1. Operadores de Mutação para a Linguagem JAVA

1.1. Nível de Método

1.1.1. Operadores aritméticos

1) AORB (Arithmetic Operator Replacement Binary): substitui operadores aritméticos binários básicos por outros operadores aritméticos binários;
2) AORS (Arithmetic Operator Replacement Short-cut): substitui os operadores aritméticos de curto-circuito por outros operadores aritméticos unários;
3) AOIU (Arithmetic Operator Insertion Unary): insere operadores aritméticos unários básicos;
4) AOIS (Arithmetic Operator Insertion Short-cut): insere operadores aritméticos de curto-circuito;
5) AODU (Arithmetic Operator Deletion Unary): exclui operadores aritméticos unários básicos;
6) AODS (Arithmetic Operator Deletion Short-cut): exclui operadores aritméticos de curto-circuito;
7) AORU (Arithmetic Operator Replacement Unary): substitui operadores aritméticos unários básicos por outros operadores aritméticos unários.

1.1.2. Operadores relacionais

8) ROR (Relational Operator Replacement): substitui operadores relacionais por outros operadores relacionais e substitui todo o predicado por verdadeiro e falso.

1.1.3. Operadores condicionais

9) COR (Conditional Operator Replacement): substitui operadores condicionais binários por outros operadores condicionais binários;
10) COD (Conditional Operator Deletion): exclui operadores condicionais unários;
11) COI (Conditional Operator Insertion): insere operadores condicionais unários.

1.1.4. Operadores de deslocamento

12) SOR (Shift Operator Replacement): substitui os operadores de deslocamento por outros operadores de deslocamento.

1.1.5. Operadores lógicos

13) LOR (Logical Operator Replacement): substitui operadores lógicos binários por outros operadores lógicos binários;
14) LOI (Logical Operator Insertion): insere operador lógico unário;
15) LOD (Logical Operator Delete): exclui operador lógico unário;

1.1.6. Operadores de atribuição

16) ASRS (Assignment Operator Replacement Short-Cut): substitui os operadores de atribuição de curto-circuito por outros operadores de curto-circuito do mesmo tipo.

1.1.7. Operadores de exclusão

17) SDL (Statement DeLetion): exclui cada instrução executável, comentando-as;
18) VDL (Variable DeLetion): todas as ocorrências de referências de variáveis são excluídas de todas as expressões;
19) CDL (Constant DeLetion): todas as ocorrências de referências constantes são excluídas de todas as expressões;
20) ODL (Operator DeLetion): cada operador aritmético, relacional, lógico, bit a bit e de deslocamento é excluído das expressões e operadores de atribuição.

1.2. Nível de Classe

1.2.1. Encapsulamento

1) AMC (Access Modifier Change): altera o nível de acesso para variáveis de instância e métodos para outros níveis de acesso.

1.2.2. Herança (Inheritance)

2) IHI (Hiding variable Insertion): insere uma variável oculta em uma subclasse;
3) IHD (Hiding variable Deletion): exclui uma variável oculta, uma variável em uma subclasse que tem o mesmo nome e tipo que uma variável na classe pai;
4) IOD (Overriding method Deletion): exclui uma declaração inteira de um método sobrescrito em uma subclasse para que as referências ao método usem a versão do pai;
5) IOP (Overriding method calling Position change): move as chamadas para métodos sobrescritos para a primeira e a última instrução do método e uma instrução para cima e para baixo.
6) IOR (Overriding method Rename): renomeia as versões desses métodos nas classes pai para que a substituição não possa afetar o método das classes pai;
7) ISI (Super keyword Insertion): insere a palavra reservada super para que uma referência à variável ou ao método vá para a variável ou método da instância sobrescrita;
8) ISD (Super keyword Deletion): exclui ocorrências da palavra reservada super para que uma referência à variável ou ao método vá para a variável ou método de instância sobrescrita;
9) IPC (explicit call of a Parent's Constructor deletion): exclui chamadas aos construtores da classe pai (super), fazendo com que o construtor padrão da classe pai seja chamado.

1.2.3. Polimorfismo (Polymorphism)

10) PNC (New method call with Child class type): altera o tipo instanciado de uma referência de objeto;
11) PMD (Member variable Declaration with parent class type): altera o tipo declarado de uma referência de objeto para o pai do tipo original declarado;
12) PPD (Parameter variable Declaration with child class type): é o mesmo que o PMD, exceto que ele opera com parâmetros em vez de variáveis locais e de instância;
13) PCI (type Cast operator Insertion): altera o tipo atual de uma referência de objeto para o pai ou filho do tipo original declarado;
14) PCD (type Cast operator Deletion): exclui o operador de conversão de tipo;
15) PCC (Cast type Change): alterar o tipo em que uma variável deve ser convertida;
16) PRV (Reference assignment with other comparable Variable): altera operandos de uma atribuição de referência a serem atribuídos a objetos de subclasses;
17) OMR (Overloading Method contents Replace): verifica se os métodos sobrecarregados são chamados adequadamente;

18) OMD (Overloading Method Deletion): exclui as declarações de sobrecarga de método, uma de cada vez;
19) OAC (Arguments of Overloading method call Change): altera a ordem ou o número dos argumentos nas invocações de métodos, mas apenas se houver um método sobrecarregado que possa aceitar a nova lista de argumentos.

1.2.4. Java-Specific Feature

20) JTI (This keyword Insertion): insere a palavra reservada this;
21) JTD (This keyword Deletion): exclui a palavra reservada this quando utilizada;
22) JSI (Static modifier Insertion): adiciona o modificador static para alterar variáveis de instância para variáveis de classe;
23) JSD (Static modifier Deletion): remove o modificador static para alterar variáveis de classe para variáveis de instância;
24) JID (member variable Initialization Deletion): remove a inicialização de variáveis membro na declaração de variável para que as variáveis membro sejam inicializadas com os valores padrão apropriados em Java;
25) JDC (java-supported Default constructor Creation): força o Java a criar um construtor padrão excluindo o construtor padrão implementado;
26) EOA (reference Assignment and content assignment replacement): substitui uma atribuição de uma referência de ponteiro por uma cópia do objeto, usando a convenção Java de um método clone ();
27) EOC (reference Comparison and content comparison replacement): substitui == por equals();
28) EAM (Acessor Method change): altera um nome de método de acesso get() para outros nomes de métodos de acesso compatíveis, em que compatível significa que as assinaturas são as mesmas;
29) EMM (Modifier Method change): faz o mesmo que o EMM, exceto que funciona com métodos de acesso set() em vez de métodos de acesso get().

1.3 Considerações

Alguns trabalhos podem conter operadores de mutação com nomes ligeiramente diferentes dos que foram apresentados neste trabalho.

2. Ferramentas de Teste de Mutação para JAVA

2.1. MuJAVA

2.1.1. Funcionamento

A ferramenta, que possui interface gráfica, é dotada de um gerador de mutantes, capaz de gerar mutantes em nível de método e de classe, um visualizador de mutantes, que permite visualizar a quantidade geral de mutantes e a quantidade de mutantes gerados por cada operador, além do código original e do respectivo mutante, e um executor de mutantes, que executa cada mutante contra o conjunto de testes, cujo resultado é informado através do escore de mutação.

Na primeira tela, o usuário da ferramenta escolhe os arquivos a serem testados, selecionam os operadores a serem aplicados e executam o gerador de mutantes. Nas demais telas é possível visualizar as respostas por nível.

2.1.2. Restrições

A visualização dos mutantes na ferramenta não é considerada boa em virtude da complexidade.

2.1.3. Vantagens

A ferramenta combina a estratégia Mutant Schemata Generation (MSG) e a tradução de bytecode, reduzindo o tempo de compilação.

2.2. MAJOR

2.2.1. Funcionamento

O componente compiler-integrated mutator, responsável por gerar as mutações, é acionado por comandos integrados ao compilador ou por meio de uma linguagem de domínio específica (DSL), ao passo que o componente mutation analyzer implementa um método para análise dos resultados.

2.2.2. Restrições

Número limitado de operadores de mutação e ausência de suporte para geração de dados de teste.

2.2.3. Vantagens

Módulo de geração de mutantes integrado ao compilador e o analisador integrado ao Apache Ant e ao Junit.
Suporte à mutação fraca (referente à apresentado de estado diferente do programa original) e mutação forte (convencional).

2.3. Pitest (PIT)

2.3.1. Funcionamento

A ferramenta é acionada por meio de linha de comando e exibe um relatório ao final do processamento. O analista de teste deverá utilizar o relatório para adicionar casos de teste para matar os mutantes.

2.3.2. Restrições

Não há interface gráfica para utilização da ferramenta, não empregada todos os operadores disponíveis para JAVA e os plug-ins disponíveis para IDEs não aproveitam todo potencial da ferramenta.

2.3.3. Vantagens

Pode ser utilizado em conjunto com IDEs, embora haja limitações.

2.4. PROTEUM/AJ

2.4.1. Funcionamento

A ferramenta, por meio de sua interface gráfica, recebe como entrada um arquivo em formato ZIP com todos os arquivos da aplicação a ser testada, um conjunto de cados de testes em JUnit e quatro arquivos de configuração. O usuário selecionará os operadores de mutação e irá aplicá-los aos elementos-alvo. Após a execução, é exibido ao usuário relatório de escore de mutação e visualização de mutantes.

2.4.2. Restrições

Dependência do AspectJ-front e do Sistema Operacional Linux de 32 bits.

2.4.3. Vantagens

Utilização de um banco de dados para armazenamento dos resultados para criação de uma base histórica.

2.5. Jester

2.5.1. Funcionamento

A ferramenta encontra código não coberto pelos testes, modifica esse código e executa os testes. Caso caso os testes continuem passando, ela exibe uma mensagem com o código fonte que foi alterado.
A classe principal é a jester.TestTester, cujo argumento a ser enviado é o nome da classe de teste para o programa a ser analisado.

2.5.2. Restrições

Não possui interface gráfica e sua execução é lenta. Ele também pode ficar parado por problemas no classpath.

2.3.3. Vantagens

A única vantagem relatada é a possível diminuição de bugs.

2.6. Jumble

2.6.1. Funcionamento

Por meio de um plug-in na IDE Eclilpse, o usuário seleciona a classe a ser testada e aciona a opção “Jumble Mutation Tester / Analyse tests of this class”.
Existe uma interface gráfica não fornecida pelo projeto.

2.6.2. Restrições

Número reduzido de operadores de mutação.

2.6.3. Vantagens

A única vantagem relatada é a possível diminuição de bugs.

2.7. Javalanche

2.7.1. Funcionamento

A ferramenta verifica se os testes podem sere executados por ela sem falhas e, caso seja possível, procede o escaneamento do projeto para determinar todas as classes que podem ser “mutadas” e executa as mutações. Ao final do processo, o resultado pode ser analisado. Um relatório em html é gerado com as mutações para cada classe.

2.7.2. Restrições

É necessário fazer a configuração do banco de dados, que não parece muito simples. A documentação é pobre, não informando, por exemplo, os operadores de mutação utilizados.

2.7.3. Vantagens

Em alguns testes com softwares reais, chegou a produzir menos de 3% de mutantes equivalentes.