Análise e Teste de Software

Automatic Test Generation

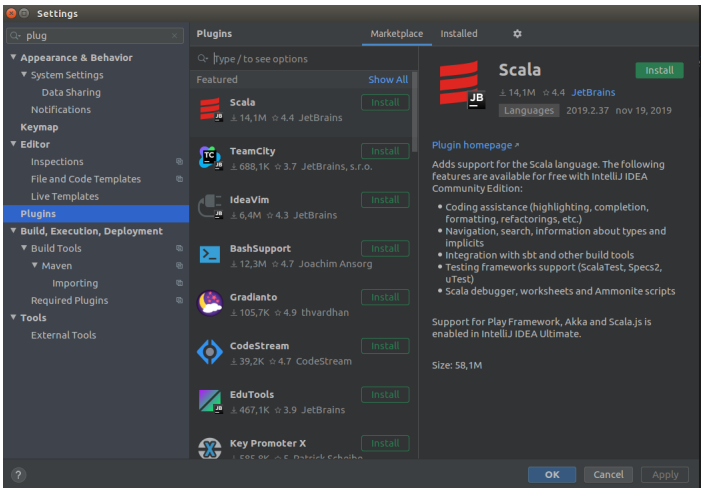
Universidade do Minho

2023/2024

1 Introdução ao EvoSuite

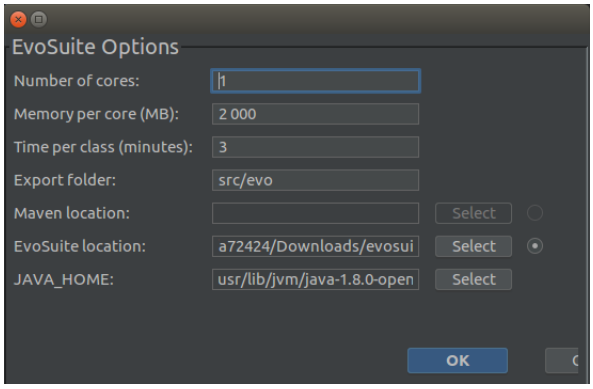
O EvoSuite é uma ferramenta com capacidade de gerar automaticamente testes unitários JUnit para uma dada classe ou projeto. Os testes unitários gerados pelo EvoSuite exercitam todos os métodos que esta ferramenta consegue entender e utilizar sem problemas.

Existem várias formas de utilização do EvoSuite, sendo a mais completa o seu uso pelo terminal. Nesta ficha o foco é o seu uso enquanto plug-in do IntelliJ IDEA. Para a instalação deste plug-in, deve-se ir ao menú de plug-ins do IntelliJ IDEA localizado em `File>Settings>Plugins`, abrindo-se assim a seguinte janela:



Deve-se aqui pesquisar pelo plugin EvoSuite (especificamente o EvoSuite (XenoAmess TPM)) e fazer a sua instalação. Para finalizar a preparação do plug-in deve-se descarregar o executável do EvoSuite, da página `http://www.evosuite.org/downloads/`.

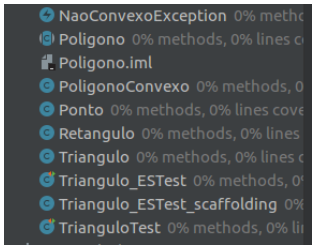
Para utilizar o EvoSuite numa classe ou projeto, deve-se fazer o clique com o lado direito do rato na classe ou projeto que se pretende testar e escolher a opção `Run EvoSuite`, sendo que uma janela semelhante a esta se deve apresentar:



É de notar que a janela pode abrir com dimensões bastante pequenas e algumas opções não aparecerem por causa disso. Isso resolve-se redimensionando a janela.

Pode-se configurar algumas opções quanto à execução do EvoSuite, mas na primeira execução é necessário indicar a localização do executável EvoSuite previamente descarregado assim como da instalação do Java (tipicamente uma pasta em `/usr/lib/jvm`). O resto das configurações relacionam-se com a quantidade de esforço que o EvoSuite deve dedicar a produzir testes, e aonde os colocar. De notar que é necessário a utilização do Java 8 no projeto a testar para o EvoSuite ser compatível.

Na seguinte imagem pode-se ver alguns ficheiros produzidos pelo EvoSuite, no meio do projeto dos polígonos utilizado em uma das aulas anteriores:



A classe `Triangulo_ESTest` contém os testes gerados pelo EvoSuite, e a respetiva classe `Triangulo_ESTest_Scaffolding` que esta estende, contém definições para que os testes sejam executados em certas condições (usando algumas notações já familiares como @Before e @After, do JUnit 4). O EvoSuite poderá produzir ficheiros duplicados que darão conflitos ao tentar executar testes no IntelliJ, sendo que se deve remover esses duplicados do projeto. Pode-se depois correr os testes e verificar que todos devem passar; na secção Exercícios são sugeridas algumas formas de corrigir erros que possam surgir entretanto.

Para execuções mais avançadas do EvoSuite, deve-se fazer a sua invocação pelo terminal, sendo que isto facilita o seu uso com outras ferramentas, nomeadamente o Maven e ferramentas de cobertura como o JaCoCo. Pode-se ler mais sobre o uso do EvoSuite no terminal neste link: `http://www.evosuite.org/documentation/tutorial-part-1/`. Este último link apenas cobre o uso no terminal sem integração com o Maven; para integração com o Maven este link descreve o processo: `https://www.evosuite.org/documentation/tutorial-part-2/`. Pode-se ler mais sobre medição da cobertura de teses EvoSuite neste link, aonde é explicada a integração com o JaCoCo e com o PIT: `http://www.evosuite.org/documentation/measuring-code-coverage/`.

2 Exercícios

O EvoSuite poderá não funcionar corretamente com uma versão do Java que não seja o Java 8. Para alterar a versão de Java utilizada num projeto no IntelliJ, deverá ir a `File>Project Structure>Project` e alterar o Project SDK e Project language level para Java 8. No mesmo menú, em `Libraries`, adicione uma Java library, e selecione o ficheiro EvoSuite que poderá descarregar da página. Caso tenha problemas de execução do EvoSuite via Maven, instale o Java 8 e poderá utilizar o comando `update-java-alternatives` para facilmente trocar entre versões de Java instaladas na sua máquina.

1. Relembre o projeto `Turma` utilizado em aulas anteriores. Utilizando o IntelliJ:
   (a) Instale o plugin do EvoSuite e descarregue o executável deste. Altere o projeto para utilizar Java 8 e adicione a dependência do EvoSuite.
   (b) Gere automaticamente testes unitários para a classe `Aluno` e execute-os sem olhar para o código fonte. Estes deverão não executar pois serão testes JUnit 4; adicione as dependências necessárias para o JUnit 4 (nao é problema ter JUnit 4 e 5 no mesmo projeto), e execute os testes. Se estes ficarem colocados numa pasta `evo/`, mova-os para a pasta um nível acima e já os deverá conseguir executar. Raciocine sobre os resultados obtidos, e o porquê destes.
   (c) Analise o código fonte dos testes gerados. Raciocine sobre as vantagens e desvantagens da utilização da geração de testes automáticos. Analise a cobertura dos testes gerados. Certifique-se que está a utilizar apenas os testes gerados pelo EvoSuite e não outros testes que já existam anteriormente no projeto.
   (d) Utilize o PIT para testar os testes gerados pelo EvoSuite. Raciocine sobre os resultados obtidos; caso estes sejam pouco interessantes, experimente gerar testes com o EvoSuite de novo, mas dando-lhe mais recursos e tempo de execução. Repita a análise com o PIT.
   (e) Repita os exercícios anteriores com a classe `Turma`.

2. Repita os exercícios anteriores, mas agora com a configuração Maven que possui para este projeto:

   (a) Siga o tutorial apontado acima para a configuração Maven do EvoSuite, mas para o seu projeto `Turma` ao invés do projeto demonstrativo mostrado nessa página.

   (b) Utilize o comando `mvn evosuite:generate` para gerar testes unitários para o seu projeto.

   (c) Consulte a pasta escondida gerada pelo EvoSuite aonde os dados gerados foram colocados, com `ls .evosuite/best-tests/`. Analise brevemente os seus conteúdos.

   (d) Siga ainda as instruções do tutorial do EvoSuite para exportar os testes gerados para a pasta de testes. Tente executar os testes.

   Poderá ter um ou dois tipos de mensagens de erro. Uma poderá surgir por faltar a dependência de runtime do EvoSuite, como descrito na página do tutorial; a outra será por o plugin de Maven do EvoSuite gerar testes JUnit 4. Felizmente, há soluções para utilizar testes JUnit 4 e 5 no mesmo projeto, nomeadamente a dependência `junit-vintage-engine`. Altere o seu ficheiro `pom.xml` para possuir esta dependência:

```
<dependency>
    <groupId>org.junit.vintage</groupId>
    <artifactId>junit-vintage-engine</artifactId>
    <version>${junit.jupiter.engine.version}</version>
</dependency>
```

   (e) Configure e utilize o JaCoCo para medir a cobertura dos novos testes gerados. Caso este não funcione corretamente, tenha em atenção que poderá ser necessário alterar a configuração do ficheiro `pom.xml` de acordo com a descrição disponível do website do EvoSuite.

   (f) Configure e utilize o PIT para medir a qualidade dos testes gerados. Mais uma vez, poderá ser necessário recorrer ao website do EvoSuite para fazer as adaptações necessárias. Caso o PIT não consiga encontrar mutações, relembre que este *poderá* necessitar de uma declaração de package válida em todas as classes em causa.