5ª MOSTRA ACADÊMICA UNIMEP
23 a 25 de outubro de 2007
UNIMEP
Universidade Metodista de Piracicaba
Tema:
EDUCAÇÃO BRASILEIRA:
Extinção ou Sustentabilidade na Universidade

5º Congresso de Pós-Graduação

DEFEITOS SEMANTICAMENTE PEQUENOS SÃO DIFÍCEIS DE DETECTAR

Autor(es)

GERSON NUNHO CARRIEL

Co-Autor(es)

JOSÉ CARLOS PERINI
PLÍNIO ROBERTO SOUZA VILELA

Orientador(es)

PLÍNIO ROBERTO SOUZA VILELA

1. Introdução

MYERS (1979) define que o teste de software é a atividade que realiza validação e verificação em partes ou no software todo desenvolvido, e tem a função de encontrar defeitos. MYERS (1979) ainda afirma que um bom teste é aquele que encontra defeitos. Um defeito em software é a deficiência mecânica ou algorítmica que se ativada pode levar a uma falha. SOMMERVILLE (2003) indica que a finalidade dos testes de detecção de defeitos é expor esses defeitos antes que o software seja entregue. Na busca por melhorar a busca por defeitos antes de um software ser entregue, OFFUTT (1996) indica que um defeito pode ser visto sintaticamente quando é verificado o que é necessário para corrigi-lo, quais linhas de código são necessárias a alteração para sua correção. Assim, se muitas linhas são necessárias para sua correção, sabemos que é um defeito sintaticamente grande, caso seja necessária uma pequena mudança, esse defeito será sintaticamente pequeno. O mesmo vale para o processo de inserir(alimentar) defeitos no código, quando existe a necessidade de alterar muita coisa para gerar o defeito, este será caracterizado como defeito sintaticamente grande, e quando a alteração se deseja inserir um defeito e a mudança é pequena, temos a caracterização de um defeito sintaticamente pequeno. A caracterização de um defeito semântico, está relacionado a saída esperada de uma aplicação em relação a saída esperada em sua especificação, qualquer valor diferente de saída gerado, qualquer valor não esperado pela especificação, está relacionado a um defeito, e essa é a caracterização de um defeito semântico. E como OFFUTT (1996) afirma, essa caracterização faz é dar diferentes formas de analisar um defeito. Mas essa caracterização apresenta alguns aspectos interessantes do tamanho do defeito. Embora um defeito sintaticamente pequeno, possa gerar tanto resultados pequenos diferentes do esperado na especificação, um defeito sintaticamente pequeno pode também apresentar resultados com grandes diferenças do especificado, logo percebemos que o tamanho sintático, não faz grande diferença em relação aos resultados e não existe relação entre os

resultados semânticos pequenos ou grandes, em comparação aos defeitos sintáticos pequenos e grandes. OFFUTT (2002) Sugere que o tamanho semântico é mais importante que o tamanho sintático no momento de realizar mudanças para análise de defeitos, como é comum em experimentos de alimentação de defeitos para validação de testes. Na ocasião onde a diferença entre o resultado esperado, segundo a especificação, e o resultado efetivo apresenta um tamanho grande, segundo OFFUTT (1996) os processos de detecção de defeitos obtém mais sucesso, ou pode-se dizer que são mais fáceis às detecções de defeitos em situações onde a caracterização do defeito semântico é grande. Em OFFUTT (2002) é reafirmado que se for inserido um defeito semanticamente grande, o defeito será fácil de detectar através de testes. Essa afirmação de que um defeito semanticamente grande é de fácil detecção e que um semanticamente pequeno é de difícil detecção motivou a realização deste experimento para avaliação dessa afirmativa.

2. Objetivos

O objetivo é comprovar ou refutar se a afirmação sugerida por OFFUT (1996) de que um defeito semanticamente pequeno é de difícil detecção. Assim o trabalho aqui proposto irá gerar subsídios teóricos para que estudos futuros possam dar continuidade ao tema.

3. Desenvolvimento

– Metodologia A metodologia utilizada para realização deste experimento, é a metodologia sugerida por KITCHENHAM (2002). Onde foi realizado o planejamento do projeto do experimento, do acompanhamento de sua execução, análise de seus resultados e apresentação. – Projeto do Experimento O experimento foi realizado, tendo como software para realização dos testes de uma aplicação chamada cal.c. A função dessa aplicação é retornar um calendário dependendo dos parâmetros passados (ou também quando nenhum parâmetro é passado). O cal.c, quando não recebe nenhum parâmetro, calc.c retorna o calendário do mês corrente, quando seguido de um número, irá apresentar um calendário com todos os meses do ano, representado pelo número passado, e também podem ser passados dois parâmetros, dois números, que representam respectivamente mês e ano, nesse caso será apresentado o calendário do mês e ano passados como parâmetro. As seguintes atividades foram realizadas: a) Definição de fácil e difícil detecção. Inicialmente, definimos o que seria fácil e difícil na detecção de defeitos, como pode ser visto abaixo: qtd >=50 : FACIL DETEÇÃO qtd >=40 e qtd <50 : MÉDIA DETEÇÃO qtd >=20 e qtd <40 : DIFICIL DETEÇÃO qtd < 20 : MUITO DIFICIL DETEÇÃO qtd representa a quantidade de casos de testes que encontraram defeitos b) Definição dos defeitos Para realização dos testes, a aplicação recebeu dois defeitos. O primeiro defeito é semanticamente grande, assim escolhemos um defeito sintaticamente pequeno, mas que nos resultasse uma quantidade grande de falhas, como pode ser visto: Linha correta (linha 119): mon[2] = 29; Linha após a inserção de defeito: mon[3] = 29; O segundo defeito era sintaticamente maior, porém era semanticamente pequeno: Foram retirados dois trechos entre a linha 38 e a linha 47 do código. Foram retiradas duas situações de condições para validação do mês de entrada e do ano de entrada para apresentação do calendário em tela: Trecho correto: m = atoi(argv[1]); if(m<1 || m>12) { fprintf(stderr, cal: %s: Mês errado. , argv[1]); exit(1); } y = atoi(argv[2]); if(y<1 || y>9999) { fprintf(stderr, cal: %s: Ano errado. , argv[2]); exit(2); } Trecho após inserção de defeito(defeito inserido foi a falta de algumas linhas): m = atoi(argv[1]); y = atoi(argv[2]); Observar que as verificações para mês errado e ano errado foram ignoradas. c) Gerar dados aleatórios. Foi criada uma planilha de cálculo para gerar os números aleatórios que seriam utilizados como casos de teste. Para realização dos testes foram criados 100 casos de testes através da geração de meses e anos aleatoriamente. Foi necessário analisar o conjunto de dados que a especificação definia como dados corretos para entrada e a planilha extrapolou um pouco o limite inferior e superior dos meses e dos anos permitidos. Na especificação somente poderiam existir meses entre 1 e 12, para o experimento, a geração de meses aleatórios teve seu intervalo aumentado de -1 para 13. Para o intervalo especificado, entre ano 1 e ano 9999, o intervalo foi alterado para gerar anos entre -1000 e 11000. Na geração aleatória, quando recebido 0 no parâmetro, foi trocado por espaço em branco, para possibilitar a passagem de apenas um parâmetro na linha de execução ou nenhum, caso ambos fossem zero. d) Realização dos testes Foram realizados os testes com os casos de teses gerados e seus resultados foram

lançados em uma nova planilha para controle e análise dos resultados. Ao final, levantou-se o número de casos de testes que passaram sem defeito e o número de casos de testes que encontraram defeitos. – Condução e coleta dos dados do experimento No dia da realização do experimento foram gerados dois aplicativos novos. O primeiro foi alimentado com um defeito que geraria uma quantidade pequena de falhas nos resultados (semanticamente pequeno), e foi gravado no disco como cal-22.exe. O segundo alimentado com outro defeito, um que geraria uma quantidade grande de falhas nos resultados (semanticamente grande), e gravado no disco como cal-10.exe. Para facilitar a geração de dados aleatórios, foi usada uma planilha de calculo, onde seriam gerados dados acima da faixa aceita pela aplicação e abaixo da faixa de dados aceita pela aplicação Abaixo um exemplo da execução dos aplicativos através de dos dados gerados aleatoriamente: Os Dados gerados: -1 1250 0 1900 12 2500 Para semanticamente grande cal-10 -1 1250 cal-10 1900 cal-10 12 2500 Para semanticamente pequeno cal-22 -1 1250 cal-22 1900 cal-22 12 2500 Observação: Os mesmos dados gerados aleatoriamente são utilizados para as duas aplicações e seus resultados lançados em uma planilha eletrônica. – Análise A análise foi realizada através da quantidade de casos de testes que encontraram defeitos. Para validar o resultado deste experimento utilizamos as faixas de valores definidos para FÁCIL, MÉDIA, DIFICIL E MUITO DIFICIL detecção de defeitos.

4. Resultados

Para a aplicação que foi alimentada com um defeito que geraria uma falha semanticamente grande, temos: Casos de testes que encontraram defeitos: 67 Caso de testes que não encontraram defeitos: 33 Para a aplicação que foi alimentada com um defeito que geraria uma falha semanticamente pequena, temos: Casos de testes que encontraram defeitos: 17 Caso de testes que não encontraram defeitos: 83 Os resultados mostraram que os defeitos que geram falhas semanticamente grandes são MUITO FÁCEIS de serem encontrados, num universo de 100 casos de testes, 67% dos casos de teste apontam falhas. Por outro lado, temos 17% apenas dos casos de testes encontrando o defeito que gerou a falha semanticamente pequena. Podemos afirmar que é MUITO DIFÍCIL encontrar defeitos semanticamente pequenos e que defeitos que geram falhas semanticamente grandes são de FÁCIL detecção.

5. Considerações Finais

O trabalho aqui realizado valida apenas a relação apresentada por OFFUT (1996), mas uma lacuna ainda existe, uma técnica que auxilie o profissional de teste de software a criar casos de testes voltados a encontrar defeitos semanticamente pequenos.

Referências Bibliográficas

KITCHENHAM, Barbara A. et all. Preliminary Guidelines for Empirical Research in Software Engineering. IEEE Transactions on Software Engineering. Vol. 28 no. 8. Agosto de 2002.

MYERS, Glenford J. The Art of Software Testing, Editora John Wiley & Sons, New Jersey, USA, 1979.

SOMMERVILLE, Ian. Engenharia de Software, 6 ed., Editora Pearson Addison Wesley: São Paulo, Brasil, 2003.

OFFUTT, J. and Haues, J. Huffman. A semantic model of program faults. The Proceedins of International Symposium on Software Testing and Analysis. ACM, San Diego, California. Janeiro 1996, 195-200

OFFUTT, J. And Hayes, J. Huffaman. Applying a Semantic Fault Model to the Empirical Study of Corrective Maintenance. 8th IEEE Workshop on Empirical Studies of Software Maintenance, Montreal, Canada, Outubro de 2002.