EvoSuite

Antonio Vélez Estévez, Universidad de Cádiz.

¿Qué es EvoSuite?

EvoSuite es una herramienta para generar casos de prueba con asertos para clases escritas en el lenguaje de programación Java.

Fue creada originalmente por G.Fraser y A.Arcuri y se presentó en “Evolutionary Generation of Whole Test Suites,” in International Conference On Quality Software (QSIC), Los Alamitos, CA, USA, 2011, pp. 31-40.

- Generación de pruebas de JUnit 4 para las clases seleccionadas.
- Optimización de distintos criterios de cobertura, como por línea, ramas, salidas y pruebas.
- Minimización de pruebas: solo los que logran cobertura se conservan.
- Generación de asertos de JUnit para capturar el comportamiento actual de las clases bajo prueba.
- Las pruebas se ejecutan en un sandbox para prevenir operaciones potencialmente peligrosas.
- Sistema de ficheros virtual.
- Red virtual.

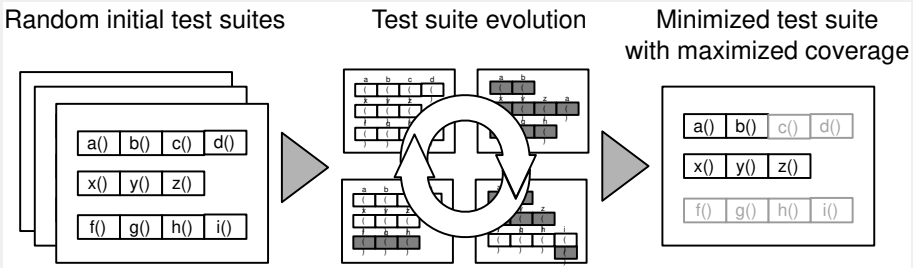

Figura: Proceso de EvoSuite (Imagen extraída del artículo citado)

1. Se genera un conjunto inicial de conjuntos de pruebas.
2. Se evolucionan usando técnicas de búsqueda que maximicen el criterio de cobertura de código escogido (algoritmo genético).
3. Se minimiza el conjunto de pruebas escogido.

Para poder usar EvoSuite con Maven necesitamos lo siguiente:

Requisitos

- Java Development Kit >= 8. **`sudo apt install openjdk-8-jre-headless`**.
- Maven. **`sudo apt-get install maven`**.
- Un editor de texto, **`vi`**, **`emacs`**, etc.
- (Opcional pero cómodo) Eclipse e IntelliJ Idea tienen plugins para integrar EvoSuite.

Para usar EvoSuite la forma más cómoda es usar Maven. Para ello primero crearemos un proyecto simple:

Orden para generar proyecto simple con Maven

```
mvn -B archetype:generate \
  -DarchetypeGroupId=org.apache.maven.archetypes \
  -DgroupId=es.uca.muiisc.psi \
  -DartifactId=evosuitepresentacion
```

Esto nos generará una carpeta con un fichero pom.xml en su interior que tendremos que modificar para añadir las dependencias.

.

```
evosuitepresentacion/
pom.xml
    src/
        main/es/uca/muiisc/psi/
            App.java
        test/es/uca/muiisc/psi/
            AppTest.java
```

Para usar EvoSuite la forma más cómoda es usar Maven. Para ello primero agregamos la dependencia con junit que usaremos para ejecutar los tests automáticos generados por EvoSuite.

Dependencias necesarias pom.xml

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Tambien agregamos la dependencia con **evosuite-standalone-runtime** para poder ejecutar las pruebas.

Dependencias necesarias pom.xml

```
<dependencies>
    [...]
    <dependency>
        <groupId>org.evosuite</groupId>
        <artifactId>evosuite-standalone-runtime</artifactId>
        <version>1.0.6</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Para usar EvoSuite de una forma más organizada y sin preocuparnos por el **classpath**, tenemos que agregar el plugin **evosuite-maven-plugin**.

Plugins necesarios pom.xml

```
<build>
   <plugins>
     <plugin>
       <groupId>org.evosuite.plugins</groupId>
       <artifactId>evosuite-maven-plugin</artifactId>
       <version>1.0.6</version>
     </plugin>
   </plugins>
 </build>
```

Para poder descargarnos el plugin de forma automática tenemos que agregar el repositorio donde se encuentra en el **pom.xml**.

Repositorio de plugins necesarios pom.xml

```
<pluginRepositories>
  <pluginRepository>
    <id>EvoSuite</id>
    <name>EvoSuite Repository</name>
    <url>http://www.evosuite.org/m2</url>
  </pluginRepository>
</pluginRepositories>
```

Para generar las pruebas primero tenemos que tener código que probar. Para probar algo diferente a lo que hay en la página de EvoSuite crearemos la clase **`Factorial.java`**, en la que implementaremos la operación factorial.

$$n! = n \cdot (n-1) \cdot (n-2) \cdots 1$$

```java
// -- clase Factorial.java --
package es.uca.muiisc.psi;

class Factorial {
    public int factorial(int n) {
        if(n == 0)
            return 1;
        return n * factorial(n-1);
    }
}
```

Para generar las pruebas con EvoSuite ejecutaremos la siguiente orden con Maven:

```
maven evosuite:generate
```

Esta orden se puede ejecutar gracias al plugin que incluimos en el fichero `pom.xml` . Las pruebas generadas por EvoSuite se guardan en la ruta absoluta:

```
$RUTA_PROYECTO$/.evosuite/best-tests/$NOMBRE_PAQUETE$
```

Siendo:

- `$RUTA_PROYECTO$` la ruta a la raíz del proyecto creado con Maven.
- `$NOMBRE_PAQUETE$` el nombre del paquete donde se encuentre la clase para la que queramos ver las pruebas que ha generado EvoSuite.

Para incluir las pruebas generadas por EvoSuite en el proyecto de Maven, tendremos que escribir la siguiente orden:

```
maven evosuite:export
```

Seguidamente, podremos ejecutar las pruebas haciendo uso de la orden **`mvn test`**.

Para realizar el análisis de cobertura de las pruebas generadas por EvoSuite podemos usar los siguientes plugins:

Plugins

- Jacoco.
- Cobertura.
- Clover.
- PIT.

Nosotros elegimos **`cobertura`** por su facilidad de instalación y configuración.

pom.xml

```
<plugins>
    [...]
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>cobertura-maven-plugin</artifactId>
      <version>2.7</version>
    </plugin>
</plugins>
```

Una vez tenemos las pruebas exportadas y ejecutadas, podemos analizar la cobertura de la siguiente forma:

- `mvn cobertura:cobertura`, le indicamos a `cobertura` que comienze a analizar.
- `mvn cobertura:dump-file`, le indicamos a `cobertura` que nos muestre los resultados por pantalla.

- `http://www.evosuite.org/documentation/tutorial-part-1/` [Accedido el 17-12-2019].
- `http://www.evosuite.org/documentation/tutorial-part-2/` [Accedido el 17-12-2019].
- `http://www.evosuite.org/documentation/measuring-code-coverage/` [Accedido el 18-12-2019].
- "Evolutionary Generation of Whole Test Suites," in *International Conference On Quality Software (QSIC)*, Los Alamitos, CA, USA, 2011, pp. 31-40.