Measuring Test Coverage & Evosuite

TAN, Shin Hwei
陈馨慧

Southern University of Science and Technology

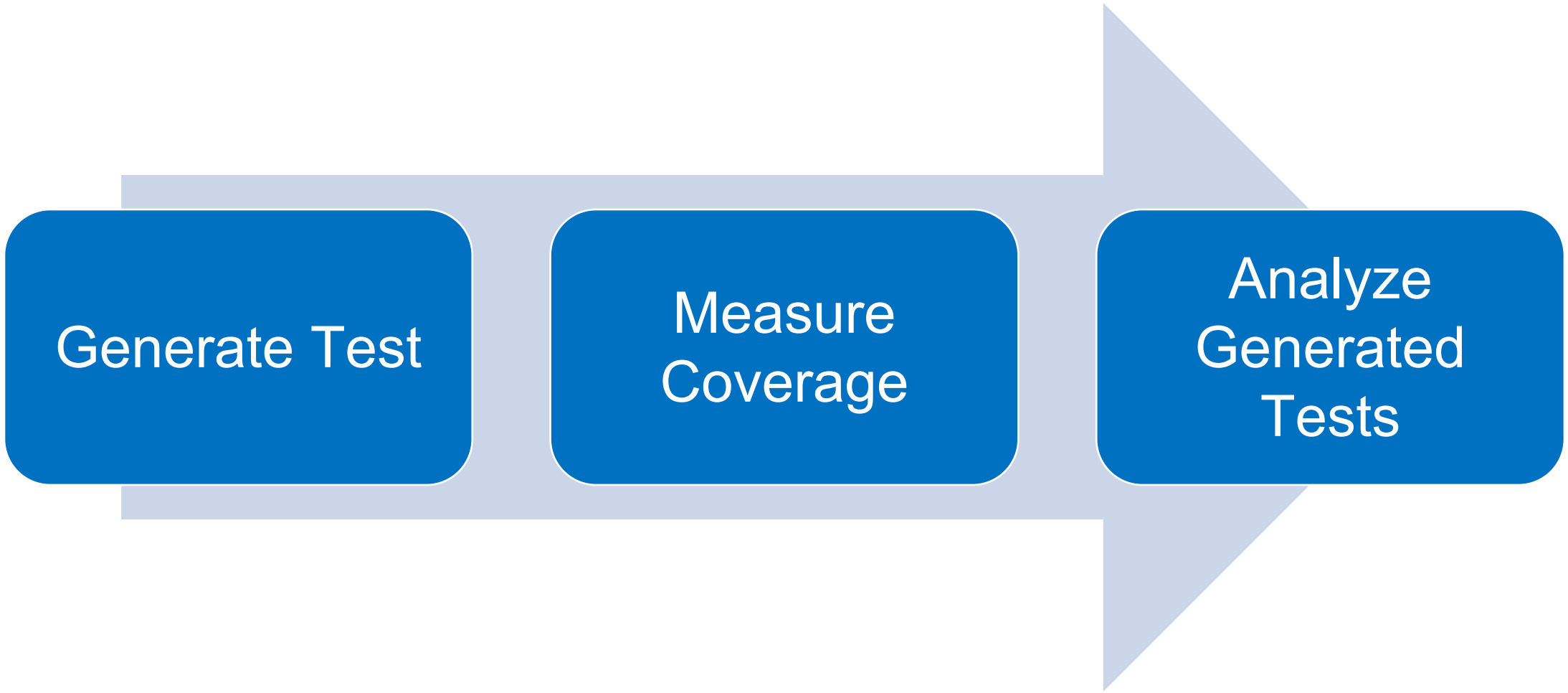
Slides adapted from https://www.mkyong.com/maven/maven-jacoco-code-coverage-example/

Outline

In this lab, you will learn about:

- How to use automated test generation tool for generating Tests
- How to measure test coverage using Jacoco plugin

Overview



Test Coverage

Get the file for the coverage-lab

https://classroom.github.com/a/S8hiiL1J

Setup Needed

- Install Maven in your IDE:
  - Eclipse: https://stackoverflow.com/questions/15633951/how-to-run-the-command-mvn-eclipseeclipse
  - IntelliJ: https://www.jetbrains.com/help/idea/maven-support.html#convert_project_to_maven
    - Maven is already installed in IntelliJ
- Put the file under a package name “cs304.coveragelab”
- Convert the project to a maven project

Setup 2: Modify pom.xml to include JaCoCo

- Declare the following JaCoCo plugin in the **pom.xml** file.

```
<plugin>
        <groupId>org.jacoco</groupId>
        <artifactId>jacoco-maven-plugin</artifactId>
        <version>0.8.2</version>
        <executions>
            <execution>
                <goals>
                    <goal>prepare-agent</goal>
                </goals>
            </execution>
            <!-- attached to Maven test phase -->
            <execution>
                <id>report</id>
                <phase>test</phase>
                <goals>
                    <goal>report</goal>
                </goals>
            </execution>
        </executions>
</plugin>
```

Setup 2: Modify pom.xml to include Evosuite

- Add the lines in red in the **pom.xml** file.

Declare evosuite version as property(similar to variable)

```
<properties>
<evosuiteVersion>1.0.6</evosuiteVersion>
</properties>
```

Evosuite runtime

```
<dependencies>
<dependency>
  <groupId>org.evosuite</groupId>
  <artifactId>evosuite-standalone-runtime</artifactId>
  <version>${evosuiteVersion}</version>
  <scope>test</scope>
</dependency>
<dependencies>
```

Evosuite plugin

```
<plugins>
  <plugin>
    <groupId>org.evosuite.plugins</groupId>
    <artifactId>evosuite-maven-plugin</artifactId>
    <version>${evosuiteVersion}</version>
    <executions><execution>
      <goals> <goal> prepare </goal> </goals>
      <phase> process-test-classes </phase>
    </execution></executions>
  </plugin>
  <plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-surefire-plugin</artifactId>
   <version>2.17</version>
   <configuration>
     <properties>
      <property>
        <name>listener</name>
        <value>org.evosuite.runtime.InitializingListener</value>
     </property>
     </properties>
  </configuration>
</plugin>
</plugins>
</pluginManagement>
```

Read more in: http://www.evosuite.org/documentation/maven-plugin/

Other ways to install Evosuite

- Eclipse
  - Help-> Install New Software-> Update Site: http://www.evosuite.org/update/
  - Intellij: http://www.evosuite.org/documentation/intellij-idea-plugin/
  - Tutorial for IntelliJ:
    - https://github.com/zhiyufan/EvoSuite-IDEA

My pom.xml

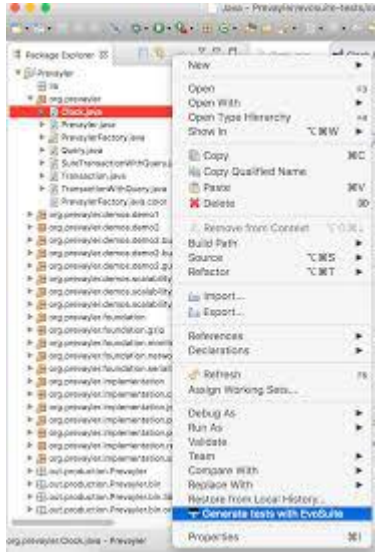
- Right Click->Run Maven clean

MessageBuilder.java | edu.cs304/pom.xml

```
1  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
2    <modelVersion>4.0.0</modelVersion>
3    <groupId>edu.cs304</groupId>
4    <artifactId>edu.cs304</artifactId>
5    <version>0.0.1-SNAPSHOT</version>
6  <properties>
7  <evosuiteVersion>1.0.6</evosuiteVersion>
8  </properties>
9  <dependencies>
10     <dependency>
11   <groupId>org.evosuite</groupId>
12   <artifactId>evosuite-standalone-runtime</artifactId>
13   <version>${evosuiteVersion}</version>
14   <scope>test</scope>
15 </dependency>
16 </dependencies>
17   <build>
18     <sourceDirectory>src</sourceDirectory>
19     <plugins>
20       <plugin>
21         <artifactId>maven-compiler-plugin</artifactId>
22         <version>3.8.0</version>
23         <configuration>
24           <source>1.8</source>
25           <target>1.8</target>
26         </configuration>
27       </plugin>
28       <plugin>
```

Verify if Evosuite is installed correctly

- Try running Evosuite plugin
  - Intellij: http://www.evosuite.org/documentation/intellij-idea-plugin/
  - Eclipse



Generate Test

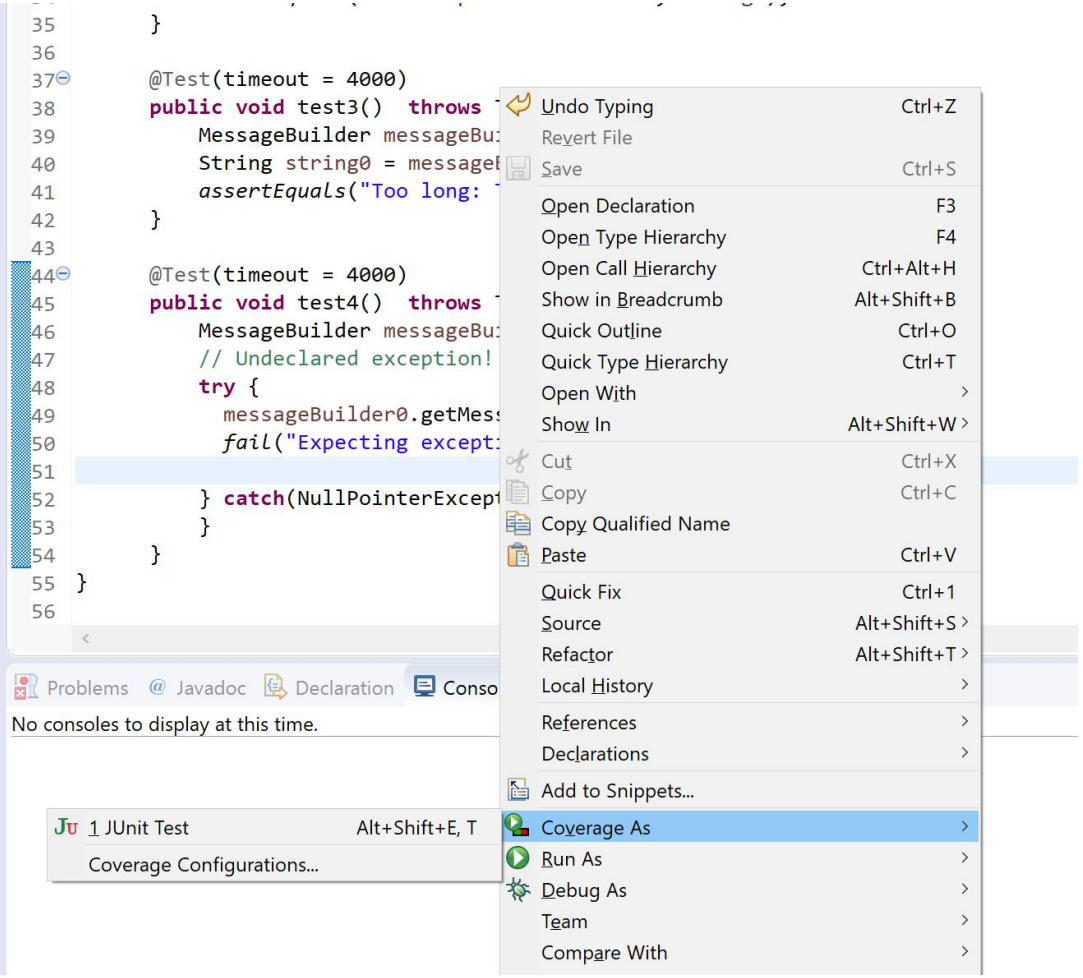
- Generate test using Evosuite for the MessageBuilder class
  - Use the command:
    - mvn -DmemoryInMB=2000 -Dcores=2 evosuite:generate evosuite:export test
  - Or use the “Generate Test” interface

Test Generated!

- If everything worked correctly, then EvoSuite has now produced two files:
  - evosuite-tests/…/MessageBuilder_ESTest.java
    - Contain the real JUnit tests
  - evosuite-tests/tutorial/MessageBuilder_ESTest_scaffolding.java
    - Lots of methods annotated with @Before and @After to ensure that these methods are executed before/after execution of each individual test.
    - The reason for all this is that EvoSuite avoids flaky tests by controlling everything that might be non-deterministic. The scaffolding ensures that tests are always executed in the same consistent state, so they should really only fail if they reveal a bug, not because they are flaky.
    - Could ignore this file

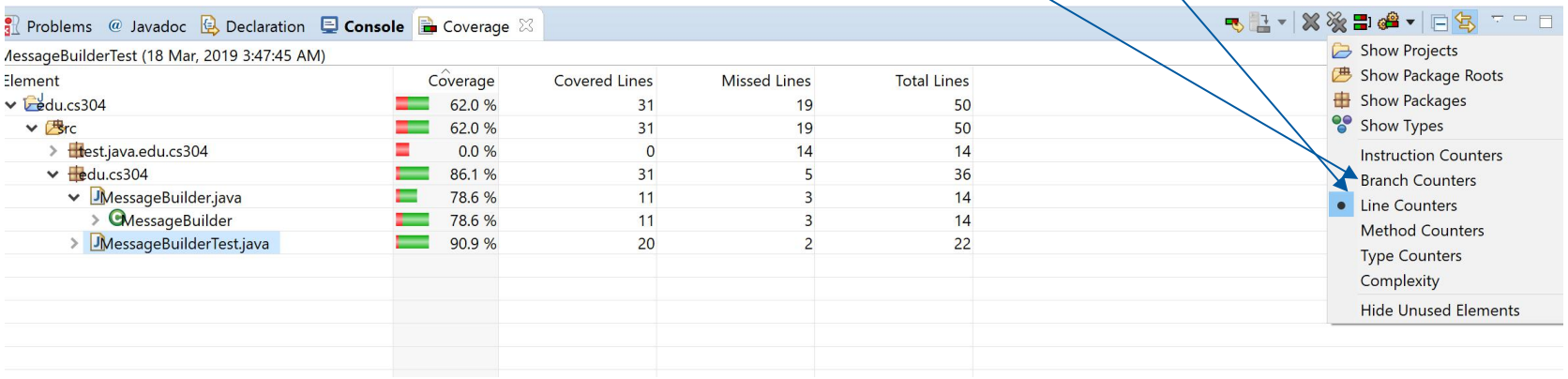
Run Coverage

- Select “Coverage As”-> Junit Test



Measure Coverage

- How many percentage of Line Coverage the automated test achieve?
- How many percentage of Branch Coverage the automated test achieve?



- If the coverage show 0% then you didn't run it correctly. Try creating a new Test class and copy the content of "MessageBuilder_ESTest.java" into the new file

Analyze each generated test

- For each test, answer the following questions:
  - Which line will this test cover?
  - Rename each test to the line where each test cover.
    - For example, if the test cover line 1, 2 and 3. Then change its name to “testL1a2a3”
    - If two tests covered the same lines, then add “-1”,”-2”,”-3” to the name...(e.g.,testL1a2a3-1)
- Could Evosuite achieve 100% line coverage?
- For each uncovered block of code, explain why the tool cannot cover.
- Write more tests to increase the code coverage.

- Put the answers in README.md. Put the JUnit test in TestMessageBuilder.java and commit all the files.

What to submit?

- README.md with **student name, student id, answers,**
- TestMessageBuilder.java

Reminder

- **MP1 part 1 and part 2 has been posted with two different deadlines. Check last week's lab if you have any questions!**
  - Part 1 due on March 24
  - Part 2 due on April 14 (require a few hours of installation and running the tools for 24 hours)
- **Progress Report has been posted due on** April 23
- **All assignments should be written in English**
  - One exception: The selected issues could be written by the developers in Chinese
- **All lab exercise should be submitted before next lab to avoid accumulating too much assignments**