9ª MOSTRA ACADÊMICA UNIMEP
08 a 10/11 de 2011
Tema: "Ambiente e Sustentabilidade"

9º Congresso de Pós-Graduação

INVESTIGANDO A EXISTÊNCIA DOS MUTANTES RESISTENTES

Autor(es)

REGINA HELENA MORETTI

Orientador(es)

PLÍNIO R. S. VILELA

1. Introdução

Critérios de teste têm sido tradicionalmente utilizados como importante ponto de apoio para decidir se a quantidade de testes executados foi suficiente para garantir a qualidade de um dado programa. O critério provê uma medida da cobertura dos testes feitos, influenciando, portanto, a decisão de quando parar a execução dos testes. A análise de mutantes é um critério de teste baseado na inserção de modificações sintáticas nos programas, simulando a ocorrência de defeitos comuns. Essas modificações são inseridas de maneira sistemática através da utilização de operadores de mutação. A cada inserção feita um novo programa é gerado, ou seja, somente uma modificação está presente em cada novo programa criado. Esses programas são chamados de mutantes. A noção de cobertura vem da quantidade de programas mutantes que são diferenciados do original pela execução dos dados de teste, cada mutante diferenciado é dito morto e a cobertura é dada pela proporção de mutantes mortos frente ao total de mutantes (não equivalentes) gerados. Esta técnica foi originalmente proposta por DeMillo, Lipton e Sayward (1978) e desde então vem sendo intensamente estudada e aplicada principalmente na mutação de programas (teste estrutural) e na mutação de especificações [Jia e Harman 2009]. 1.1. Análise de Mutantes A idéia básica do processo de teste de mutantes é que após submeter um programa P a um determinado conjunto de casos de testes T e efetuar as correções necessárias para remover os defeitos encontrados, é determinada uma medida que indica o quão adequadamente os dados de teste testaram P. Para determinar esta medida, chamada de escore de mutação, são geradas mutações do programa original P, através da introdução de uma única variação sintática em P para cada mutante M gerado. Para cada variação sintática, existe uma chance de que uma variação semântica seja produzida. Estas modificações sintáticas são inseridas no programa original através da aplicação de um conjunto de regras chamadas de operadores de mutação. Estas regras são dependentes da linguagem de programação utilizada. Operadores de mutação típicos, por exemplo, substituem cada operando por todos os outros operandos sintaticamente válidos ou modificam expressões substituindo operadores e inserindo novos operadores ou ainda apagando instruções inteiras [Ma e Offutt 2002], [Ma, Kwon, Offut 2005]. Um conjunto de dados de testes capaz de diferenciar um grande número de mutantes é considerado adequado para testar P. Para diferenciar o programa original e um mutante, ambos são submetidos ao mesmo conjunto de dados de entrada. Se os resultados diferem, o mutante é marcado como morto; e a confiança nos dados de teste aumenta. Isso porque, uma vez que os dados de teste foram capazes de diferenciar o programa original de um mutante, se espera que se houvessem defeitos no programa esses dados de teste também seriam adequados para detectá-los. O escore de mutação representa o percentual de mutantes eliminados. Toda vez que chegamos a uma pontuação de mutação não suficiente, então novos casos de testes são acrescentados ao conjunto original, e todo o processo é refeito, aumentando a qualidade dos casos de testes e da atividade de testes como um todo.

2. Objetivos

Este artigo apresenta e analisa dois experimentos que exploram a análise de mutantes, com o objetivo de demonstrar a existência e avaliar a eficácia na detecção de defeitos de um determinado grupo de mutantes, os mutantes resistentes. Estes mutantes se caracterizam pelo fato de precisarem de dados específicos para ser eliminados.

3. Desenvolvimento

3.1. Mutantes Resistentes Existe uma percepção comum aos praticantes de teste de mutantes de que mesmo o conjunto de dados de testes mais trivial, pequeno e sem criatividade é capaz de eliminar a maior parte dos mutantes [Harman, Jia e Langdon 2010]. Esta percepção forma a base para os experimentos descritos nas sessões seguintes. Durante o processo de execução de testes de mutantes, a maior parte dos mutantes será eliminada facilmente com os primeiros testes realizados. Restará então um grupo de mutantes formados por mutantes equivalentes (aqueles que sempre produzirão os mesmos resultados que o programa original) e por mutantes de difícil eliminação, que somente irão morrer com dados não triviais criados especialmente para eles. Este grupo especial de mutantes foi denominado mutantes resistentes. Definindo mais formalmente, os mutantes resistentes são mutantes não-equivalentes de um determinado programa que necessitam de dados específicos não-triviais para serem eliminados. 3.2. Experimentos com Mutantes Resistentes Esta Seção apresenta e analisa dois experimentos que exploram a análise de mutantes, com o objetivo de demonstrar a existência e avaliar a eficácia na detecção de defeitos dos mutantes resistentes. Durante os experimentos, cinco classes escritas em Java foram submetidas a baterias de testes de aproximadamente dez horas, utilizando dados de teste gerados aleatoriamente. Foram coletados os dados de eliminação/sobrevivência dos mutantes em relação a cada dado utilizado, para que se pudesse analisar a dependência dos mutantes resistentes com relação aos dados de teste utilizados. 3.3. Preparação para os Experimentos Para cada uma das classes selecionadas para os experimentos, foi implementada uma classe de teste, contendo o método de geração de dados aleatórios e o método de caso de teste. O método de caso de teste executa (direta ou indiretamente) todos os métodos da classe para os quais existem mutantes e a invocação dos métodos públicos necessariamente utilizou parâmetros aleatórios (quando estes existiam), gerados através da chamada do método de geração de dados aleatórios. Para todos os experimentos realizados, o µJava (sistema de mutantes para programas escritos em Java) foi responsável por gerar o conjunto de mutantes dos programas utilizados e executar casos de teste [Ma, Offutt e Kwon 2005].

4. Resultado e Discussão

4.1. Experimento 1 – Existência Objetivos O objetivo deste primeiro experimento foi validar a percepção de que grande parte dos mutantes pode ser eliminada com poucos dados de testes, e que apenas uma pequena parcela dos mutantes precisa de casos de testes específicos para serem mortos. O objetivo, portanto, é comprovar a existência dos mutantes resistentes. Procedimento Cada classe e seus respectivos mutantes foram submetidos a uma bateria de teste utilizando dados aleatórios com duração aproximada de 10 horas. Esta duração foi escolhida imaginando-se que em uma abordagem prática, esta análise pudesse ser feita durante o período noturno, quando os recursos computacionais não estivessem em uso pelo desenvolvedor. Assim, o número total de conjuntos de dados aleatórios varia de classe para a classe. Os escores de mutação obtidos variaram de 69,85% a 99,28%. Análise Para cada um dos mutantes gerados para as classes em teste, foi calculada a sua taxa de sobrevivência (TS), que é a razão entre o número de vezes que o mutante sobreviveu pelo número de vezes que ele foi executado. Após o cálculo das TS dos mutantes, foi montado um histograma como o exibido na Figura 1, para possibilitar a análise da distribuição da TS. O histograma exibe o total de mutantes cuja TS está em uma das faixas determinadas (de 0% a 20%, de 20,01% a 40%, de 40,01% a 60%, de 60,01% a 80%, de 80,01% a 90%, de 90,01% a 95%, de 95,01% a 99,99%, acima de 100%), bem como o total de mutantes equivalentes. A partir da análise dos histogramas de distribuição das TS dos mutantes, determinou-se para cada classe a taxa de sobrevivência de corte (TSC). Somente os mutantes com TS maior que a TSC são considerados resistentes. Discussão dos Resultados: A partir dos histogramas montados para cada uma das classes, observou-se que realmente existe um grupo de mutantes que são mais difíceis de serem eliminados que a maioria. A Tabela 1 apresenta um resumo dos resultados obtidos no experimento 1. Para todas as classes analisadas são exibidos o número de mutantes gerados para a classe, a TSC, o número total de mutantes resistentes e o percentual que eles representam em relação ao número total A análise das classes indica que os mutantes resistentes são um grupo restrito de mutantes, cuja quantidade média é de 20,09% do número total gerado. Este número médio de 20,09% pode ser utilizado em uma abordagem alternativa para se determinar os mutantes resistentes: ao invés de se observar o histograma e determinar uma TSC, pode se considerar como resistentes os 20% de mutantes com as maiores TS. Outro ponto de atenção deste experimento é que 49% dos mutantes resistentes necessitam de dados específicos para serem eliminados. Uma possível melhoria deste experimento seria produzir classes de teste que produzem escores de mutação maiores. Portanto, é necessário identificar pontos novos critérios de escrita das classes de teste. 4.2. Experimento 2 - Eficácia Objetivos O objetivo deste experimento foi verificar se os dados que eliminam os mutantes resistentes são eficazes na detecção de defeitos em programas, através da medida do escore de mutação produzidos por estes dados. Procedimento Cada classe e seus respectivos mutantes foram submetidos a uma bateria de testes, com um conjunto de dados de teste gerados aleatoriamente. Foram calculados, então os escores de mutação para duas situações distintas: primeiro, considerando-se todos os conjuntos de dados de teste gerados aleatoriamente; depois, considerando-se apenas os parâmetros aleatórios que eliminaram os mutantes resistentes. Análise A Tabela 2 apresenta os escores de mutação obtidos para duas situações distintas: primeiro, considerando-se todos os conjuntos de dados de teste gerados aleatoriamente (definidos na Tabela 2 como Cenário 1); depois, considerando-se apenas os parâmetros aleatórios que eliminaram os mutantes resistentes (definidos na Tabela 2 como Cenário 2). Discussão dos Resultados: O escore de mutação é uma indicação do quão bem os testes testam uma classe ou programa. Portanto, podemos medir a eficácia dos dados que eliminam

mutantes resistentes através da comparação dos escores de mutação obtidos através da utilização destes dados ou dos outros dados gerados aleatoriamente. Não houve variação dos escores obtidos, portanto os dados que eliminam mutantes resistentes substituem completamente o conjunto total de parâmetros utilizados. Um estudo futuro poderia avaliar se existem dados redundantes, cujos resultados são iguais para todos os mutantes. Estes dados redundantes poderiam ser eliminados, com o objetivo de se reduzir o total de dados de teste utilizados.

5. Considerações Finais

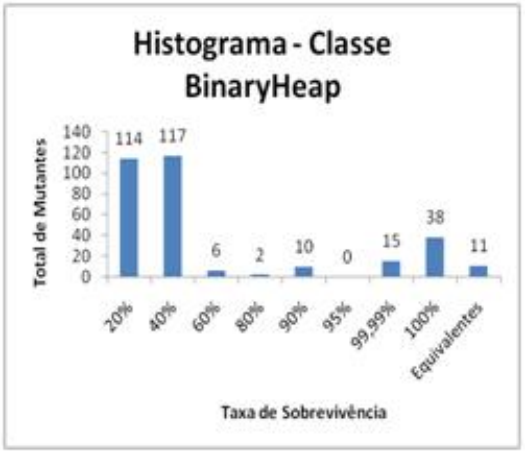
Os experimentos realizados evidenciam a existência de mutantes que são mais difíceis de serem eliminados do que a grande maioria. Estes mutantes foram denominados mutantes resistentes, e representam em média 20% do total de mutantes gerados para uma determinada classe. A principal característica dos mutantes resistentes é que eles necessitam de dados específicos para serem eliminados, dado um cenário de testes em que todos os métodos públicos são exercitados ao menos 1 vez, e os parâmetros recebidos por estes métodos são gerados aleatoriamente. Se considerarmos somente os dados que eliminam mutantes resistentes, os escores de mutação obtidos foram iguais aos obtidos na bateria completa de 10 horas de execução, realizada no primeiro experimento. A identificação e caracterização dos mutantes resistentes podem ser úteis na definição de uma estratégia de redução do custo humano no teste de software. Trabalhos futuros deverão avaliar uma abordagem prática, onde a análise de mutantes e os mutantes resistentes são utilizados não para se determinar o grau de adequação dos testes realizados, mas sim para selecionar dados de teste adequados que posteriormente serão utilizados para se testar o programa ou classe original. Para que isso seja possível, são necessários estudos adicionais que determinem novos critérios para a escrita das classes de teste, com o objetivo de produzir escores de multação maiores. É necessário também analisar se existem dados redundantes entre os dados que eliminam mutantes resistentes, pois estes dados poderiam ser desconsiderados, diminuindo-se o total de dados de teste selecionados. Outro ponto a se explorar é se estes dados específicos que eliminam mutantes resistentes têm alguma relação com os dados de teste gerados através de técnicas de teste conhecidas, como por exemplo, análise de valor ou limite particionamento em classe de equivalência [Pressman 2006].

Referências Bibliográficas

DeMillo, R. A., Lipton, R. J. e Sayward, F. G. (1978), “Hints on Test Data Selection: Help for the Practicing Programmer”, Computer, vol. 11, nº. 4, pp. 34–41. Harman, M., Jia, Y. e Landgdon, W. B. (2010), “A Manifesto for Higher Order Mutation Testing”, icstw, pp.80-89, 2010 Third International Conference on Software Testing, Verification, and Validation Workshops. Jia, Y. e Harman, M. (2009), “An Analysis and Survey of the Development of Mutation Testing”, CREST King’s College London, Technical Report TR-09-06 Ma, Yu-Seung; Kwon, Yong-Rae; Offutt, A. J. (2005), “Description of Method-level Mutation Operators for Java”. Disponível em: http://cs.gmu.edu/~offutt/mujava/. Acesso em: 09 de outubro de 2010. Ma, Yu-Seung; Offutt, A. Jefferson (2002), “Inter-Class Mutation Operators for Java” In: International Symposium on Software Reliability Engineering, 13. Proceedings of the 13th International Symposium on Software Reliability Engineering, Annapolis, MD, USA. IEEE Computer Society Press, p. 352-363. Offutt, A. J. (1995) “A Practical System for Mutation Testing: Help for the Common Programmer”, Proc. 12th Int’l Conf. Testing Computer Software (ICST 95), IEEE CS Press, pp. 99–109. Pressman, R. S. (2006) Engenharia de Software. 6ª edição. McGraw-Hill.

Anexos

Classe	TSC	# de Mutantes	# de Mutantes Resistentes	% de Mutantes Resistentes
BinaryHeap	80%	313	63	20,13%
BinarySearchTree	60%	208	30	14,42%
Calendar	80%	462	105	22,73%
DoubleLinkedList	80%	226	95	42,04%
Sort	40%	881	10	1,14%
Média				**20,09%**



	Cenário 1		Cenário 2		
Classes	**# de Dados de Teste**	**Escore de Mutação**	**# de Dados de Teste**	**Escore de Mutação**	**Variação entre os escores**
BinaryHeap	2000	87,42%	696	87,42%	0,00%
BinarySearchTree	4500	86,19%	1822	86,19%	0,00%
Calendar	2000	84,47%	1647	84,47%	0,00%
DoubleLinkedList	4000	69,85%	664	69,85%	0,00%
Sort	840	99,28%	665	99,28%	0,00%