Resultados de uma Aplicação do Critério Análise de Mutantes e do Critério Todos-Potenciais-Usos Restritos

Inali Wisniewski Soares
inali@unicentro.br
Departamento de Informática
Universidade Estadual do Centro-Oeste - UNICENTRO
CP: 730, 85015-430, Guarapuava, PR

Silvia Regina Vergilio
silvia@inf.ufpr.br
Departamento de Informática
Universidade Federal do Paraná - UFPR
CP: 19081, 81531-970, Curitiba, PR

Resumo

Critérios de teste de software visam conduzir e avaliar a qualidade da atividade de teste. Entre os vários critérios propostos, destacam-se os critérios estruturais, baseados em fluxo de controle e baseados em fluxo de dados, e o critério Análise de Mutantes, baseado em erros. O critério Análise de Mutantes é um critério bastante atrativo, porém possui um alto custo computacional. Os "Critérios Restritos" foram propostos para aumentar a eficácia dos critérios estruturais, pois possuem uma grande habilidade para detectar erros. Este trabalho apresenta resultados de uma comparação entre os Critérios Restritos e o critério Análise de Mutantes, considerando os fatores eficácia, em termos de número de erros revelados e, custo, em termos do número de casos de testes necessários.

Palavras-chave: critério de teste estrutural, Critérios Restritos, Análise de Mutantes.

1. Introdução

O desenvolvimento de software está sujeito a vários tipos de erros, por isso, na Engenharia de Software, atividades de teste e validação são fundamentais para garantir a qualidade de software [Pre92]. O principal objetivo do teste é revelar erros; um bom caso de teste é aquele que possui uma grande probabilidade de encontrar um erro ainda não descoberto.

Para obter testes eficazes e de baixo custo, é fundamental a aplicação de técnicas e critérios de teste. Entre os vários critérios propostos, destacam-se os critérios estruturais, baseados em fluxo de controle e baseados em fluxo de dados [Nta84, URA88, Mal91, RAP82], e o critério Análise de Mutantes [Dem78], baseado em erros.

A idéia do critério Análise de Mutantes [Dem78] é criar a confiança de que um programa P está correto, produzindo-se um conjunto de programas chamados mutantes de P, modificados através de simples alterações sintáticas e construir casos de teste que provoquem diferença de comportamento entre P e seus mutantes.

Gerar conjuntos de dados de teste é o processo de identificar dados de entrada para o programa e assim satisfazer um critério selecionado. São encontradas diversas categorias de técnicas utilizadas para gerar dados de teste para os critérios existentes. Uma delas é a geração de dados de teste sensíveis a erros, que possui elevada probabilidade de revelar alguns tipos de erros específicos. Estão incluídas nessa categoria as técnicas: Teste de Domínio [WC80]; Teste Baseado em Restrições [DMO91]; Teste de Operador Relacional e Booleano e Teste Operador Relacional e Booleano com parâmetro ϵ [Tai93].

Os critérios de testes são vistos como complementares e devem ser utilizados em conjunto a fim de obter uma atividade de teste de boa qualidade. Nesse sentido, a realização de estudos empíricos intensificou-se nos últimos anos, procurando, através da comparação entre os critérios existentes, obter uma estratégia que seja eficaz para revelar a presença de erros no programa e que apresente um baixo custo de aplicação [Mal97].

O critério Análise de Mutantes tem se mostrado mais eficaz em termos de número de erros revelados que os critérios estruturais baseados em fluxo de dados, porém mais custoso em termos de número de casos de teste e número de execuções [WMM94, Sou96].

Com o objetivo de aumentar a eficácia dos critérios estruturais, Vergilio [Ver97], introduziu os "Critérios Restritos". Esses critérios permitem que os critérios estruturais sejam utilizados juntamente com os princípios de

técnicas de geração de dados de testes sensíveis a erros, a idéia básica é inserir uma restrição que descreva um tipo particular de erro, e que essa restrição seja satisfeita pelo dado de teste gerado, para satisfazer um dado critério estrutural.

Vergilio [Ver97] utilizando uma extensão da ferramenta POKE-TOOL [Cha91] realizou um experimento de validação dos Critérios Restritos, onde foi comparado o critério todos-potenciais-usos (um critério estrutural baseado em fluxo de dados) com o critério todos-potenciais-usos restritos. Resultados mostraram um aumento da eficácia dos dados gerados, aplicando-se os Critérios Restritos.

O objetivo desse trabalho é apresentar resultados preliminares de um experimento que está sendo realizado para comparar os Critérios Restritos com o critério Análise de Mutantes, considerando os fatores eficácia, em termos de número de erros revelados e custo, em termos do número de casos de testes necessários.

Esse texto está organizado da seguinte forma: na Seção 2 são introduzidos a terminologia, os principais conceitos e ferramentas de teste utilizados nesse trabalho. A Seção 3 descreve como foram coletados os dados e realizada a comparação entre os critérios, e a Seção 4 contém as conclusões.

2. Trabalhos Relacionados

Diversas técnicas de teste têm sido propostas para selecionar bons casos de teste e assim revelarem a maioria dos erros utilizando um mínimo de tempo e esforço. Geralmente as técnicas utilizadas são: Técnica Funcional - deriva os casos de teste baseados na especificação do software; Técnica Estrutural - baseia-se na estrutura interna do programa para derivar os casos de teste; Técnica Baseada em Erros - os casos de teste são obtidos através do conhecimento dos principais erros que geralmente ocorrem durante o processo de desenvolvimento do software.

Essas técnicas devem ser vistas como complementares e as melhores vantagens de cada uma devem ser exploradas produzindo dessa forma testes eficazes e de baixo custo. Elas estabelecem critérios de teste, que podem ser utilizados tanto para auxiliar na geração de conjuntos de dados de teste como para auxiliar na avaliação da adequação desses conjuntos.

Entre os vários critérios propostos para se conduzir e avaliar a qualidade da atividade de teste, destacam-se os critérios estruturais, baseados em fluxo de controle e baseados em fluxo de dados [Nta84, URA88, Mal91, RAP82], e o critério Análise de Mutantes [Dem78], baseado em erros.

A técnica baseada em erro utiliza informações sobre os tipos de erros mais freqüentes no processo de desenvolvimento de software para derivar os requisitos de teste. Análise de Mutantes é um critério baseado em erros, a idéia básica desse critério foi apresentada por DeMillo em 1978 [Dem78], conhecida como hipótese do programador competente, assume que programadores experientes escrevem programas corretos ou muito próximos do correto. De acordo com essa hipótese, os erros nos programas são introduzidos através de pequenos desvios sintáticos e produzem um resultado incorreto. O objetivo é revelar esses erros, então programas mutantes são gerados a partir do programa em teste. Casos de teste devem ser construídos para diferenciar o comportamento do programa original do comportamento de um programa mutante. O mutante é morto quando ocorre essa diferença. O critério Análise de Mutantes exige que todos os mutantes sejam mortos.

O critério Análise de Mutantes pode se tornar caro, porque exige múltiplas execuções do programa em teste e dos mutantes gerados. A determinação de mutantes equivalentes é a principal limitação desse critério. Os programas mutantes equivalentes possuem comportamento idêntico ao programa original, pois eles computam a mesma função. Determinar se dois programas são ou não equivalentes é uma questão indecidível e em geral, só é possível através de heurísticas [BS79].

Os critérios estruturais baseados em fluxo de dados, têm como objetivo testar definição e conseqüentes usos de variáveis. Uma definição de variável ocorre sempre que um novo valor é armazenado na posição de memória correspondente e um uso ocorre quando esse valor é referenciado [RAP82]. Nesse trabalho é utilizado apenas o critério todos-potenciais-usos da família de Critérios Potenciais Usos definida por Maldonado et al [MCJ88], como representante dos critérios estruturais. Eles estão baseados num grafo de programa onde nós representam conjuntos de comandos executados seqüencialmente e arcos representam os desvios de fluxo de controle do programa. Uma seqüência de nós deste grafo é considerado um caminho.

O critério todos-potenciais-usos, exige que todas as associações (i,(j,k),x) sejam exercitadas, onde o nó i possui uma definição de x e existe no grafo um caminho livre de definição com respeito a x de i para o arco (j,k) ou nó j [Mal91].

Os Critérios Restritos introduzidos por Vergilio [Ver97], são extensões para as diversas famílias de critério de teste estrutural para aumentar a eficácia desses critérios, ou seja, para aumentar a capacidade desses critérios em revelarem erros, utilizando os fundamentos de técnicas de geração de dados sensíveis a erros [WC80,

DMO91, Tai93]. A cada elemento requerido é associada uma restrição a ser satisfeita durante a execução do caminho que exercitará esse elemento. Mais de um caminho candidato a cobrir cada elemento requerido pode ser exercitado, ou ainda um caminho poderá ser executado com um ou mais dados de teste que satisfarão as restrições que descrevem erros e portanto têm uma maior probabilidade de revelar um erro ainda não descoberto.

Critérios Restritos requerem elementos restritos, um elemento requerido é assim denominado porque seu domínio de entrada foi restringido. Quando uma restrição é associada a um elemento, o domínio de entrada do elemento restrito gerado fica restrito aos pontos que satisfazem a restrição dada.

O critério todos-potenciais-usos restritos requer associações restritas. A cada associação requerida pelo critério todos-potenciais-usos uma ou mais restrições são associadas. Uma associação restrita é dada por ((i,(j,k),x),C) e será coberta se um caminho livre de definição c.r.a x, ou seja, um caminho que cobre a associação correspondente for executado, e se C for satisfeita durante a sua execução [Ver97].

A principal limitação dos critérios estruturais é a existência de caminhos não executáveis. Um caminho é não executável se não existe um conjunto de valores para as variáveis de entrada do programa que causam a sua execução. Determinar se um caminho é ou não executável é uma questão indecidível [FRA87].

A seguir são descritas brevemente as ferramentas POKE-TOOL e Proteum que estão sendo utilizadas no experimento:

a) **POKE-TOOL:** A ferramenta de teste POKE-TOOL [Cha91], desenvolvida no DCA/FEEC/Unicamp, foi implementada em linguagem C para possibilitar o uso dos Critérios Potenciais Usos. Ela permite a execução de um conjunto de casos de teste e faz a análise de adequação do conjunto executado em relação a esses critérios. Uma extensão à ferramenta POKE-TOOL foi proposta em [Ver97] e implementada para dar apoio a utilização dos Critérios Potenciais-Usos Restritos.

b) **Proteum:** A ferramenta Proteum [Del93], desenvolvida no ICMSC-USP, apóia o teste de mutação para programas C. A ferramenta Proteum oferece recursos ao testador para, através da aplicação do critério Análise de Mutantes, avaliar a adequação de, ou gerar um conjunto de casos de teste T para determinado programa P.

Devido a diversidade de critérios de teste existentes, é difícil saber qual critério utilizar e como utilizá-los de forma complementar para obter o melhor resultado com o menor custo. Os estudos empíricos buscam, através da comparação entre os critérios, uma estratégia eficaz para revelar a presença de erros no programa e que também apresente um baixo custo de aplicação [Mal97].

Segundo Wong [Won93], custo, eficácia e dificuldade de satisfação (*strength*), são fatores básicos para comparar critérios de teste. Custo: é medido através do número de casos de teste requeridos para satisfazer o critério, e representa o esforço necessário para a utilização do mesmo. Eficácia: é a capacidade de um critério em revelar uma quantidade maior de erros em relação a outro. Dificuldade de satisfação (*strength*): é verificar o quanto consegue-se satisfazer um critério C_1 tendo satisfeito um critério C_2.

A próxima seção descreve uma aplicação que possibilitou a comparação entre o Critério Restrito e o critério Análise de Mutantes.

3. Critério todos-potenciais-usos restritos e critério Análise de Mutantes

Essa seção apresenta resultados da aplicação do critério todos-potenciais-usos restritos e do critério Análise de Mutantes para 3 programas: *cal, tr e checkeq*. Inicialmente serão descritos os passos seguidos durante a aplicação dos critérios. Esses passos estão sendo utilizados em um experimento maior [Soa00].

Para realizar este experimento estão sendo utilizadas as ferramentas Proteum [Del93] e POKE-TOOL [Cha91] para teste de programas no nível de unidade. Os programas utilizados são programas utilitários UNIX escritos na linguagem C, que são de domínio público. Tais programas fazem parte de um benchmark usado inicialmente por Wong [WMM94] e posteriormente por Vergilio [Ver97]. O motivo para escolher esses programas foram:

1. a existência de um conjunto de programas incorretos para os programas a serem testados, cujos erros encontram-se classificados em diferentes categorias, o que permite a análise da eficácia dos critérios.
2. estão sendo utilizados os mesmos conjuntos de restrições e de testes gerados no experimento de Vergilio [Ver97], não sendo necessário repetir o experimento com os critérios todos-potenciais-usos e todos-potenciais-usos restritos. Dados sobre estes conjuntos encontram-se na Tabela 1.

A aplicação do experimento consistiu de duas fases principais: satisfazer o critério Análise de Mutantes e verificar o custo e a eficácia dos critérios envolvidos.

I. Satisfação do Critério Análise de Mutantes:

1. Geração dos programas mutantes: para geração dos mutantes foram utilizados todos os operadore disponíveis na ferramenta Proteum [Del93].

2. Satisfação do critério Análise de Mutantes. Isso envolveu:
   - Utilização dos mesmos conjuntos de dados de testes gerados de maneira "Ad-hoc" e aleatória utilizados em [Ver97];
   - Submissão do conjunto à ferramenta Proteum e obtenção da cobertura inicial;
   - Elaboração de um conjunto adicional de casos de teste, submissão do conjunto à ferramenta Proteum e obtenção da cobertura;
   - Análise dos mutantes vivos para determinar a equivalência: Os mutantes foram analisados para cada programa, foram determinados os mutantes equivalentes e novos casos de teste foram submetidos à ferramenta Proteum para matar os mutantes restantes. A cobertura é então obtida.

II. Custo e eficácia dos critérios:

A Tabela 1 apresenta o número de elementos requeridos e a cardinalidade dos conjuntos PU, PU-R e AM adequados respectivamente aos critérios todos-potenciais-usos, todos-potenciais-usos restritos e análise de mutantes para os programas: *cal, tr e checkeq*.

Essa tabela também apresenta a cobertura obtida para os três critérios. A cobertura não é 100% pois o número de elementos não executáveis e mutantes equivalentes não foram excluídos e são apresentados na última coluna.

Tabela 1 - Custo dos Critérios PU, PU-R e AM

Programa	Critério	NºElementos Requeridos	Nº de Casos de Teste	Cobertura (%)	Mutantes Equivalentes /Elementos não executáveis
cal	PU	242	17	71,49 (%)	69 (28,51 %)
	PU-R	488	17	63,11 (%)	180 (36,89 %)
	AM	4324	32	93,00 (%)	305 (07,00 %)
tr	PU	1527	30	38,90 (%)	933 (61,10 %)
	PU-R	3040	35	33,75 (%)	2014 (66,25 %)
	AM	4422	72	82,13 (%)	790 (17,87 %)
checkeq	PU	582	76	80,41 (%)	114 (19,59 %)
	PU-R	1539	164	69,40 (%)	471 (30,60 %)
	AM	3075	74	93,00 (%)	214 (07,00 %)

Através dos dados iniciais pode-se comparar o tamanho dos conjuntos adequados aos critérios, em termos de números de casos de teste. Pode-se observar que para os programas *cal* e *tr*, não existem grandes diferenças em termos de custo entre os critérios PU e PU-R, embora a cobertura do critério PU-R seja ligeiramente menor e este apresente um maior número de elementos não executáveis, o custo do critério AM para o programa *cal* foi 88,23% maior que o custo de PU e PU-R, e para o programa *tr*, 140% maior que PU e 105,71% maior que PU-R.

No entanto, para o programa *checkeq*, o resultado é bastante diferente. O custo do critério AM foi 45% inferior ao critério PU-R e 2,6% inferior ao critério PU. Uma explicação para isso é a estrutura desse programa, que possui uma grande quantidade de variáveis e if's, que faz com que um grande número de elementos sejam requeridos pelo critério PU; Outra explicação, é o tipo de restrição usada no experimento (restrições BOR) que se aplicam a predicados compostos. A maioria dos if's do programa *checkeq* possui predicados compostos. Esses fatos ocasionam o aumento do número de casos de testes necessários.

Wong [WMM94] derivou para cada programa um conjunto de versões incorretas. Cada versão tem somente um erro simples. Através dessas versões é possível analisar a eficácia dos critérios. Os erros são classificados em erros de domínio, computacional e estrutura de dados [WC80, Ver97]. Os conjuntos de testes adequados ao critério Análise de Mutantes foram usados para executar as versões incorretas de cada programa. A Tabela 2 apresenta a análise de eficácia para os programas: *cal, tr e checkeq*, e descreve os dados obtidos no experimento de Vergilio [Ver97] para os critérios PU, PU-R e AM. Pode-se observar que o critério AM foi mais eficaz, ele revelou 3,22 % mais erros que o critério PU e 1,61% mais erros que o critério PU-R.

Tabela 2 - Eficácia dos Critérios PU, PU-R e AM

Programa	Critério	Erros Revelados	Erros não Revelados
cal	PU	19 (95%)	1 (5%)
	PU-R	20 (100%)	
	AM	20 (100%)	
tr	PU	20 (100%)	
	PU-R	20 (100%)	
	AM	20 (100%)	
checkeq	PU	20 (90,91%)	2 (9,09%)
	PU-R	20 (90,91%)	2 (9,09%)
	AM	21 (95,46%)	1 (4,54%)

4. Conclusões

Esse trabalho apresentou resultados preliminares de um estudo realizado com o objetivo de comparar o critério todos-potenciais-usos restritos e o critério Análise de Mutantes, considerando os fatores eficácia e número de casos de testes necessários.

Os resultados mostram evidências que o critério Análise de Mutantes é o mais caro dos três, exceto quando o programa em teste possui estruturas que podem aumentar o número de elementos requeridos para um critério estrutural: o programa *checkeq* exemplifica isso. Quanto à eficácia, os Critérios Restritos se mostraram mais eficazes que o correspondente critério estrutural porém menos eficazes que o critério Análise de Mutantes.

Os resultados iniciais mostram evidências de que os Critérios Restritos se constituem critérios intermediários entre os estruturais e o critério Análise de Mutantes em termos de custo e eficácia, no entanto, somente o experimento que está sendo finalizado e futuros estudos poderão comprovar essas evidências. Esses estudos empíricos são importantes para estabelecer uma estratégia que auxilie a utilização das diversas técnicas e critérios de teste.

Agradecimentos

As autoras agradecem à W. Eric Wong (Telecordia Technologies) e ao Prof. José Carlos Maldonado (USP – São Carlos) por fornecerem os programas e casos de teste utilizados no experimento.

Referências Bibliográficas

[BS79] D. Baldwin and F. Sayward. *Heuristics for Determining Equivalence of Program Mutations*. CT, Res. Rep. 276, Department of Computer Science - Yale University, New Haven, 1979.

[Cha91] M.L. Chaim. POKE-TOOL - *Uma ferramenta para Suporte ao Teste Estrutural de Programas Baseado em Análise de Fluxo de Dados*. Master Thesis, DCA/FEEC/Unicamp, Campinas - SP, Brazil, April 1991. (in Portuguese).

[Del93] Delamaro, M.E., "*Proteum - Um ambiente de teste baseado na Análise de Mutantes*", Dissertação de Mestrado, ICMSC/USP - São Carlos, SP, Brasil, Outubro, 1993.

[Dem78] R.A De Millo. *Software Testing and Evaluation*. The Benjamin/Cummings Publishing Company, Inc. 1987.

[DMO91] R.A. De Millo and A.J. Offutt. *Constraint-based automatic test data generation*. IEEE Transactions on Software Engineering, Vol. SE-17(9):900-910, September 1991.

[FRA87] F.G. Frankl. *The use of Data Flow Information for the Selection an Evaluation of Software Test Data*. PhD Thesis, Department of Computer Science, New York University, New York, U.S.A., October 1987.

[Mal91] J.C. Maldonado. *Critérios Potenciais Usos: Uma Contribuição ao Teste Estrutural de Software.* Doctorate Dissertation, DCA/FEEC/Unicamp, Campinas - SP, Brazil, July 1991.

[Mal97] J.C. Maldonado. *Critérios de Teste de Software: Aspectos Teóricos, Empíricos e de Automatização.* Livre Docência, ICMC-USP, São Carlos, SP, Janeiro 1997.

[MCJ88] J.C. Maldonado, M.L. Chaim, and M. Jino. Seleção de casos de teste baseada nos critérios potenciais usos. *In II Simpósio Brasileiro de Engenharia de Software*, pages 24-35. Sociedade Brasileira de Computação – SBC, Canela – RS, Brazil, October 1988 (in Portuguese).

[Nta84] Ntafos, S.C.; *"On Requirement Element Testing"*, IEEE Transactions on Software Engineering, SE-10(16), Novembro, 1984.

[Pre92] R.B. Pressman. *Software Engineering: A Practitioner's Approach.* McGraw-Hill, New-York, EUA, third edition, 1992.

[RAP82] S. Rapps, E.J. Weyuker, *"Data Flow Analysis Techniques for Test Data Selection"*, In Proc. Int. Conf. Software Eng., Tokio, Japão, Set 1982.

[Soa00] I.W. Soares. *Análise de Mutantes e Critérios Restritos: Uma avaliação empírica.* Dissertação de Mestrado, UFPR, Curitiba - PR, 2000. (em elaboração).

[Sou96] S.R.S. Souza. *Avaliação do Custo e Eficácia do Critério Análise de Mutantes na Atividade de Teste de Programas.* Dissertação de Mestrado, ICMSC/USP, São Carlos-Brazil, June 1996. (em Português).

[Tai93] K.C. Tai. *Predicate-based teste generation for computer programs.* In Proceedings of International Conference on Software Engineering, pages 267-276. IEEE Press, May 1993.

[URA88] Ural, H. & Yang, B.; "A Structural Test Selection Criterion", *Information Processing Letters*, 28, 1988, pp. 157-163.

[Ver97] S.R. Vergilio. *Critérios Restritos de Teste de Software: Uma Contribuição para Gerar Dados de Teste mais eficazes. Doctorate Dissertation*, DCA/FEEC/Unicamp, Campinas - SP, Brazil, July 1997. (em Português).

[WC80] L.J. White and E.I. Cohen. *A domain strategy for computer program testing. IEEE Transactions on Software Engineering*, Vol. SE-6(3):247-257, May 1980.

[Won93] W.E. Wong. – *On Mutation and Data Flow.* PhD Thesis, Department of Computer Science, Purdue University, West Lafayette-IN, USA, December 1993.

[WMM94] W.E. Wong, A.P. Mathur, and J.C. Maldonado. *Mutation versus all-usos: An Empirical evaluation of cost, strength and effectivenes.* In Software Quality and Producrivity-Theory, Pratice, Education and Training, Hong Kong, December 1994.