Extensão e Validação de Ferramenta para Injeção de Falhas de Software J-SWFIT

Autor: **Bruno Pacheco Sanches** - brunopsanches@gmail.com
Orientadora: **Profa.Dra. Regina Lúcia de Oliveira Moraes** - regina@ft.unicamp.br

FT - FACULDADE DE TECNOLOGIA
UNICAMP - UNIVERSIDADE ESTADUAL DE CAMPINAS

Programa de Bolsas de Iniciação Científica - PIBIC/SAE

Palavras-chave: Falhas de *Software*, Injeção de Falhas, *Dependability*

Introdução

A tendência da indústria de *software* é desenvolver produtos a partir de componentes, trazendo dificuldades na integração e aumento da complexidade do *software*, aumentando a quantidade de falhas. Falhas de *software* são enganos cometidos por programadores que permanecem no código do programa, sendo uma das principais causas de defeitos. Como a completa eliminação das falhas é uma tarefa difícil, é natural presumir a existência de falhas em algum componente. Mesmo quando exaustivamente testados, defeitos se apresentam e, de forma inesperada, impactam o uso do software. A técnica de Injeção de Falhas é eficaz para o entendimento e validação do comportamento de sistemas de software em presença de falhas e exige ferramentas específicas para realizar a injeção de falhas e a monitoração da aplicação sob teste. O presente trabalho definiu os operadores que serão emulados, a arquitetura da ferramenta, o *framework* para a manipulação de *bytecodes* e desenvolveu o primeiro protótipo da ferramenta J-SWFIT, com o objetivo de injetar falhas de forma escalável e mensurar seu impacto. A ferramenta baseia-se na técnica G-SWFIT, cujo esquema está representado na Figura 1.

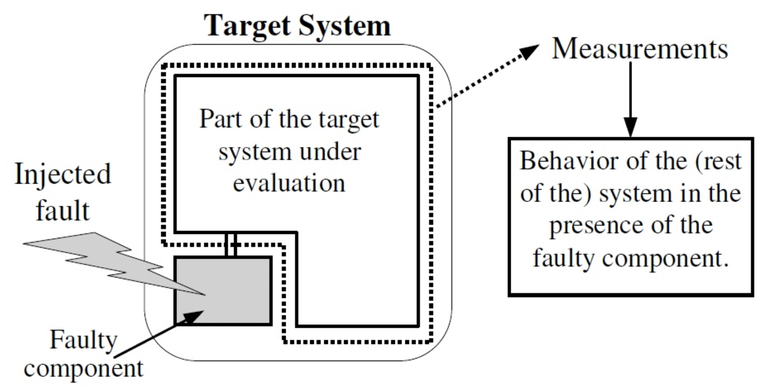

Figura 1: Ilustração da técnica de injeção, adaptado de (Durães e Madeira, 2006)

Metodologia

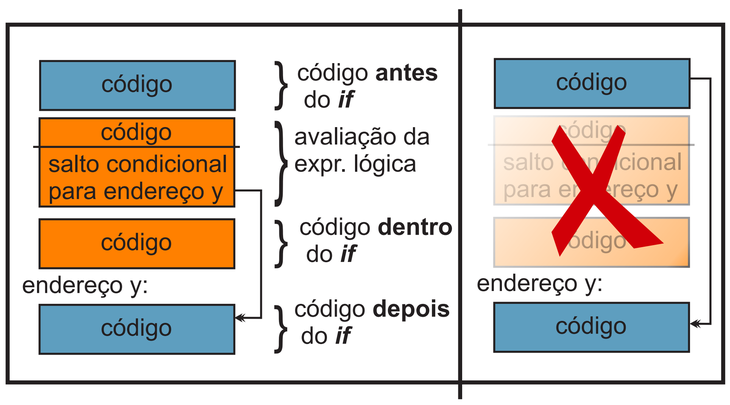

Figura 2: Padrão do operador MIFS analisado e modificado

As falhas só devem ser emuladas onde possam realmente existir, considerando as estruturas da lógica de programação. O significado e a lógica do encadeamento das instruções são interpretados. A identificação dos locais adequados à emulação do tipo de falha pretendido é feita sem a necessidade do código fonte original uma vez que a análise e a alteração é feita diretamente nos *bytecodes*. A Figura 2 ilustra o processo de emulação do operador MIFS (*Missing IF Construct*). Alterações que correspondam a erros sintáticos não são falhas realistas, pois uma vez apontadas pelo compilador elas seriam corrigidas antes da execução do programa. A metodologia da ferramenta consiste em analisar *bytecodes* de arquivos Java compilados, encontrar locais onde falhas específicas poderiam existir, injetar cada falha independentemente, executar o sistema com a falha presente, monitorar seu resultado, e ao final, comparar o comportamento na presença e na ausência de cada falha. O passo a passo da metodologia é apresentado na Figura 3.

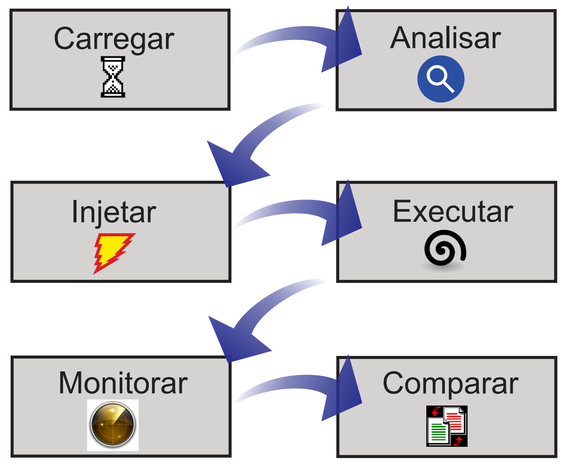

Figura 3: Passos que a ferramenta J-SWFIT se propõe realizar

Resultados e Discussão

O principal resultado atingido neste projeto foi a extensão e validação da ferramenta JSWFIT, melhorando sua arquitetura, complementando o desenvolvimento e integrando à uma interface gráfica para prover usabilidade a terceiros. Apesar de ainda ser limitada em termos dos operadores que consegue emular, foi construída com base em uma arquitetura bem distribuída, funcional, e extensível. A Figura 4 mostra a simplicidade da interface gráfica que a ferramenta J-SWFIT oferece. A Figura 5 representa a arquitetura detalhada que foi proposta em um nível de abstração que pode ser facilmente entendida e estendida. A técnica de polimorfismo é a grande responsável pela extensibilidade da ferramenta, uma vez que se usa extensão de classes abstratas, definição de algoritmos de localização e injeção de falhas para se implementar o operador a ser emulado.

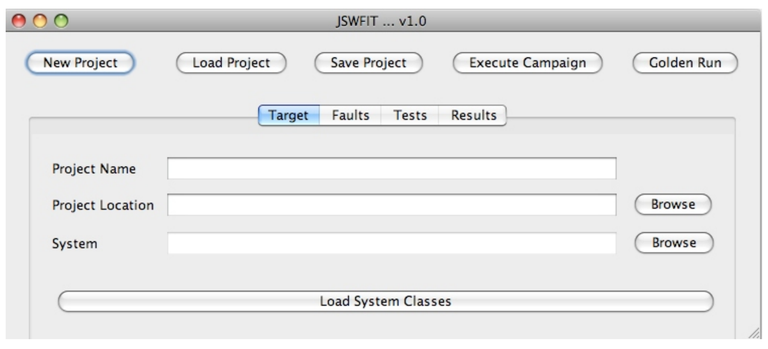

Figura 4: Interface da ferramenta J-SWFIT

A ferramenta permite escalar o processo de injeção de falhas para todos os operadores propostos.

Um uso importante da ferramenta é fornecer uma maneira de realizar comparações entre componentes de software para que se possa definir qual destes componentes proporciona maior robustez quando integrado a um sistema crítico.

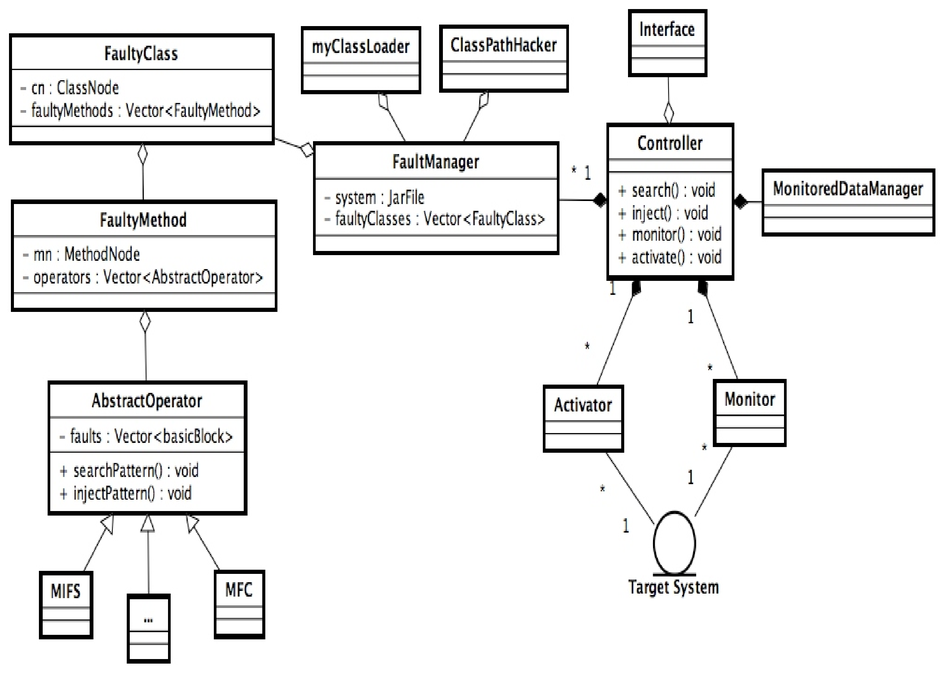

Figura 5: Diagrama de Classes da ferramenta J-SWFIT

Conclusões

A ferramenta J-SWFIT está construída sobre uma arquitetura sólida e estruturada. Seu desenvolvimento mostra que a técnica G-SWFIT pode ser adaptada para códigos feitos em Java, que de fato é possível automatizar todo o processo de injeção de falhas, e que isto é escalável. A sua evolução e aumento de funcionalidade dependem principalmente do desenvolvimento de novos operadores de falhas, inclusive para plataformas diversas. Conclue-se ainda que a injeção de falhas é uma técnica eficiente para a validação de sistemas computacionais, tendo na J-SWFIT uma ferramenta implementada com sucesso e que poderá auxiliar para o aumento de *dependability* destes sistemas.

Referências Bibliográficas

DURAES, J., MADEIRA, H. "Emulation of Software Faults: A Field Data Study and PracticalApproach". IEEE Transactions on Software Engineering, vol. 32, n. 11, November 2006.

LYU M., “Handbook of Software Reliability Engineering”, IEEE Computer Society Press, MCGraw-HILL, 1996.