Um Esquema de Injeção de Defeitos Baseado em Operadores de Mutação

Elisa Yumi Nakagawa
José Carlos Maldonado

ICMSC – USP
Av. Dr. Carlos Botelho, 1465
C. Postal· 668, CEP: 13560-970
São Carlos - SP, Brasil
{elisa, jcmaldon}@icmsc.sc.usp.br

Resumo

A Injeção de Defeitos (*Fault Injection*) é uma técnica que vem sendo utilizada para a construção de sistemas que precisam ser altamente confiáveis. Para a sistematização dessa técnica, a utilização de taxonomias ou modelos de defeitos tem um papel fundamental. Neste artigo é proposto um esquema de injeção de defeitos baseado nos operadores de mutação do critério de teste Análise de Mutantes.

Abstract

Fault Injection has been used for high dependable system development. The use of a taxonomy or a fault model is fundamental for making it a systematic activity. In this article, a fault injection scheme based on mutation operators from Mutation Analysis testing criterion is presented.

1. Introdução

Na área de Tolerância a Defeitos (*Fault Tolerance*), a Injeção de Defeitos (*Fault Injection*) é uma técnica emergente que tem sido empregada para a construção de sistemas que precisam ser altamente confiáveis [CLA95]. De um modo geral, essa técnica envolve a injeção controlada de defeitos e a observação do aspecto comportamental do sistema frente à presença dos defeitos.

Para auxiliar na sistematização da atividade de Injeção de Defeitos, tornando-a mais objetiva e eficaz, são requeridos taxonomias ou modelos de defeitos que representem os defeitos a serem injetados. Assim, dentro desse contexto, o estudo e o estabelecimento de taxonomias ou modelos de defeitos tornam-se bastante relevante.

Dentre os trabalhos sobre Injeção de Defeitos, existem estudos que englobam defeitos de software e de hardware. Observa-se que existem poucos trabalhos relacionados à injeção de defeitos de software na literatura [SOM97]. Segundo Martins [MAR93] isso ocorre devido à falta de modelos de defeitos representativos e amplamente aceitos, além da existência de um número limitado de mecanismos visando à tolerância a defeitos de software. Além da falta de modelos de defeitos de software, não existe na literatura um consenso sobre o que seria, exatamente, um modelo de defeitos de software e quais itens — tipo, freqüência, criticalidade dos defeitos, entre outros — deveriam estar presentes no modelo.

Observa-se que existem diversos trabalhos na literatura, como o de Basili/Perricone [BAS84], de Beizer [BEI90], de DeMillo/Mathur [DEM95], entre outros, que visam à classificação dos defeitos encontrados nas diversas fases do ciclo de vida de desenvolvimento do software. Essa diversidade deve-se ao fato da classificação dos defeitos ser uma tarefa difícil, pois os defeitos podem ser classificados considerando-se vários fatores: o comportamento do programador, o efeito causado pelo defeito no programa, entre outros.

Esses estudos enfatizam a relevância do estabelecimento de modelos de defeitos de software concretos e que representem os defeitos "reais" que ocorrem em um sistema de software. O estudo sobre modelos de defeitos de software é ainda um ponto em aberto do ponto de vista da pesquisa científica, diferente dos modelos de defeitos de hardware, sobre os quais pode-se identificar diversos trabalhos.

Atualmente, a injeção de defeitos de software mostra-se relevante por questões de qualidade e produtividade, por isso um dos objetivos deste trabalho é abordar a injeção de defeitos em sistemas de software, mais especificamente no programa fonte do sistema. Assim, na Seção 2 é apresentado o esquema de injeção de defeitos de software e na Seção 3, a ferramenta de injeção de defeitos implementada.

2. Um Esquema de Injeção de Defeitos

Neste trabalho, foi estabelecido um esquema de injeção de defeitos de software baseado na taxonomia de defeitos de DeMillo/Mathur [DEM95] e nos operadores de

mutação [AGR89] do critério de teste denominado Análise de Mutantes [BUD81, DEL93].

Na taxonomia de DeMillo/Mathur é definido um total de quatro classes com características bem definidas baseadas em transformações sintáticas. Na Tabela 1 são apresentadas as classes e as correspondentes porcentagens; por exemplo, 53,26% do total de defeitos no código de um programa são pertencentes à classe Defeito de Entidade Simples Incorreta.

Classe	%
Defeito de Entidade Simples Incorreta (*Simple Incorrect Entity Fault*)	53.26
Defeito de Falta de Entidade (*Missing Entity Fault*)	40.55
Defeito de Entidade Espúria (*Spurious Entity Fault*)	0.69
Defeito de Entidade Mal Empregada (*Misplaced Entity Fault*)	5.50

Tabela 1: Classes da Taxonomia de DeMillo/Mathur

Para a definição do esquema de injeção de defeitos foi realizado o mapeamento da taxonomia de defeitos de DeMillo/Mathur para os operadores de mutação do critério de teste Análise de Mutantes. Esses operadores têm a função de realizar transformações sintáticas no código do programa em avaliação. O mapeamento não é uma tarefa trivial, uma vez que muitos dos operadores possuem comportamento bastante variados dependendo do código no qual está sendo aplicado.

O grupo de pesquisa do ICMSC/USP tem desenvolvido trabalhos relacionados aos critérios de teste de software, incluindo o critério Análise de Mutantes. Utilizando-se da experiência adquirida, os operadores de mutação estão sendo utilizados como mecanismo de geração de defeitos (transformações sintáticas no código) de software, uma vez que retratam os defeitos mais comuns cometidos pelo programador ao longo do processo de desenvolvimento de software [DEM78].

Esses operadores foram definidos por Agrawal e totalizam 71 operadores divididos em quatro classes: Mutação de Comandos (15 operadores), Mutação de Operadores (46 operadores), Mutação de Constantes (3 operadores) e Mutação de Variáveis (7 operadores) [AGR89].

Realizado o mapeamento, para a classe Defeito de Entidade Simples Incorreta foram relacionados 59 operadores, para a classe Defeito de Falta de Entidade relaciona-se dois operadores, para a classe Defeito de Entidade Espúria tem-se nove operadores e para a classe Defeito de Entidade Mal Empregada, um operador.

Como exemplo do mapeamento, considere o operador SSDL (*Statement Deletion*), pertencente à classe Mutação de Comandos, que tem a função de apagar comandos no código do programa, retratando defeitos de falta de comandos. Esse operador foi então relacionado à classe Defeito de Falta de Entidade da taxonomia adotada. Um outro exemplo é o operador ORRN (*Operator Relational Relational Non-assignment*), pertencente à classe Mutação de Operador, que troca um operador relacional por outro relacional, representando defeitos de operadores relacionais empregados incorretamente.

Por realizar esse tipo de mudança no código, esse operador foi classificado como Defeito de Entidade Simples Incorreta.

Durante a atividade de injeção de defeitos, a porcentagem relacionada a cada uma das classes é considerada no cálculo do número de defeitos a ser injetado, ou melhor, ocorre uma seleção dos defeitos gerados pelos operadores de modo que o número de defeitos a serem injetados corresponda à porcentagem de cada classe a que pertencem esses operadores.

3. Ferramenta de Injeção de Defeitos

Dada a crescente complexidade dos sistemas computacionais, a atividade de injeção de defeitos é inviável sem a disponibilidade de uma ferramenta que automatize tal atividade; dessa forma, uma ferramenta de injeção de defeitos, denominada ITool, foi implementada. Tal automatização reduzirá os defeitos conseqüentes da intervenção humana, bem como o tempo e os custos necessários se essa atividade fosse realizada manualmente, quando possível.

Essa ferramenta apresenta uma interface gráfica construída em Tcl/Tk [WEL95]. Na Figura 1 é apresentada a janela principal dessa ferramenta e na Figura 2 a janela que permite ao usuário a ativação/desativação de defeitos no processo de injeção de defeitos.

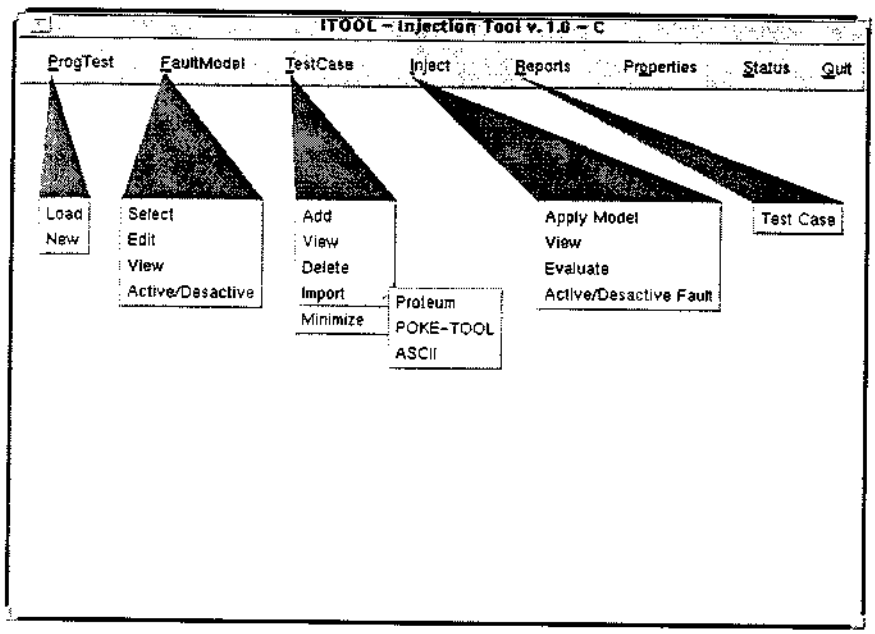

Figura 1: Janela Principal da ITool

A injeção de defeitos gerados pelos operadores de mutação não é uma tarefa trivial, uma vez que ocorrem problemas de priorização de aplicação de defeitos se dois ou mais defeitos relacionam-se a um mesmo ponto de aplicação. Esses e outros problemas são discutidos em mais detalhes em [NAK98].

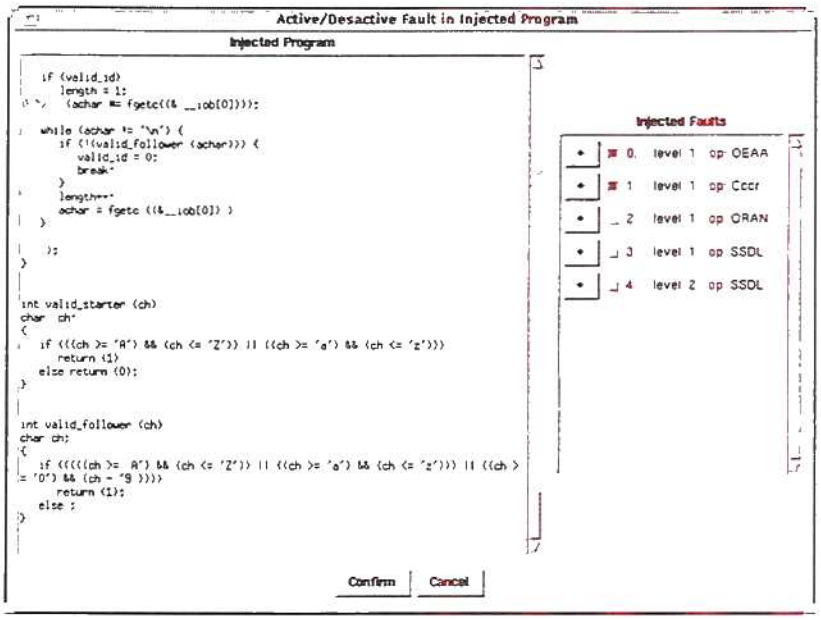

Figura 2: Janela para Ativação/Desativação de Defeitos

De maneira geral, a ferramenta trabalha com sessões de teste (opção *ProgTest* do menu da janela principal apresentada na Figura 1) que são caracterizadas por um conjunto de informações, como por exemplo, o nome do arquivo fonte e do executável do programa sendo avaliado, o conjunto de casos de teste e os resultados alcançados. A ferramenta ainda permite a manipulação de casos de teste (*TestCase*), além da manipulação do modelo de defeitos (*FaultModel*), bem como do programa com defeitos injetados (*Inject*). Permite ainda a impressão de relatórios (*Report*) e acompanhar o andamento da avaliação dos defeitos injetados (*Status*).

Com relação à manipulação do programa com defeito injetados, a ferramenta permite injetar, ativar/desativar defeitos, visualizar o programa com defeitos injetados, além de permitir a edição do programa com defeitos injetados, quando o usuário identifica um defeito "real" no programa.

Atualmente, a ITool está sendo empregada em um estudo de caso utilizando-se o programa Space que trata-se de um sistema real desenvolvido pela ESA (European Space Agency). Aplicando-se o esquema de injeção de defeitos estabelecido com uma taxa de 10 FKLOC (10 defeitos a cada 1000 LOC´s), tem-se a injeção de 37 defeitos divididos em: 20 defeitos da classe Defeito de Entidade Simples Incorreta, 15 da classe Defeito de Falta de Entidade, nenhum defeito da classe Defeito de Entidade Espúria e dois da classe Defeito de Entidade Mal Empregada.

4. Conclusão

As taxonomias ou modelos de defeitos têm um papel fundamental no processo de desenvolvimento de software, pois podem retratar dos defeitos que podem a vir ocorrer durante o processo de desenvolvimento do software, permitindo assim que desenvolvedores concentrem suas atenções em determinadas partes — requisitos,

especificação, código, entre outros — do software.

No tocante à Injeção de Defeitos, os modelos de defeitos podem auxiliar na sistematização dos defeitos a serem injetados. Assim, esse trabalho concentra-se no estabelecimento de um esquema de injeção de defeitos baseado nos operadores de mutação do critério de teste Análise de Mutantes, utilizando-se como base a taxonomia de defeitos definida por DeMillo/Mathur.

Atualmente, a ferramenta ITool está sendo utilizada em um estudo de caso utilizando-se o programa Space para mostrar a factibilidade da utilização de injeção de defeitos de software baseado no esquema de injeção de defeito e no mapeamento sobre os operadores de mutação realizado.

Referências

[AGR89] Agrawal, H.; DeMillo, R.A.; Hathaway, B.; Hsu, W.; Hsu, Wy.; Krauser, E.W.; Martin, R.J.; Mathur, A.P.; Spafford, E.; *Design of Mutant Operators for the C Programming Language*, Technical Report SERC-TR-41-P, Software Enginnering Research Center, Purdue University, março 1989.

[BAS84] Basili, V.R.; Perricone, B.T.; *Software Errors and Complexity: An Empirical Investigation*, Communications of the Acm, V.27, N.1, janeiro 1984.

[BEI90] Beizer, B.; *Software Testing Techniques*, 2ª Edição, Van Nostrand Eeinhold, New York, 1990.

[BUD81] Budd, T.A.; *Mutation Analysis: Ideas, Examples, Problems and Prospects*, Computer Program Testing, North-Holand Publishing Company, 1981.

[CLA95] Clark, J.A.; e Pradham, D.K.; *Fault Injection - A Method for Validating Computer-System Dependability*, IEEE Computer, V.28, N.6, junho 1995, p.47-56.

[DEL93] Delamaro, M.E.; *Proteum - Um Ambiente de Teste Baseado na Análise de Mutantes*, Dissertação de Mestrado, ICMSC/USP, São Carlos, SP, outubro 1993.

[DEM78] DeMillo, R.A.; *Software Testing and Evaluation*, The Benjamim/ Commings Publishing Company, Inc, 1978.

[DEM95] DeMillo, R.A.; Mathur, A.P.; *A Grammar Based Fault Classification Scheme and its Application to the Classification of the Errors of TEX*, novembro 1995 (correspondência pessoal).

[MAR93] Martins, E.; *Validação Experimental da Tolerância a Falhas: A Técnica de Injeção de Falhas*, V Simpósio de Computadores Tolerantes a Falhas, São José dos Campos, SP, outubro 1993.

[NAK98] Nakagawa, E.Y.; Maldonado, J.C.; *Estudo e Priorização da Aplicação dos Operadores de Mutação para Injeção de Defeitos*, Relatório Técnico, ICMSC/USP, São Carlos, SP, março 1998 (em preparação).

[SOM97] Somani, A.K.; Vaidya, N.H.; *Understanding Fault Tolerance and Reliability*, IEEE Computer, V. 30, N. 4, abril, 1997.

[WEL95] Welch, B.B.; *Practical Programming in Tcl and Tk*, Prentice Hall, 1995.