Ferramentas de teste: JaBUTi e MuJAVA

Henrique Neves da Silva
Silvia Regina Vergilio

UFPR - Curitiba

29 de Abril de 2019

Sumário

Introdução

Ferramentas de teste

- Teste Estrutural $\longrightarrow$ JaBUTi.
- Teste de Mutação $\longrightarrow$ MuJAVA.

Ferramenta JaBUTi

Um pouco sobre suas características

- JaBUTi (Java Bytecode Understanding and Testing).
- Ferramenta desenvolvida no Instituto de Ciências Matemáticas e de Computação – ICMC/USP.
- Apoia o teste estrutural para programas Java.
- Implementa os critérios baseados em fluxo de controle e critérios baseados em fluxo de dados.
- Realiza a análise sobre o bytecode Java (.class) e não sobre o código fonte (.java).

Ferramenta JaBUTi

Sobre o arquivo bytecode

- Uma das razões pela popularização do Java.
- O arquivo binário é interpretado pela Java Virtual Machine (JVM).
- A partir de uma coleção de instruções bytecode, o código `a = b + c` pode ser traduzido de acordo com a imagem abaixo.

Slot	Variable
...	...
2	a
3	b
4	c
...	...

(a) Local Variable Vector

Bytecode Instruction	Operand Stack
12: iload_3	Value of b ...
13: iload_4	Value of c Value of b ...
14: iadd	Value of b + c ...
15: istore_2	...

(b) Operand Stack Simulation

Ferramenta JaBUTi

Parte prática - passo a passo

1. `git clone https://github.com/neves01/JaBUTi4Run.git` ou simplesmente baixe o ZIP `https://github.com/neves01/JaBUTi4Run/archive/master.zip`.
2. Na pasta jabuti, execute `chmod -v +x run.sh` e depois `./run.sh`
3. Clique em File $\longrightarrow$ Open Class $\longrightarrow$ Selecione o arquivo .class que será testado.
4. Em **classpath** coloque o caminho completo até o arquivo já selecionado, no meu caso `/home/henrique/Downloads/JaBUTi4Run-master/example/TriTyp/`
5. Clique em OK $\longrightarrow$ Em **User Packager** selecione o nome da classe que será testada TriTyp e clique no segundo >>.
6. No botão **Select** e dê um nome ao projeto. $\longrightarrow$ OK.

Ferramenta JaBUTi

Continuação parte prática...

1. Clique em **File** $\longrightarrow$ **Save Instrumented Classes** $\longrightarrow$ **Yes**.
2. Clique em **Test Case** $\longrightarrow$ **Executing JUnit Test Set**.
   - Informe o caminho para o fonte e binário do arquivo de teste. `/.../JaBUTi4Run-master/example/TriTyp/`
   - Test suite full qualified name: `paper.TriTypTest`
   - JaBUTi library: `jabuti.jar`
   - Clique em **Compile Test Case**.
   - Clique em **Run Normally (no trace)**.
   - Clique em **Run Collecting Trace Information**.
3. Na tela principal Update $\longrightarrow$ Update.
4. Customize o escopo de cobertura em **Summary**.
5. Relatório é gerado em **Reports** $\longrightarrow$ **Custom Reports**.

Ferramenta MuJava

Um pouco sobre suas características

- Ferramenta para teste de mutação em programas Java.
- Provê uma coleção de operadores de mutação tradicionais e voltado para o escodo de orientação a objeto.
- Gera automaticamente os mutantes, executa-os junto ao conjunto de testes T e posteriormente apresenta o escore.
- As funções da ferramenta consiste em:
  - Geração de mutantes.
  - Análise de mutantes.
  - Gerenciamento de casos de teste fornecidos pelo usuário.

Ferramenta MuJAVA

Parte prática - passo a passo

1. `git clone https://github.com/neves01/MuJava4Run.git` ou simplesmente baixe o ZIP `https://github.com/neves01/MuJava4Run/archive/master.zip`.
2. Na pasta **configuration**, determine o caminho do projeto.
   - `mujava.properties` $\longrightarrow$ `/.../MuJava4Run/examples/session1`
   - `mujavaCLI.properties` $\longrightarrow$ `/.../Downloads/MuJava4Run`

   OBS: Confira o nome do arquivo `mujava/jar/mujava.jar`
3. A pasta do projeto deve estar estruturada da seguinte forma:
   - session1
     - classes
     - result
     - src
     - testset
   - Ou utilizar o script `makeStructure.sh`

Ferramenta MuJAVA

Continuação parte prática...

- Na pasta `src` o arquivo a ser mutado.
- Na pasta `classes` colocar o arquivo `.class` do arquivo presente na pasta `src`.
- Na pasta `testset` colocar o arquivo `.class` do arquivo de teste.
- Na pasta `result` ficará os mutantes gerados.
  OBS: Criar esta pasta, ela está faltando.
- Para gerar os mutantes executar `./generator.sh`
- Para testar os mutantes gerados executar `./tester.sh`

Referências

Artigos utilizados

- Ma, Y. S., Offutt, J., Kwon, Y. R. (2005). **MuJava: an automated class mutation system**. Software Testing, Verification and Reliability, 15(2), 97-133.
- Vincenzi, Auri E Wong, W Delamaro, Márcio Maldonado, José. (2003). **JaBUTi: A Coverage Analysis Tool for Java Programs**. XVII Simpósio Brasileiro de Engenharia de Software(SBES 2003).
- Manual e arquivos: `https://jacksonpradolima.github.io/teaching/2016-01-UFPR`.

Ferramentas de teste: JaBUTi e MuJAVA

Henrique Neves da Silva
Silvia Regina Vergilio

UFPR - Curitiba

29 de Abril de 2019