmente, na área de protocolos de comunicação [7, 17, 18].

As atividades de teste e validação de MEF são fundamentalmente baseadas em um modelo de erros, que corresponde a um conjunto restrito e preestabelecido dos erros a serem explorados durante as atividades de teste. O teste baseado em MEF é, normalmente, tratado como o problema de testar a implementação, ou seja, verificar se a implementação está em conformidade com a especificação, o que significa verificar se apresenta ou não o

comportamento esperado. Petrenko e Bochmann [7] comentam que a abordagem de teste baseada em um modelo de erros é muito prática no sentido de captar a intenção do testador para detectar erros de implementação e, além disso, como as MEFs são muito utilizadas na fase de especificação, discutem aspectos de cobertura no teste de especificações baseadas em Máquinas de Estados Finitos.

Morell [19] define o teste baseado em erros como sendo aquele cujo objetivo é demonstrar a ausência de erros preestabelecidos. Assim, tendo essa estratégia a finalidade de encontrar tipos específicos de erros, ela é, normalmente, uma estratégia bem sucedida pois, em geral, os programadores têm a tendência de cometer determinados erros. Com a identificação desses erros e a definição de um conjunto preestabelecido dos mesmos, a atividade de teste pode ser adequada ao ambiente de desenvolvimento do software, com uma melhor relação custo/benefício.

Identificam-se na literatura diversos métodos de geração de seqüências de teste para a validação de especificações baseadas em Máquinas de Estados Finitos, como os métodos W [2], Wp – "Partial W-Method" [3], DS – "Distinguishing Sequences" [4], TT – "Transition-Tour" [5] e UIO – "Unique-Input-Output" [6]. Segundo Fujiwara [3], um método de geração de seqüências de teste tem como objetivo oferecer a possibilidade de se conduzir as atividades de teste e validação de forma sistemática, através de procedimentos bem definidos para a geração das seqüências de teste, favorecendo a obtenção de melhores características de qualidade. No caso de MEFs, os métodos de teste exigem que elas satisfaçam alguns requisitos, como por exemplo, ser minimal, determinística, completamente especificada, fortemente conectada entre outras, o que na prática, em geral, não acontece. Este é um problema para a aplicação desses métodos, além do comprimento das seqüências de teste geradas.

Na definição e avaliação desses métodos existe a preocupação fundamental em caracterizar a capacidade do método em gerar seqüências de forma que certos tipos de erros sejam detectados. Na essência, esses métodos procuram fornecer mecanismos para detectar certos tipos de erros. Por exemplo, Chow [2] ao propor o método W, faz a análise da capacidade dos métodos em revelar três classes de erros: erros de transferência, erros de operação e erros de estados extras e/ou ausentes. Observa-se assim, que de forma subjacente, existe uma preocupação com modelos de erros ao se definir e avaliar métodos de geração de seqüências de teste.

Na década de 90 observam-se diversas iniciativas para a caracterização de um modelo de erros concretizado na aplicação de mutações na especificação original [7, 8, 9, 10, 11].

A técnica proposta por Chung e Sidhu [9] consiste na geração de seqüências de teste probabilísticas para o teste de conformidade de protocolos e na avaliação da cobertura dessas seqüências. Para a verificação da cobertura de seqüências de teste probabilísticas, geram-se, randomicamente, através do procedimento da simulação Monte-Carlo [20] com algumas modificações, máquinas erradas que possuem pequenas diferenças em relação à máquina especificada e, em seguida, aplica-se a seqüência à cada uma das máquinas geradas para a análise, verificando se estão realmente em conformidade com a máquina especificada.

Outro trabalho na área de teste de conformidade é o método proposto por Wang e Liu [10], no qual os modelos de falhas são deixados para os usuários, que podem definir os erros que eles considerem mais importantes. Os casos de teste são gerados comparando as diferenças entre a especificação em teste e a especificação mutante. É definida uma entrada inicial, que é aplicada tanto na especificação como no mutante e essa entrada vai sendo expandida até que seja alcançada uma situação na qual a especificação e o mutante geram um comportamento diferente.

As propostas desses autores assemelham-se ao critério Análise de Mutantes, tradicionalmente aplicado no teste de programas, na medida que caracterizam um conjunto de máquinas mutantes e requerem a geração de seqüências de teste capazes de distinguir o comportamento das máquinas mutantes da máquina original; diferenciam-se da Análise de Mutantes nos mecanismos utilizados para a definição/geração das mutações e nas hipóteses básicas: especificador competente e efeito de acoplamento. Probert e Guo [8] e Fabbri *et al*. [11] exploram mais especificamente a Análise de Mutantes.

Probert e Guo [8] propõem a técnica de teste E-MPT (Estelle-directecd Mutation-based Protocol Testing) para o teste de especificações de protocolos escritas em Estelle. Na abordagem apresentada pelos autores, são definidos cinco operadores, em termos da especificação Estelle, e esses operadores de mutação atuam na especificação Estelle gerando um mutante, com base no qual é gerado um código C, que por sua vez é completado pelo usuário para tornar-se equivalente à especificação mutante, em Estelle. Após esse processo, é que os casos de teste são aplicados, em nível do programa C. Ressalta-se que na execução desses passos, principalmente com a interferência do usuário para a complementação do código em C, novos erros podem estar sendo introduzidos.

Na Análise de Mutantes, um ponto fundamental é a definição de um conjunto de operadores de mutação. Para Máquinas de Estados Finitos os operadores de mutação foram definidos formalmente por Fabbri [13], caracterizando precisamente o tipo de erro que eles modelam. Esses operadores, na realidade, caracterizam um modelo de erros e consistem de transformações sintáticas na máquina original dando origem às máquinas mutantes.

Fabbri, considerando as limitações de ordem prática dos métodos de teste e validação de MEF e a experiência do grupo de Engenharia de Software do ICMC-USP, no estudo e aplicação do critério Análise de Mutantes no teste de programas, explorou a aplicação sistemática desse critério no contexto de Sistemas Reativos, mais especificamente no contexto de Máquinas de Estados Finitos, Redes de Petri e Statecharts. Dado o escopo do presente trabalho, essa linha de pesquisa é melhor caracterizada na seção seguinte.

3. Critério Análise de Mutantes no Contexto de Máquinas de Estados Finitos

A Análise de Mutantes é um dos critérios da técnica de teste baseada em erros e foi proposto originalmente para o teste de programas [21]. No nível de especificação, esse critério pode ser resumido da seguinte maneira: dada uma especificação **S**, em teste, tem-se por objetivo avaliar a qualidade de um conjunto de seqüências de teste **T** em relação a **S**. Para isso, utiliza-se um conjunto de especificações ligeiramente diferentes de **S**, denominadas mutantes de **S**, e busca-se obter um conjunto de casos de teste que consiga revelar as diferenças de comportamento entre **S** e seus mutantes. No caso do comportamento do mutante ser distinguido, diz-se que ele é um mutante *morto*; caso contrário, diz-se que ele permanece *vivo* e então, ou ele é equivalente a **S**, ou é necessário introduzir uma nova seqüência de teste em **T** capaz de matá-lo. O ideal é encontrar um conjunto **T** AM-adequado, ou seja, que consiga matar todos os mutantes gerados. Os mutantes são gerados através de alterações da especificação original, que são feitas com base em um conjunto de operadores denominados **operadores de mutação**, sendo que cada operador pode estar associado a um tipo ou uma classe de erros que se pretende revelar.

Com a aplicação de critérios de teste, torna-se possível suprir a necessidade de quantificar-se a atividade de teste, com o uso, por exemplo, da análise de cobertura que indica quanto dos requisitos de um critério foi satisfeito no teste em questão [13]. Particularmente, com o uso do critério Análise de Mutantes, é possível atribuir um valor quantitativo à

atividade de teste, que fornece um indicativo da qualidade do teste que está sendo conduzido, mostrando se a especificação testada contém ou não os erros retratados nos mutantes. Essa avaliação quantitativa é dada pelo escore de mutação que corresponde à taxa entre o número de mutantes mortos pelo número de mutantes não equivalentes gerados.

A Análise de Mutantes tem sido aplicada em vários contextos e, em nível de especificação, a utilização desse critério concentra-se principalmente no teste de conformidade de protocolos de comunicação, mais particularmente, através de especificações baseadas em Máquinas de Estados Finitos. Isso pelo fato do teste e validação dos protocolos serem fundamentalmente baseados em um modelo de erros, que pode ser visto como um conjunto de operadores de mutação. No caso das MEFs, os operadores foram baseados na hipótese do especificador competente, que considera que especificações feitas por profissionais experientes estão, em geral, próximas do correto e no efeito de acoplamento, que considera que os casos de teste que detectam erros simples são também capazes de detectar erros complexos, o que possibilita a utilização de k-mutantes, sendo k=1, que corresponde ao mutante possuir apenas uma alteração em relação à especificação original [13].

Os operadores de mutação para Máquinas de Estados Finitos foram inspirados na classificação de erros proposta por Chow [2] que classifica os erros de sequenciamento de uma MEF em 3 tipos: erros de transferência, erros de operação e erros de estados extras ou ausentes. Com base nessa classificação, os operadores de mutação para MEFs foram definidos de acordo com a estrutura da Máquina de Estado Finito que sofre a alteração quando o operador é aplicado, podendo ser uma transição, uma saída ou um estado. Assim, na Tabela 1 apresentam-se os operadores de mutação relacionando-os com a classificação de Chow.

Tabela 1 - Operadores de Mutação para Máquinas de Estados Finitos [13]

Classificação Chow	Sigla dos Operadores	Função dos Operadores
Erros de Transferência	TraIniStaAlt (***Tra***nsition ***Ini***tial ***Sta***te ***Alt***eration)	Alteração do Estado Inicial
	TraArcDel (***Tra***nsition ***Arc Del***etion)	Arco Faltando
	TraEveDel (***Tra***nsition ***Eve***nt ***Del***etion)	Evento Faltando
	TraEveIns (***Tra***nsition ***Eve***nt ***Ins***ertion)	Evento Extra
	TraEveAlt (***Tra***nsition ***Eve***nt ***Alt***eration)	Evento Trocado
	TraDesStaAlt (***Tra***nsition ***Des***tination ***Sta***te ***Alt***eration)	Destino Trocado
Erros de Operação	OutDel (***Out***put ***Del***etion)	Saída Faltando
	OutAlt (***Out***put ***Alt***eration)	Saída Trocada
Erros de Estados Extras/Ausentes	StaDel (***Sta***te ***Del***etion)	Estado Faltando
	StaInsOutSep (***Sta***te ***Ins***ertion ***Out***put ***Sep***aration)	Isola Saída Relevante

Com base nesse conjunto de operadores, um dos experimentos realizados por Fabbri [13] aplicou a Análise de Mutantes em uma Máquina de Estados Finitos que especifica o Protocolo de Transporte Classe 0 da ISO [22]. Além de evidenciar o aspecto complementar da Análise de Mutantes, esse experimento também confirmou a dificuldade de se aplicar um critério de teste sem apoio automatizado. Por esse motivo, foi desenvolvida a ferramenta Proteum-RS/FSM (**Pro**gram **Te**sting **U**sing **M**utants – **R**eactive **S**ystems/**F**inite **S**tate **M**achine) [23] que apóia a aplicação do critério Análise de Mutantes na validação de especificações baseadas em Máquinas de Estados Finitos.

A Análise de Mutantes tem se mostrado bastante eficaz para eliminação de erros em programas, o que pode ser visto em trabalhos empíricos e teóricos na área. Entretanto, alguns problemas com a sua aplicação existem e são procuradas formas alternativas para minimizá-los. Um dos problemas é o de decidir pela equivalência de um mutante, o que é um trabalho

dispendioso e depende da interferência humana. Esse problema permanece no contexto de Máquinas de Estados Finitos Estendidas. Outro problema é o grande número de mutantes para serem compilados e executados com um ou mais casos de teste, gerando um consumo elevado de tempo e de recursos computacionais.

Na seção seguinte apresenta-se o experimento conduzido para contribuir na determinação de estratégias alternativas que visem minimizar o custo de aplicação da Análise de Mutantes na validação de MEFs.

4. Descrição do experimento [1]

Nesta seção discutem-se algumas alternativas para minimizar o custo de aplicação da Análise de Mutantes na validação de MEFs. Com base no procedimento descrito no Apêncide A, determina-se um conjunto essencial de operadores para Máquinas de Estados Finitos, a partir dos operadores de mutação descritos na seção anterior, os quais estão implementados na ferramenta Proteum-RS/FSM. O experimento consiste, basicamente, de três etapas descritas nas seções seguintes.

4.1 Seleção das Máquinas de Estados Finitos

Foram selecionadas dez especificações, de diferentes contextos, extraídas da literatura, sendo duas delas, especificações de protocolos de comunicação [2, 7, 22, 24, 25]. As características principais das MEFs selecionadas podem ser observadas na Tabela 2.

Tabela 2 – Características das Máquinas de Estados Finitos Selecionadas

	MEF	Estados	Entradas	Saídas	Transições	Funcionalidade	Referência
Completamente Especificadas	01	04	03	05	12	reconhecedora de comentários de programas	[2]
	02	02	03	02	06	máquina de café	[7]
	03	03	02	02	06	exemplo ilustrativo	[7]
	04	03	03	02	09	exemplo ilustrativo	[7]
	05	04	02	02	08	Local Map	[25]
	06	02	03	03	06	Local Map	[25]
	07	02	02	02	04	Local Map	[25]
Não Compl. Especificadas	08	08	05	04	10	protocolo do Bit Alternante	[24]
	09	07	04	02	08	exemplo ilustrativo	[24]
	10	04	10	10	21	protocolo de transporte	[22]

4.2 Geração dos Conjuntos de Seqüências de Teste AM-Adequados

Inicialmente foi elaborado um conjunto de seqüências de teste adequado em relação à Análise de Mutantes para cada máquina, com base nas especificações das MEFs. Desse conjunto foram extraídos subconjuntos AM-adequados para cada operador, ou seja,

[1] As siglas utilizadas nesta seção são apresentadas na Tabela A1, do Apêndice A.

subconjuntos de seqüências de teste capazes de distinguir todos os mutantes gerados por um determinado operador pertencente ao conjunto total de operadores. Na Tabela 3 é apresentado o número total de mutantes gerados, o número total de mutantes equivalentes e o tamanho dos conjuntos de seqüências de teste adequados à Análise de Mutantes. Na Tabela 4 a informação da Tabela 3 referente aos mutantes gerados está apresentada por operador, separando-se o número de mutantes gerados equivalentes.

Tabela 3 – Mutantes Gerados, Mutantes Equivalentes e Tamanho dos Conjuntos de Seqüências de Teste de cada MEF selecionada

	MEF	Total de Mutantes	Mutantes Equivalentes	Tamanho do Conj. de Seqüências de Teste
Completamente Especificadas	01	135	0	20
	02	27	6	5
	03	45	0	8
	04	69	1	10
	05	71	0	6
	06	38	3	6
	07	21	0	3
Não Compl. Especific.	08	265	0	14
	09	182	0	26
	10	463	30	44

Tabela 4 – Total de Mutantes e Mutantes Equivalentes por Operador

	Completamente Especificadas							Não Compl. Espec.			
MEF Operadores	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**Média Total**
TraIniStaAlt	3/0	1/0	2/0	2/0	3/0	1/0	1/0	7/0	6/0	3/0	2,9/0
TraArcDel	9/0	4/2	6/0	6/0	8/0	4/0	4/0	10/0	8/0	9/1	6,8/0,3
TraEveDel	12/0	6/4	6/0	9/0	8/0	6/0	4/0	10/0	8/0	21/3	9,0/0,7
TraEveIns	0/0	0/0	0/0	0/0	0/0	0/0	0/0	36/0	22/0	43/8	10,1/0,8
TraEveAlt	0/0	0/0	0/0	0/0	0/0	0/0	0/0	36/0	22/0	91/15	14,9/1,5
TraDesStaAlt	36/0	6/0	12/0	18/0	24/0	6/0	4/0	70/0	48/0	63/0	28,7/0
OutDel	13/0	2/0	6/0	9/0	8/0	6/0	4/0	10/0	8/0	18/0	8,4/0
OutAlt	50/0	8/0	6/0	18/0	8/0	12/0	4/0	30/0	16/0	200/0	35,2/0
StaDel	12/0	0/0	6/0	6/0	12/0	0/0	0/0	56/0	42/0	12/0	14,6/0
StaInsOutSep	0/0	0/0	1/0	1/1	0/0	3/3	0/0	0/0	2/0	3/3	1,0/0,7
Total	135/0	27/6	45/0	69/1	71/0	38/3	21/0	265/0	182/0	463/30	131,6/4,0

Após selecionar os conjuntos de seqüências de teste adequados aos operadores de mutação, aplicou-se cada conjunto aos demais operadores para avaliar o escore de mutação que cada operador determina em relação aos demais, ou seja, a capacidade de um conjunto de seqüências de teste *op*-adequado em distinguir os mutantes de cada operador de OP. Esse processo foi repetido para todas as Máquinas de Estados Finitos. Na Tabela 5 apresenta-se as médias dos escores de mutação para o conjunto total das MEFs. Ressalta-se que para o conjunto de MEFs completamente especificadas não são gerados mutantes pelos operadores TraEveIns e TraEveAlt, justamente pelo fato dessas MEFs serem completamente especificadas.

Tabela 5 - Média dos Escores de Mutação para Todas as MEFs Selecionadas

Operador	TraIni StaAlt	TraArc Del	TraEve Del	TraEve Ins	TraEve Alt	TraDes StaAlt	OutDel	OutAlt	StaDel	StaIns OutSep	Média Geral
TraIniStaAlt	1,000000	0,452167	0,388611	0,000000	0,254120	0,195583	0,386611	0,403000	0,532789	0,500000	0,411288
TraArcDel	1,000000	0,900000	0,802778	0,000000	0,859649	0,482746	0,810333	0,826772	0,778912	1,000000	0,746119
TraEveDel	1,000000	0,900000	0,900000	0,000000	1,000000	0,517905	0,900000	0,915000	0,778912	1,000000	0,791182
TraEveIns	1,000000	0,725000	0,632407	1,000000	0,962963	0,532262	0,632407	0,618333	0,966270	1,000000	0,806964
TraEveAlt	1,000000	1,000000	1,000000	0,015152	1,000000	0,638757	1,000000	0,966667	0,575397	1,000000	0,819597
TraDesStaAlt	1,000000	1,000000	0,994444	0,090909	0,986842	1,000000	1,000000	1,000000	0,850340	1.000000	0,892254
OutDel	1,000000	0,900000	0,894444	0,000000	0,969263	0,519143	0,900000	0,910000	0,778912	1,000000	0,787176
OutAlt	1,000000	1,000000	1,000000	0,000000	0,969298	0,640571	1,000000	1,000000	0,778912	1,000000	0,838878
StaDel	1,000000	0,900952	0,795635	0,397307	0,802632	0,558418	0,806429	0,798651	1,000000	1,000000	0,806002
StaInsOutSep	1,000000	0,645834	0,645834	0,045455	0,000000	0,385417	0,333334	0,645834	0,702381	1,000000	0,540409
Média Geral	1,000000	0,842395	0,805415	0,154882	0,780477	0,547080	0,776911	0,808426	0,774282	0,950000	

Algumas informações podem ser extraídas da Tabela 5, como por exemplo, considerando o operador TraEveIns (linha) tem-se que o escore de mutação do conjunto TraEveIns-adequado em relação a TraEveDel (coluna) é de 0.632407, indicando que o conjunto TraEveIns-adequado é capaz de distinguir, em média, 63% dos mutantes de TraEveDel.

Outra informação diz respeito ao *strength*. O *strength* indica a dificuldade de conjuntos de casos de teste adequados a um operador op_i de OP em distinguir os mutantes de *op* e ele é calculado através da operação: *1- escore de mutação em relação a op*. Por exemplo, considerando o operador TraEveAlt (coluna) tem-se que o *strength* deste operador em relação a TraEveDel (linha) é 0 (1.000000 – 1.000000), indicando que o conjunto TraEveDel-adequado é também TraEveAlt-adequado; já em relação ao operador TraIniStaAlt, o *strength* de TraEveAlt é de 0.74588 (1.000000 – 0.254120), ou seja, o conjunto TraIniStaAlt-adequado não é capaz de distinguir, em média, 70% dos mutantes de TraEveAlt.

As médias gerais de escore e *strength*, obtidas através da Tabela 5, são apresentadas nas Tabelas 6(a) e 6(b), respectivamente. Outro aspecto relevante é o custo associado a cada operador, ou seja, o número de mutantes gerados por cada operador, como é apresentado na Tabela 6(c).

Tabela 6 - Valores de (a) Escore (b) *Strength* (c) Número de Mutantes

(a)

Operador	Média Escore
TraDesStaAlt	0,892254
OutAlt	0,838878
TraEveAlt	0,819597
TraEveIns	0,806964
StaDel	0,806002
TraEveDel	0,791182
OutDel	0,787176
TraArcDel	0,746119
StaInsOutSep	0,540409
TraIniStaAlt	0,411288

(b)

Operador	Média *Strength*
TraEveIns	0,845118
TraDesStaAlt	0,452920
StaDel	0,225718
OutDel	0,223089
TraEveAlt	0,219523
TraEveDel	0,194585
OutAlt	0,191574
TraArcDel	0,157605
StaInsOutSep	0,050000
TraIniStaAlt	0,000000

(c)

Operador	Nº Mutantes
OutAlt	35,2
TraDesStaAlt	28,7
TraEveAlt	14,9
StaDel	14,6
TraEveIns	10,1
TraEveDel	9,0
OutDel	8,4
TraArcDel	6,8
TraIniStaAlt	2,9
StaInsOutSep	1,0

Considerando-se as informações apresentadas, a partir da Tabela 4 pode-se observar que o número de mutantes gerados por operador depende das características das MEFs. Por exemplo, o operador OutAlt gera 200 mutantes para a MEF 10 pelo fato do tamanho do seu conjunto de saída, como mostra a Tabela 2. Quanto ao operador TraDesStaAlt, o número de mutantes gerados já depende do número de estados e também do número de transições; por exemplo, considerando-se as MEFs 08 e 10, o número de mutantes gerados por esse operador é próximo, mas no caso da MEF 08 têm-se vários estados e poucas transições e na MEF 10 têm-se poucos estados e várias transições. Outra observação relevante diz respeito aos mutantes equivalentes que, para as MEFs completamente especificadas corresponde a um número pequeno e, nesses casos, a determinação da equivalência pode ser feita automaticamente [1]. Para as não completamente especificadas, que são na realidade de maior interesse para a aplicação da Análise de Mutantes, os mutantes equivalentes concentram-se nos operadores TraEveAlt e TraEveIns que exploram justamente a não completitude das MEFs e, nesses casos, heurísticas poderiam ser exploradas para determinar a equivalência.

Além disso, pelos resultados apresentados na Tabela 6, uma abordagem incremental de teste poderia ser estabelecida considerando-se o custo (número de mutantes gerados), ou seja, poderia ser estabelecido um conjunto ordenado de operadores de forma a se obter um escore de mutação 1 com um baixo custo. Uma das estratégias que poderia ser adotada para estabelecer esse conjunto seria aplicar os operadores, um a um, pela ordem crescente de custo (do operador StaInsOutSep para o operador OutAlt) de acordo com a Tabela 6(c) até ser obtido um escore igual a 1, para todas as máquinas. Uma outra estratégia, levando também em consideração o *strength* de cada operador e a relação de inclusão entre eles, poderia ser da seguinte maneira: observando-se a Tabela 6(c) os operadores TraIniStaAlt e StaInsOutSep são os que apresentam menor custo; no entanto, pela Tabela 6(b) esses operadores apresentam baixo *strength* podendo ser descartados numa primeira instância uma vez que conjuntos de teste adequados a outros operadores de maior *strength* provavelmente serão adequados também a esses operadores. O operador TraArcDel, próximo candidato de acordo com o custo, embora possua um *strength* relativamente baixo, deveria ser aplicado uma vez que o teste de todos os arcos (todas as transições) é considerado um requisito mínimo para a atividade de teste. Os três operadores seguintes – OutDel, TraEveDel e TraEveIns – poderiam ser analisados conjuntamente por possuírem um custo relativamente próximo. Nesse caso, observando-se o *strength* de cada um deles, o valor associado ao operador TraEveIns é bem superior aos demais, o que levaria a considerá-lo como o segundo operador a ser aplicado na atividade de teste; além disso, observando-se a Tabela 5 essa escolha se justifica também pela relação de inclusão, pois como pode ser visto, o conjunto TraEveIns-adequado distingue 63% dos mutantes de TraEveDel e 63% dos mutantes de OutDel, enquanto que os conjuntos TraEveDel-adequado e OutDel-adequado não distinguem mutante algum do operador TraEveIns. Em outras palavras, o operador TraEveIns seria aquele que promoveria o maior incremento de escore em relação à Análise de Mutantes, na perspectiva de se obter um escore de mutação igual a 1. O próximo operador a ser analisado, de acordo com a Tabela 6(c), é o StaDel. Analisando-se a relação de inclusão dos operadores TraArcDel e TraEveIns, já selecionados, verifica-se que não é necessária a aplicação desse operador, ou seja, a maioria de seus mutantes já estaria morta pelo conjunto {TraArcDel, TraEveIns}-adequado. Analogamente, o operador TraEveAlt também já é incluído pelos operadores previamente selecionados. O operador TraDesStaAlt possui um alto *strength* e a relação de inclusão dele pelos outros operadores é baixa, o que justifica também a sua escolha, constituindo o terceiro operador a ser aplicado. Finalmente, o operador OutAlt, pela Tabela 6(b) possui um *strength* dos mais baixos e, pela relação de inclusão seria incluído pelos outros três. Resumindo, uma

possível ordem de aplicação dos operadores, de forma incremental seria: TraArcDel, TraEveIns e TraDesStaAlt.

Utilizando-se esses três operadores ter-se-ia uma redução de custo de, aproximadamente, 66% e ainda assim, na maioria dos casos, obter-se-ia um escore de mutação em relação à Análise de Mutantes igual ou muito próximo de 1.

Essa abordagem incremental pode ser considerada uma forma de mutação restrita, mencionada na Seção 1, uma vez que, dependendo do tempo e custo destinados à atividade de teste, conjuntos específicos de operadores podem ser aplicados a fim de melhor atender os benefícios esperados.

A seguir, explora-se a determinação de um conjunto essencial de operadores de mutação, segundo o procedimento de Barbosa [16] que consta do Apêndice A.

4.3 Determinação do Conjunto Essencial para as MEFs Selecionadas

Cada passo do procedimento, apresentado com detalhes no Apêndice A, foi aplicado às dez especificações em teste. A seguir, comenta-se cada um dos passos e apresentam-se os resultados obtidos em cada um deles. Observa-se que no caso de MEFs são três as classes de mutação: transições, saídas e estados.

Passo 1: Selecionar operadores *op* cujos casos de teste *op*-adequados determinem alto escore de mutação em relação à Análise de Mutantes

Neste passo procura-se selecionar os operadores que determinam alto escore de mutação. Essa seleção é feita a partir de um índice médio de escore (IMES). O CE_{pre} é composto pelos operadores que, em média, determinem um escore de mutação em relação à Análise de Mutantes igual ou superior a IMES.

Utilizando-se IMES = 0.810 ± 0.005, os cinco primeiros operadores da Tabela 6(a) são selecionados para compor CE_{pre}. Portanto:

CE_{pre} = {TraDesStaAlt, OutAlt, TraEveAlt, TraEveIns, StaDel}

Passo 2: Procurar selecionar um operador de cada classe de mutação

Neste passo procura-se garantir que ao menos um operador de cada classe de mutação esteja presente em CE_{pre}. Para cada classe de mutação CM_i que não estiver representada em CE_{pre}, procura-se selecionar o operador $op \in CM_i$ que determine o maior escore e não seja incluído empiricamente pelos operadores de CE_{pre}. Neste caso, o CE_{pre} obtido no Passo 1 apresenta operadores de todas as classe de erros, portanto:

CE_{pre} = {TraDesStaAlt, OutAlt, TraEveAlt, TraEveIns, StaDel}

Passo 3: Avaliar inclusão empírica entre operadores

Neste passo é realizada uma análise com respeito à relação de inclusão empírica entre os operadores de CE_{pre} e aqueles que forem incluídos empiricamente são selecionados para compor o conjunto de operadores candidatos a serem eliminados de CE_{pre} (*CandElim*). A partir de *CandElim* elimina-se o operador *op* que for mais incluído empiricamente pelos demais operadores de CE_{pre}, mesmo que este seja o único operador representativo de uma determinada classe de mutação CM_i.

Sempre que um operador é eliminado, o conjunto *CandElim* deve ser gerado novamente. Este processo é repetido enquanto existirem operadores em CE_{pre} que sejam incluídos empiricamente pelos demais operadores de CE_{pre}.

As relações de inclusão empírica[2] observadas foram as seguintes:

{TraDesStaAlt, TraEveAlt, TraEveIns, StaDel} $\overset{1.000}{\Rightarrow}$ {OutAlt}

{TraDesStaAlt, OutAlt, TraEveIns, StaDel} $\overset{1.000}{\Rightarrow}$ {TraEveAlt}

{TraDesStaAlt, OutAlt, TraEveAlt, TraEveIns} $\overset{1.000}{\Rightarrow}$ {StaDel}

desse modo, *CandElim* = {OutAlt, TraEveAlt, StaDel}. Dos operadores de *CandElim*, todos foram igualmente incluídos, portanto de acordo com a Tabela 6(c), OutAlt é o que gera o maior número de mutantes e, por esse motivo, foi eliminado de CE_{pre}. Após a remoção de OutAlt, CE_{pre} foi avaliado novamente. Observou-se que os operadores TraEveAlt e StaDel continuaram a ser incluídos pelos demais operadores, fazendo o *CandElim* = {TraEveAlt, StaDel}. Da mesma forma, os operadores continuaram a ser igualmente incluídos, então o operador eliminado desta vez foi o TraEveAlt e assim foi feito até acabar todos os elementos do *CandElim*, observando que todos os elementos de *CandElim* foram eliminados de CE_{pre} por serem incluídos empiricamente. Portanto:

CE_{pre} = {TraDesStaAlt, TraEveIns}.

Passo 4: Estabelecer uma estratégia incremental de aplicação

O objetivo deste passo é estabelecer uma estratégia incremental de aplicação entre os operadores de CE_{pre}. Os operadores são ordenados de acordo com uma ordem de aplicação das classes de mutação e dentro de uma mesma classe os operadores são ordenados segundo seu custo. Definida a ordem em que os operadores de CE_{pre} serão aplicados, esta deve ser obedecida sempre que um operador for inserido no conjunto.

No caso, a ordem de aplicação deve ser: Erros de Transição, Saída e Estado. Essa ordem foi assim estabelecida considerando-se como requisito mínimo de teste a execução de todas as transições da MEF. Como CE_{pre} possui operadores somente da classe de erros de Transição, observando-se a Tabela 6(c), o operador TraEveIns gera menos mutantes que o operador TraDesStaAlt, portanto:

CE_{pre}={TaEveIns, TraDesStaAlt}

Passo 5: Selecionar operadores que proporcionem incremento no escore de mutação

Neste passo procura-se adicionar a CE_{pre} *x* operadores de cada classe de mutação, desde que tais operadores proporcionem um incremento no escore igual ou superior a um índice de incremento mínimo (IIM) preestabelecido e não seja incluído empiricamente pelos operadores de CE_{pre}. Para isso, a partir dos operadores com índice de incremento igual ou superior a IIM, são selecionados para fazer parte do conjunto de operadores candidatos a serem inseridos em CE_{pre} (*CandIns*) o operador *op* da classe CM_i que determine o melhor índice de incremento e demais operadores de CM_i, cujos índices de incremento encontrem-se na mesma faixa do índice de *op*. Dos operadores de *CandIns*, adiciona-se a CE_{pre} o operador de maior *strength* em relação a CE_{pre}. A inserção dos operadores em CE_{pre} é feita de forma intercalada, respeitando a ordem de classes estabelecida no Passo 4, até que *x* operadores de cada classe de mutação tenham sido adicionados a CE_{pre} ou quando não houver operadores com índice de incremento igual ou superior a IIM.

Para o nível de especificação, decidiu-se não aplicar esse passo, pois verificou-se em

[2] Neste passo *ms* = 0.99, ou seja, considera-se que o critério C_1 inclui empiricamente o critério C_2 se e somente se o escore de mutação que C_1 determina em relação a C_2 for igual ou superior a 0.99.

todas as especificações que nenhum operador proporcionava um incremento no escore, ou seja, todos os operadores que não pertenciam a CE_{pre} foram incluídos empiricamente.

Passo 6: Selecionar operadores de alto *strength*

Este passo determina quais operadores de alto *strength* devem ser inseridos em CE_{pre}. A seleção dos operadores é feita a partir de um índice médio de *strength* (IMS).

Os operadores de $\overline{CE}_{pre}$ que apresentarem *strength*, em relação à Análise de Mutantes, igual ou superior a IMS e não forem incluídos empiricamente pelos operadores de CE_{pre} são selecionados para fazer parte do conjunto de operadores de alto *strength* (CAS). Dos operadores de CAS, adiciona-se a CE_{pre} o operador de maior *strength* em relação a CE_{pre}.

Sempre que um novo operador é adicionado a CE_{pre}, o conjunto CAS deve ser gerado novamente. Este passo termina quando todos os operadores de alto *strength* que não são incluídos empiricamente por CE_{pre} tenham sido adicionados ao conjunto.

Utilizando-se IMS = 0.300 ± 0.005, observa-se na Tabela 6(b) que os operadores que apresentam um alto *strenght*, superiores a IMS, já pertencem a CE_{pre}, não formando o conjunto CAS. Após a aplicação do Passo 6 obtém-se o conjunto essencial final:

CE={TraEveIns,TraDesStaAlt}.

A Tabela 7 apresenta os conjuntos preliminares (CE_{pre}) e o conjunto essencial (CE) obtidos com a aplicação do procedimento, bem como o número de mutantes e o escore de mutação desses conjuntos em relação à Análise de Mutantes.

Tabela 7 - Operadores Obtidos a cada Passo do Procedimento

Passo	Operadores	Escore	Custo
1, 2	{TraDesStaAlt, OutAlt, TraEveAlt, TraEveIns, StaDel}	1,00000	78,64%
3	{TraDesStaAlt, TraEveIns}	0,999769	29,48%
4, 5, 6	{TraEveIns, TraDesStaAlt}	0,999769	29,48%

De acordo com a aplicação do Procedimento Essencial às dez MEFs selecionadas, observa-se que é possível reduzir o número de mutantes gerados, com uma redução de custo em torno de 70%, mantendo-se um escore de mutação muito próximo de 1. Além disso, o que foi observado é que quando a Máquina de Estado Finito apresenta a propriedade de ser completamente especificada o Conjunto Essencial reduz-se apenas ao operador TraDesStaAlt, que retrata erros de destino trocado; e, quando a máquina não é completamente especificada, o Conjunto Essencial é composto pelos operadores TraDesStaAlt e TraEveIns, que retratam erros de eventos extras. Isso ocorre pelo fato de que para as MEFs completamente especificadas, o operador TraEveIns não gera mutante algum e o escore de mutação que o operador TraDesStaAlt determina em relação aos demais é 1.000, ou seja, o conjunto TraDesStaAlt-adequado é capaz de distinguir 100% dos mutantes dos outros operadores, o que faz com que ele corresponda ao Conjunto Essencial para Máquinas de Estados Finitos Completamente Especificadas.

Comparando-se esse resultado com a abordagem incremental discutida anteriormente, nota-se que os dois operadores do conjunto essencial também fazem parte do conjunto determinado por aquela abordagem. Isso indica que dentro do escopo deste experimento os resultados apontam numa mesma direção, evidenciando que com a aplicação de um conjunto restrito de operadores de mutação consegue-se reduzir o custo de aplicação do critério Análise de Mutantes na validação de Máquinas de Estados Finitos sem prejuízo da qualidade da atividade de validação. Ainda, a partir do conjunto essencial, poder-se-ia determinar uma

seqüência para aplicação incremental de operadores com o objetivo de garantir o escore 1, o que certamente não implicaria um aumento de custo significativo considerando-se o alto escore já obtido pelo conjunto essencial.

5. Conclusão

Para a validação de Máquinas de Estados Finitos, uma das técnicas mais utilizadas na especificação de protocolos de comunicação, os métodos propostos na literatura são baseados fundamentalmente em modelos de erros. Mais recentemente, o critério Análise de Mutantes tem sido utilizado nesse contexto e as mutações definidas através dos operadores de mutação podem ser vistas como um modelo de erros e o critério pode ser considerado como uma alternativa para o teste de especificações baseadas em MEF. Vários experimentos, no nível de programa, mostram que esse critério é bastante eficaz para revelar erros. No entanto, um problema associado a ele é o alto custo de aplicação devido ao grande número de mutantes que podem ser gerados. Nesse sentido, várias alternativas são propostas para minimizar esse problema, entre elas a mutação restrita.

Neste artigo, através da condução de um estudo empírico utilizando-se 10 máquinas de estados finitos, com o apoio da ferramenta Proteum-RS/FSM, foram fornecidas evidências da viabilidade de aplicação do critério Análise de Mutantes na validação de MEFs. Procurando contribuir para a redução de seu custo de aplicação, foram exploradas duas abordagens: definição de uma estratégia de teste incremental e determinação de um conjunto essencial de operadores de mutação, utilizando o procedimento proposto por Barbosa *et al*. [16]. Os resultados mostram que é possível reduzir sensivelmente o número de mutantes gerados, mantendo um escore de mutação bastante elevado, próximo de 1 em relação à Análise de Mutantes; na realidade, eles contribuem para a definição de critérios de mutação restrita no contexto de MEFs e podem ser lidos na perspectiva de um modelo de erros, ou seja, caracterizam um modelo de erros mínimo que seria essencial no teste de MEFs e que garantiria uma cobertura dos demais erros típicos que têm sido abordados na literatura.

Os resultados obtidos motivam a continuidade dessa linha de pesquisa, procurando conduzir outros experimentos que visem à comparação com outros critérios de teste, o aumento do número de máquinas utilizadas e o uso de um amplo conjunto de seqüências de teste, para que os resultados tenham uma maior significância do ponto de vista estatístico.

Referências Bibliográficas

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

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

[3] **Fujiwara, S.; Bochmann, G.V.; Khendek,F.; Amalou, M.; Ghedamsi, A.** "Test Selection Based on Finite State Models", *IEEE Transaction on Software Engineering*, Vol. 17, N. 6, Junho, 1991.

[4] **Gonenc, G.** "A Method for the Design of Fault-Detection Experiments", *IEEE Transaction on Computer*, Vol C-19, pp 551-558, Junho, 1970.

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

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

[7] **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.

[8] **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.

[9] **Chung, A.; Sidhu, D.** "Fault Coverage of Probabilistic Test Sequences", in *Proceedings of the IFIP TC6 Third International Workshop on Protocol Test Systems*, North-Holland, pp. 305-316, 1991.

[10] **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 Communications Societies*, Vol.2, pp. 774-781, 1993.

[11] **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.

[12] **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, 1994.

[13] **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, Outubro, 1996.

[14] **Mathur, A. P.;** "Performance, Effectiveness, and Reliability Issues in Software Testing", *in Proceedings of the Fifteenth Annual International Computer Software and Applications Conference*, Tóquio, Japão, 1991, pp. 604-605.

[15] **Offutt, A.J.; Rothermel, G.; Untch, R.H.; Zapf, C.** "An Experimental Determination of Sufficient Mutant Operators", *ACM Transactions on Software Engineering Methodology*, accepted for publication, 1996.

[16] **Barbosa, E. F.; Vincenzi, A. M. R.; Maldonado, J. C.** "Uma Contribuição para a Determinação de um Conjunto Essencial de Operadores de Mutação no Teste de Programas C", in *Anais 12º Simpósio de Engenharia de Software*, 1998.

[17] **Tan, Q.M.; Petrenko, A.; Bochmann, G.v**. "A Test Generation Tool for Specifications in the Form of State Machines", http://www.iro.umontreal.ca/pub/teleinfo/TRs/P1016.ps.gz, 1996.

[18] **Bochmann, G.v.; Petrenko, A.** "Protocol Testing: Review of Methods and Relevance for Software Testing", in Proceedings of the ISSTA'1994 - International Symposium on Software Testing and Analysis, ACM - Software Engineering Notes, pp. 109-124, 1994.

[19] **Morell, L.J.** "A Theory of Fault-Based Testing", IEEE Transactions on Software Engineering, Vol.18, N.8, Agosto, 1990.

[20] **Sidhu, D.P.; Leung, T.K.** "Formal Methods for Protocol Testing: A Detailed Study", IEEE Transactions on Software Engineering, Vol.SE-15, N.4, pp.413-425, Abril, 1989.

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

[22] **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.

[23] **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, 1995.

[24] **Silva, G. S.; Pedrosa, A. C. P.** "Um sistema para geração automática de seqüências de teste para protocolos de comunicação", *in Anais dos trabalhos selecionados para o VIII Simpósio de Brasileiro de Redes de Computadores.*

[25] **Perrim, D. "Local Maps",** *in Lecture Notes in Computer Science, 316, Automata Networks***,** France, Maio, 1986, pp. 29-41.

[26] **Rapps, S.; Weyuker, E.J.** "Selecting Software Testing Data Using Data Flow Information", *IEEE Transactions on Software Engineering*, Vol.SE-11, pp.367-375, Abril, 1985.

Apêndice A - Procedimento Essencial

O Procedimento Essencial foi proposto por Barbosa *et al.*, para determinação de um conjunto essencial de operadores de mutação para a linguagem C. Ele é composto por 6 passos e para sua definição foi utilizada a terminologia reproduzida na Tabela A1, extraída de Barbosa *et al.* [16], bem como os seguintes conceitos que devem ser considerados.

Considere CM_1, CM_2, …, CM_n conjuntos que representam classes de mutação. O critério Análise de Mutantes (AM) utiliza, na sua concepção original, o conjunto formado por todos os operadores de mutação definido por $OP = CM_1 \cup CM_2 \cup \ldots \cup CM_n$. Qualquer subconjunto de operadores $SC \in 2^{\{op\}}$ constitui um critério de mutação restrito.

Tabela A1 - Terminologia utilizada no Procedimento Essencial [16]

Termo	Descrição/Definição
$\bigcup_{i=1}^{n} CM_i$	Classes de operadores de mutação (para a ferramenta Proteum-RS/FSM, n = 3 classes: transições, saídas e estados).
OP	Conjunto de todos os operadores de mutação $OP = CM_1 \cup CM_2 \cup ... \cup CM_n$
CE	Conjunto essencial de operadores de mutação ($CE \subseteq OP$)
CE_{pre}	Conjunto essencial preliminar de operadores de mutação ($CE_{pre} \subseteq OP$)
$\overline{CE}_{pre}$	Complemento de CE_{pre} ($\overline{CE}_{pre} = OP - CE_{pre}$)
CandElim	Conjunto de operadores de CE_{pre} candidatos a serem eliminados em virtude da relação de inclusão empírica ($CandElim \subseteq CE_{pre}$)
CandIns	Conjunto de operadores de $\overline{CE}_{pre}$, pertencentes a determinada classe CM_i, candidatos a serem inseridos em CE_{pre} ($CandIns \subseteq (\overline{CE}_{pre} \cap CM_i)$)
CAS	Conjunto de operadores de alto *strength* ($CAS \subseteq \overline{CE}_{pre}$)
x	Número de operadores de cada classe CM_i que se deseja incluir em CE_{pre} (definido pelo testador)
ms	Escore de mutação (definido pelo testador)
IMES	Índice médio de escore (definido pelo testador)
IIM	Índice de incremento mínimo (definido pelo testador)
IMS	Índice médio de *strength* (definido pelo testador)
$f_{MS}(C_1, C_2)$	Função que retorna o escore de mutação determinado por C_1 em relação a C_2
f_{MSM} (op)	Função que retorna a média dos escores que *op* determina em relação a cada operador de OP
$f_{STR}(C_1, C_2)$	Função que retorna o *strength* de C_1 em relação a C_2
f_{STRM} (op)	Função que retorna a média dos *strength* de *op* em relação a cada operador de OP
f_{INCR}(op, C_1)	Função que retorna o índice de incremento que *op* proporciona em relação a Análise de Mutantes se for adicionado a C_1 (este índice é truncado após o primeiro dígito significativo)

Dado um critério de mutação restrito *SC*, diz-se que um conjunto de casos de teste **T** é SC-adequado se **T** obtiver um escore de mutação igual a 1 em relação aos mutantes gerados pelos operadores de *SC*; ou seja, se **T** for capaz de revelar as diferenças de comportamento existentes entre **S** (especificação em teste) e os mutantes não equivalentes gerados pelos operadores de *SC*. Quando *SC* for composto por um único operador de mutação *op* (*SC* = {*op*}), em alguns casos, por simplicidade de notação, será utilizado *op*.

Na prática, por limitações de tempo e custo, obter-se um escore de mutação próximo de 1 pode ser satisfatório. Seja *ms* um escore de mutação definido pelo testador. Dados um critério C e um conjunto de casos de teste **T**, diz-se que **T** é empiricamente adequado a C (T é C-adequado*) se **T** obtiver um escore de mutação igual ou superior a *ms*.

Uma caracterísitca importante utilizada na comparação de critérios de teste é a relação

de inclusão, definida por Rapps e Weyuker [26]. Dados dois critérios C_1 e C_2, diz que C_1 inclui C_2 ($C_1 \Rightarrow C_2$) se para todo conjunto de casos de teste T_1 C_1-adequado, T_1 é C_2-adequado e existe um T_2 C_2-adequado que não é C_1-adequado; C_1 e C_2 são equivalentes se para qualquer T C_1-adequado, T é C_2-adequado e vice-versa.

Além disso, diz-se que C_1 inclui empiricamente C_2 com um determinado escore *ms* ($C_1 \overset{ms}{\Rightarrow}{}^{*} C_2$) se para todo conjunto de casos de teste T_1 C_1-adequado, t_1 é C_2-adequado* para um dado ms e existe um T_2 C_2-adequado que não é C_1-adequado*; C_1 e C_2 são equivalentes* se para qualquer T C_1-adequado, T é C_2-adequado* e vice-versa.

No Quadro A1 é apresentado o algoritmo do Procedimento Essencial:

$CE_{pre} \leftarrow \{op_j \in OP \mid f_{MSM}(op_j) \geq IMES\}$

para *i* de 1 até *n* faça
  se $CM_i \cap CE_{pre} = \emptyset$ então
    se $\exists\ op_j \in CM_i \mid CE_{pre} \overset{ms}{\nRightarrow}{}^{*} \{op_j\} \wedge f_{MSM}(op_j) \geq f_{MSM}(op_k), \forall\ op_k \in \{op \in CM_i \mid CE_{pre} \overset{ms}{\nRightarrow}{}^{*} \{op\}\}$
      então $CE_{pre} \leftarrow CE_{pre} \cup \{op_j\}$
fim para

$CandElim \leftarrow \{op_j \in CE_{pre} \mid (CE_{pre} - \{op_j\}) \overset{ms}{\Rightarrow}{}^{*} \{op_j\}\}$
enquanto $CandElim \neq \emptyset$ faça
  $CE_{pre} \leftarrow CE_{pre} - \{op\} \mid op \in CandElim \wedge f_{MS}(CE_{pre} - \{op\}, \{op\}) \geq f_{MS}(CE_{pre} - \{op_k\}, \{op_k\}),$
    $\forall\ op_k \in CandElim$
  $CandElim \leftarrow \{op_j \in CE_{pre} \mid (CE_{pre} - \{op_j\}) \overset{ms}{\Rightarrow}{}^{*} \{op_j\}\}$
fim enquanto

Estabelecer uma ordenação de acordo com os requisitos de teste que se deseja priorizar

$cont\_oper \leftarrow 1$
enquanto $cont\_oper \leq x$ faça
  para *i* de 1 até *n* faça
    $CandIns \leftarrow \{op \in \overline{CE}_{pre} \cap CM_i \mid f_{INCR}(op, CE_{pre}) \geq IIM$
      $\wedge\ (\forall\ op_j \in \overline{CE}_{pre} \cap CM_i, f_{INCR}(op_j, CE_{pre}) \leq f_{INCR}(op, CE_{pre}))\}$
    se $CandIns \neq \emptyset$ então
      $CandIns \leftarrow CandIns \cup \{\forall\ op_k \in \overline{CE}_{pre} \cap CM_i \mid$
        $op_k \neq op \wedge f_{INCR}(op, CE_{pre}) - f_{INCR}(op_k, CE_{pre}) = 0\}$
      $CE_{pre} \leftarrow CE_{pre} \cup \{op\} \mid op \in CandIns \wedge CE_{pre} \overset{ms}{\nRightarrow}{}^{*} \{op\} \wedge$
        $f_{STR}(\{op\}, CE_{pre}) \geq f_{STR}(\{op_j\}, CE_{pre}), \forall\ op_j \in CandIns$
    fim se
  fim para
  $cont\_oper \leftarrow cont\_oper + 1$
fim enquanto

$CAS \leftarrow \{\forall\ op_j \in \overline{CE}_{pre} \mid f_{STRM}(op_j) \geq IMS \wedge CE_{pre} \overset{ms}{\nRightarrow}{}^{*} \{op_j\}\}$
enquanto $CAS \neq \emptyset$ faça
  $CE_{pre} \leftarrow CE_{pre} \cup \{op\} \mid op \in CAS \wedge f_{STR}(\{op\}, CE_{pre}) \geq f_{STR}(\{op_k\}, CE_{pre}), \forall\ op_k \in CAS$
  $CAS \leftarrow \{\forall\ op_j \in \overline{CE}_{pre} \mid f_{STRM}(op_j) \geq IMS \wedge CE_{pre} \overset{ms}{\nRightarrow}{}^{*} \{op_j\}\}$
fim enquanto
$CE \leftarrow CE_{pre}$

Quadro A1 – Algoritmo do Procedimento Essencial [16]