Estratégia de Redução do Custo da Atividade de Teste

Autores

Plinio Roberto Souza Vilela
Waldo Luis de Lucca

Apoio Financeiro

Fap

1. Introdução

O grande volume de software que é produzido na atualidade chama a atenção para a necessidade de se aprimorar as técnicas que podem contribuir para a melhoria da qualidade, tanto do produto como do processo. Nesta direção, muitas metodologias e técnicas têm sido desenvolvidas ao longo dos anos, dentre as quais estão algumas técnicas de teste estrutural e funcional.

Muitas das técnicas desenvolvidas se tornaram clássicas, mas as que mais são utilizadas são aquelas voltadas para a realização das atividades de análise de requisitos, especificação de sistemas e especificação de projeto.

Porém, no contexto da qualidade de software, existem atividades fundamentais, como verificação, validação e teste de software, cujas técnicas disponíveis nem sempre são tão aplicadas como as clássicas de análise e projeto, embora sejam tão clássicas quanto estas. Uma hipótese é que estas atividades consomem recursos e tempo em demasia e, por este motivo, muitos desenvolvedores preferem abreviá-la para cumprir o orçamento ou o cronograma. O problema desta abreviação é a entrega de software com mais defeitos para seus usuários.

O objetivo do teste é revelar a presença de defeitos no software (MYERS, 1979). Sua premissa é de que o software possui defeitos que estão escondidos e eles precisam ser descobertos para que o software possa, assim, ser entregue a seu usuário. Assim, há a necessidade de se buscar conhecer ainda mais os atributos de software que são necessários para a realização de um teste com maior eficácia.

Considerando que a eficácia do teste de software é um fator de grande importância para se garantir a qualidade de um software e que o custo desta atividade é um fator limitador de sua aplicação efetiva, o estudo da relação entre a eficácia e o custo pode revelar informações que contribuam para a formulação de técnicas de teste que tenham a máxima eficácia para um reduzido custo.

2. Objetivos

O teste de software é considerado, em clássicos da literatura em Engenharia de Software, como uma atividade que consome entre 30 e 50% do esforço total de desenvolvimento (PRESSMAN, 2000; SOMMERVILLE, 2004). Porém, o teste de software inclui uma série de atividades, em diferentes níveis, sendo difícil de mensurar o impacto de cada uma delas no custo total do desenvolvimento e manutenção do software.

O custo do teste de software deve levar em conta o nível em que é realizado (teste de unidade, teste de integração, teste de sistema, etc.), as atividades que compõe o processo de teste, os critérios aplicados, além de outros fatores como hardware, ferramentas, engenheiros e testadores.

Se o número de defeitos de um software fosse conhecido antes de se iniciar o teste, esta atividade seria direcionada a encontrá-los. Porém, definir com certeza o número de defeitos de um software é impossível, embora possa ser estimado. Várias pesquisas têm buscado desenvolver técnicas para predição do número de defeitos em um sistema (GAFFNEY, 1984; LIPOW, 1982; MOELLER e PAULISH, 1993; WONG et al., 1994; WONG et al., 1998). Estas técnicas são partes de modelos de predição de defeitos. Métricas de tamanho e de complexidade de software são utilizadas neste modelos, associando-se o número de defeitos encontrados durante o desenvolvimento de um software a estas medidas.

O objetivo deste trabalho é contribuir para o estudo sobre a capacidade de métricas de tamanho de software auxiliarem na predição do número de defeitos de um software. Esta contribuição deve ser considerada na formulação de estratégias de redução de custos do teste de software. O trabalho apresenta um estudo realizado por meio de um experimento com um software com defeitos reais conhecidos, buscando-se identificar, em nível de módulo, a relação entre o número de defeitos e o tamanho. Este estudo complementa outro experimento realizado no âmbito do mesmo projeto (VILELA et al., 2004), que utilizou um conjunto pequeno de métricas de tamanho e complexidade, coletado a partir de um nível menor de granularidade, ou seja, em nível de segmentos de código que implementam requisitos de teste.

3. Desenvolvimento

O presente trabalho consistiu na realização de um experimento, utilizando um programa com defeitos reais documentados. O software selecionado para o experimento foi um programa desenvolvido pela Agência Espacial Européia (*European Space Agency* - ESA), chamado Space, desenvolvido em linguagem C. Este programa provê uma interface que permite o usuário descrever a configuração de um conjunto de antenas utilizando uma linguagem de alto nível (CANCELLIERI e GIORGI, 1994). A escolha deste programa se deu pelo fato de ter defeitos reais documentados, além de já ter sido usado em outros experimentos.

Os objetos da pesquisa foram os módulos do programa. O Space é um programa com 4296 linhas de código (*Lines of Code* - LOC), divididos em 135 módulos. Ao todo, estão documentados 34 defeitos, presentes em 25 módulos (correspondente a 18,5% do total de módulos). A densidade de defeitos (número de defeitos a cada mil linhas de código) é 7,734.

As métricas escolhidas para o experimento foram linhas de código (LOC), número de defeitos por módulo, densidade de defeitos e o número de mutantes. Mutantes são programas derivados do programa em estudo, contendo uma única alteração sintática em relação ao mesmo, baseado em operadores que simulam os defeitos mais comuns (DEMILLO, 1978).

O objetivo do experimento foi analisar se há maior concentração de defeitos em módulos maiores, partindo da hipótese de que um módulo maior tem maior probabilidade de conter defeitos.

O método utilizado no experimento foi composto das seguintes etapas:

1. Coleta das métricas de cada módulo (LOC, mutantes, defeitos e densidade de defeitos).

2. Análise da correlação entre as métricas de tamanho (LOC e número de mutantes) e os defeitos e densidades de defeitos por módulo, através do cálculo da dependência linear entre cada métrica e o número ou densidade de defeitos: Foi calculado o coeficiente de correlação de cada par (X,Y), onde X é o valor da métrica e Y o número de defeitos.

3. Ordenamento dos módulos por intervalos de acordo com as métricas de tamanho e cálculo da correlação entre as métricas e o número de defeitos em cada intervalo.

4. Resultados

Uma análise inicial pode ser feita em relação à dependência linear entre as métricas LOC e número de defeitos. O coeficiente de correlação é de -0,17977, o que demonstra não haver relação entre o número de defeitos e o tamanho do módulo. Esta dependência muda radicalmente se considerarmos a densidade de defeitos do módulo, em vez do número de defeitos. Neste caso o coeficiente de correlação passa para -0,64498, ou seja, quase 70% de correlação inversamente proporcional.

Em relação ao número de mutantes, o coeficiente de correlação com o número de defeitos é -0,15366, este número passa para -0,33441 quando a dependência analisada é entre número de mutantes e densidade de defeitos. Considerar a relação com densidade de defeitos, em vez de número de defeitos, com o número de mutantes não produz a mesma variação do coeficiente de correlação que houve na análise em relação ao número de linhas de código.

A partir desta análise inicial, no âmbito deste experimento, a métrica LOC passou a ser analisada em mais detalhes, buscando-se avaliar sua utilidade como indicador do número de defeitos para, a partir daí, buscar estratégias de teste em que os casos de teste sejam gerados inicialmente para os módulos com maior número de linhas de código.

O número de linhas de código dos módulos do Space variam de 1 a 144 LOC em cada módulo, tendo como média 32,5 LOC. Se os módulos forem divididos em dois conjuntos, sendo o primeiro dos módulos com LOC menor que a média e o segundo, dos módulos com LOC maior que a média, temos a seguinte distribuição: 87 módulos (64,4%) são menores que a média e 35 módulos (35,6%) são maiores que a média.

Os 34 defeitos do Space estão, em temos de dados brutos, divididos quase eqüitativamente entre os módulos maiores e menores. 18 defeitos (52,9%) são encontrados nos módulos com LOC acima da média, enquanto 16 defeitos (47,1%) estão nos módulos com tamanho abaixo da média.

Porém, como o número de módulos de cada conjunto varia significativamente, é necessário observar em quais módulos os defeitos se encontram. 60% dos módulos com defeito estão entre os módulos com LOC acima da média. Os 16 defeitos dos módulos com LOC abaixo da média estão concentrados em apenas 10 módulos (11,5%) do total de módulos desta partição. Já os defeitos encontrados nos módulos com LOC acima da média estão presentes em 15 módulos, representando 31,3% do total de módulos desta partição. Assim, é possível observar que, dentre os módulos maiores, há uma maior quantidade (quase o triplo) de módulos com defeitos do que dentre os módulos menores.

Outra análise realizada no experimento foi a distribuição dos defeitos pelos módulos ordenados por tamanho, particionado em 5 blocos, de 27 módulos cada um. O número de defeitos e de linhas de código em cada bloco é apresentado na Tabela 1.

Os dados da Tabela 1 mostram que os menores módulos do Space (bloco 1, incluindo os 20% de módulos que são os menores) não tem defeitos. O tamanho desses módulos variam de 1 a 11 LOC. Já no segundo bloco, cujos módulos têm tamanho entre 11 e 18 LOC, começam a aparecer os defeitos. Se analisarmos a correlação entre o percentual de LOC e o percentual de defeitos de cada bloco, o coeficiente é de 0,53894. Mas quando consideramos o percentual acumulado, este coeficiente sobe para 0,94734, demonstrando haver uma forte dependência entre o percentual acumulado de defeitos e o percentual acumulado de LOC dos módulos. Esta análise permite observar que se a atividade de teste iniciar com os módulos maiores, há uma tendência de que todos os defeitos sejam encontrados mais rapidamente do que utilizar uma ordem

aleatória de execução de testes nos módulos.

5. Considerações Finais

Predizer o número de defeitos em um software é um problema com grau elevado de dificuldade. O presente trabalho buscou contribuir para o conhecimento empírico acerca da adequação de métricas de tamanho para predizer o número de defeitos de software, por meio de um experimento em nível de granularidade de módulo.

Pelos resultados obtidos, constata-se que há uma tendência dos defeitos de um software estarem, em sua maioria, nos módulos maiores. Também pôde ser observado que a totalidade dos defeitos pode ser identificada antes se a atividade de teste for realizada em uma ordem decrescente de tamanho (LOC) em relação ao teste dos módulos em ordem aleatória.

Novos experimentos devem ser realizados, com outros programas, buscando confirmar as evidências percebidas a partir do trabalho realizado. A partir destes resultados, pode-se desenvolver técnicas e estratégias de teste visando à redução do custo desta atividade e aumento de sua eficácia, contribuindo assim para a melhoria da qualidade de software.

Referências Bibliográficas

CANCELLIERI A; GIORGI, A. Array PreProcessor User Manual, Technical Report IDS-RT94/052, 1994.

GAFFNEY, J. R. Estimating the Number of Faults in Code. IEEE Transactions on Software Engineering, v. 25, n. 5, p. 675-689, setembro/outubro 1999.

DEMILLO, R. A.; LIPTON, R. J.; SAYWARD, F. G. Hints on Test Data Selection: Help for the Practicing Programmer. IEEE Computer, vol. 11, n. 4, abril de 1978, p. 34-41.

LIPOW, M. Number of Faults per Line of Code. IEEE Transactions on Software Engineering, v. 10, n. 4, julho/agosto 1982.

McCABE, T. J. A Complexity Measure. IEEE Transactions on Sfotware Engineering, v. 2, n. 4, p. 308-320, dezembro 1976.

MOELLER, K. H.; PAULISH, D. An Empirical Investigation of Software Fault Distribution. In: FIRST INTERNATIONAL SOFTWARE METRICS SYMPOSIUM, p. 82-90, 1993.

MYERS, G. The Art of Software Testing. Willey, 1979.

PRESSMAN, R. S. Software Engineering: A Practitioner\'s Aproach. 5th Edition. McGraw Hill, 2000.

SOMMERVILLE, I. Software Engineering. Addison Wesley, 2004.

VILELA, P. R. S.; LUCCA, W. L.; CORSO, A.; JINO, M.. Comparison of Size and Complexity Metrics as Predictors of the Number of Faults. In: IV JORNADAS IBEROAMERICANAS EN INGENIERÍA DEL SOFTWARE E INGENIERÍA DEL CONOCIMENTO- JIISIC\'04, 2004, Madrid - Espanha, v. I, p. 633-644, 2004.

WONG, E.; HORGAN, J. R.; LONDON, S.; MATHUR. Effect of Test Set Minimization on the Fault Detection

Effectiveness of teh All-Uses Criterion. Relatório Técnico, Purdue University, 1994.

WONG, E.; HORGAN, J. R.; LONDON, S.; MATHUR, A. Effect of Test Set Minimization on Fault Detection Effectiveness. Software – Practice and Experience, v. 28, n. 4, p. 347-369, abril 1998.

Anexos

Tabela 1 - Distribuição de defeitos por blocos de módulos

	LOC				Defeitos			
Blocos	#	%	# acum.	% acum.	#	%	# acum.	% acum.
Até 20% menores	143	3,3%	143	3,3%	0	0,0%	0	0,0%
20 a 40%	390	8,9%	533	12,1%	11	32,4%	11	32,4%
40 a 60%	625	14,2%	1158	26,3%	5	14,7%	16	47,1%
60 a 80%	981	22,3%	2139	48,7%	8	23,5%	24	70,6%
80 a 100%	2257	51,3%	4396	100,0%	10	29,4%	34	100,0%