Estudo Teste de Mutação Pitest

Segundo site oficial Pitest:

Teste de mutação do mundo real

O PIT é um sistema de teste de mutação de última geração , fornecendo cobertura de teste padrão ouro para Java e jvm. É rápido, escalonável e se integra a ferramentas modernas de teste e construção.

Instalação simples Maven AQUI

Pode ser utilizado via maven ou linha de comando

PIT intellij plugin

Pitclipse

O que o teste de mutação faz?

Como o próprio nome diz, mutações(alterações) são feitas no código.

Qual o objetivo?

Cobrir a maior parte de código possível para evitar de um erro escapar quando alterações provenientes de melhorias ou correções de bugs forem feitas, ou seja, se um código for alterado e estiver gerando algum problema os testes devem ser capazes de informar este erro.

Termos:

Mutante: Alteração feita no código pela ferramenta;
Mutante morto: Os testes da aplicação cobriram a alteração feita pela ferramenta(testes falharam como esperado).
Mutante sobreviveu: Os testes da aplicação não cobriram a alteração feita pela ferramenta(testes não falharam).
Obs.: O fato de um mutante sobreviver não é necessariamente errado, pois uma alteração feita pela ferramenta pode não ser coesa em termos de lógica de programação.

Lista de mutações possíveis e personalização:

https://pitest.org/quickstart/mutators/

No geral ele pode retornar nulo, negar a condição atual e inverter o operador que está no código original, por exemplo:

Original conditional	Mutated conditional
==	!=
!=	==
<=	>
>=	<
<	>=
>	<=

Exemplo código original:

`

```
if(idade >= 18){
   pessoa.setDescricao("maior de idade");
}else if(idade < 18){
   pessoa.setDescricao("menor de idade");
}
```

```
if(idade >= 18){
   pessoa.setDescricao("maior de idade");
}else if(idade <= 18){
   pessoa.setDescricao("menor de idade");
}
```

O operador lógico foi alterado de “<” para “<=” e neste cenário o teste de mutação encontrou um mutante que não foi morto.

Será que é um problema mesmo este mutante?

Avaliando o cenário é possível ver que isso não ocorreria, pois o primeiro if seria verdadeiro caso fosse igual a 18, logo este mutante não apresenta perigo para a aplicação.

Prós:

- Gera relatório visual(em HTML) explicativo do código;
- Automatiza processo de sabotação de código, para validar se o problema está coberto pelos testes;
- Não é preciso escrever mais códigos, o próprio PIT cria essas variantes do fonte;
- Demonstra pontos que estão com cobertura, mas que não estão efetivamente cobertos.

Contras:

- Aumenta o tempo da fase de testes;
- Alguns problemas apresentados por ele, às vezes, podem não ser 100% assertivos;
- Tempo:
    - Pacote xxxxxlevou 12 minutos;
    - Pacote xxxxxxxlevou 32 minutos;
    - Um projeto completo pode levar mais de 12 horas