TESTGAP – correct & efficient software tests

Speaker

Marco Schulz ElmarDott
studied at HS Merseburg, Germany, computer science and holds an engineers degree in software engineering. The main topics in his field of work are Software Architectures, automatism of the software development process and Software Configuration Management. Since more than 15 years he work in different large Web Application Projects. Currently he work as independent Consultant and is also writer of many articles in computer magazines.

Joachim Reiter kreativgeist.at
deals since the year 1997 with relational databases, automatic generation of forms, tables, queries and their relations. During his studies at TU Graz he also investigates possibilities to provide a versioning or history of the data inside a Database Management Systems (DBMS). In his role as software architect he is mainly focused on testing ability, maintenance and the reuse of source code.

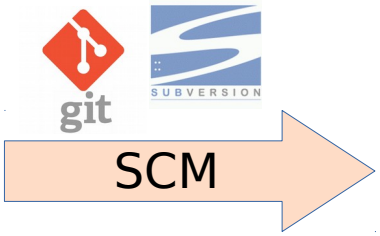
Benefits of automated testing

- ✔ improve quality & correctness
- ✔ formulate exactly defined application behavior
- ✔ avoid re-occurring errors
- ✔ avoid errors after code changes
- ✔ reduce (manual) testing expenses

Strategies, Tools & Stages

0 Stage: local (IDE)

Developer
|Unit test



Build Server (CI)

1 Stage: quality
>nightly build

Build Management
|Integration test

2 Stage: acceptance
>release candidate

Business Division
|Acceptance test

3 Stage: production
>release

End-User
|Patience test

Unit Tests

Function:

```
public T functionToTest(...) {
  return ...;
}
```

AAA-Principle

- ✔ Arrange (setup for test)
- ✔ Act (execute function)
- ✔ Assert (compare result)

Test:

```
public void doSomethingTest() {
  //Arrange
  T result;
  T expected = ...;
  Object o=new Object();
  //Act
  result = o.functionToTest(…);
  //Assert
  assert.equals(expected,result);
}
```

A simple setup: Maven & JUnit

```
<dependencies>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>5.5.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.junit.platform</groupId>
    <artifactId>junit-platform-runner</artifactId>
    <version>1.5.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

BUILD: Maven Surefire Plugin

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>3.0.0-M3</version>
      <configuration>
        <testFailureIgnore>true</testFailureIgnore>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Dream team: NetBeans & Maven

Unit Test Report : Surefire

Surefire Report

Summary

[Summary] [Package List] [Test Cases]

Tests	Errors	Failures	Skipped	Success Rate	Time
191	0	1	5	96.859%	6.776

MailClientScenarioTest

⚠	org.europa.together.service.MailClientScenarioTest + [Detail]	0
	Assumption failed: assumption is not true	
✓	testCalculateSHA512Hashes	0.001
✗	testShrinkContent + [Detail]	0.003
	expected: <true> but was: <false>	
✓	testConcatString	0
✓	testIsNotEmpty	0.001

Unit-tests with JUnit

Function

```
public int power4(int x) {
  int p2;
  p2 = power2(x);
  return power2(p2);
}

private int power2(int x) {
  return x * x;
}
```

Unit Test Case

```
@Test
public void testPower4() {
  int result = Calc.power4(2);
  assertEquals(<?>, result);
}

@Test
public void testPower2() {
  int result = Calc.power2(2);
  assertEquals(<?>, result);
}
```

Coverage levels

$$Coverage = \frac{Reached\ number\ of\ ...}{Total\ number\ of\ ...}$$

- Lines of code
- Statements
- Instructions
- Branches
- Functions

Estimate number of unit tests

Calculate the complexity (McCabe) of a function.

```
public void doSomething (boolean test) {
    if(test) {
        System.out.println('We are testing.');
    } else {
        System.out.println('Nothing to do.');
    }
}
```

Algorithm:
Count the decisions and add 1.

=> complexity is 2
=> 2 test cases should be provided

Test Coverage Report :: JaCoCo

CORE > org.europa.together.application > TreeWalkerImpl Sessions

TreeWalkerImpl

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
prune(TreeNode)		95%		84%	5	17	3	43	0	1
addNode(TreeNode)		100%		78%	3	8	0	14	0	1
toString()		100%		100%	0	2	0	15	0	1
getElementByName(String)		100%		100%	0	5	0	10	0	1
isLeaf(TreeNode)		100%		100%	0	5	0	13	0	1
merge(String, TreeWalker)		100%		100%	0	3	0	10	0	1
getRoot()		100%		90%	1	6	0	9	0	1
addRoot(TreeNode)		100%		100%	0	2	0	9	0	1
getNodeByUuid(String)		100%		83%	1	4	0	9	0	1
getLeafs()		100%		100%	0	3	0	7	0	1
isElementOfTree(TreeNode)		100%		100%	0	4	0	8	0	1
isNameUnique(String)		100%		100%	0	4	0	8	0	1
removeNode(TreeNode)		100%		100%	0	2	0	7	0	1
TreeWalkerImpl(TreeNode)		100%		n/a	0	1	0	5	0	1
TreeWalkerImpl()		100%		n/a	0	1	0	4	0	1
clear()		100%		n/a	0	1	0	4	0	1
static {...}		100%		n/a	0	1	0	1	0	1
isEmpty()		100%		n/a	0	1	0	1	0	1
countNodes()		100%		n/a	0	1	0	1	0	1
getTree()		100%		n/a	0	1	0	1	0	1
Total	8 of 717	98%	10 of 104	90%	10	72	3	179	0	20

Line & Branch coverage in detail

```
207.    @Override
208.    public void addNode(final TreeNode node) {
209.        boolean add = false;
210.        if (!this.tree.isEmpty()
211.                && node != null
212.                && !node.getUuid().equals(this.rootUuid)) {
213.
214.            for (TreeNode check : this.tree) {
215.
216.                if (!node.getUuid().equals(check.getUuid())
217.                        || !node.getParent().equals(check.getParent())
218.                        && !node.getNodeName().equals(check.getNodeName())) {
219.
220.                    add = true;
221.                    LOGGER.log("Node [" + node.getNodeName() + "] added.",
222.                            LogLevel.DEBUG);
223.                    break;
224.
225.                } else {
226.                    LOGGER.log("Node with the same name and parent or the same UUID already exist.",
227.                            LogLevel.WARN);
228.                }
229.            }
230.
231.            if (add) {
232.                tree.add(node);
233.            }
234.        }
235.    }
```

The agile solution

```
207.    @Override
208.    public boolean addNode(final TreeNode node) {
209.        boolean add = true;
210.        if (!this.tree.isEmpty()
211.                && node != null) {
212.
213.            for (TreeNode check : this.tree) {
214.
215.                if (node.getUuid().equals(check.getUuid())
216.                        || node.getParent().equals(check.getParent())
217.                        && node.getNodeName().equals(check.getNodeName())) {
218.
219.                    LOGGER.log("Node with same name AND parent OR the same UUID already exist.",
220.                            LogLevel.WARN);
221.                    add = false;
222.                    break;
223.                }
224.            }
225.
226.            if (add) {
227.                tree.add(node);
228.                LOGGER.log("Node [" + node.getNodeName() + "] added.",
229.                        LogLevel.DEBUG);
230.            }
231.        }
232.        return add;
233.    }
```

What does it mean 100% ?

Calc:

```
float getPPU
  (float price, int items)
{

  return price / items;
}
```

```
@Test
void testGetPPU() {
  int items = 2;
  float price = 10f;
  float exp = 5f;
  float res;

  Calc c = new Calc();
  res = c.getPPU(price, items);

  assertEquals(exp, res, 0.1f);
}
```

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
getPPU(float, int)		100%		n/a	0	1	0	1	0	1
Calc()		100%		n/a	0	1	0	1	0	1
Total	0 of 8	100%	0 of 0	n/a	0	2	0	2	0	2

Repair coverage to death

Calc:

```
float getPPU
  (float price, int items)
{

  if (items == 0) {
    return price;
  }
  else {
    return price / items;
  }
}
```

@Test

```
void testGetPPU() {
  int items = 2;
  float price = 10f;
  float exp = 5f;
  float res;

  Calc c = new Calc();
  res = c.getPPU(price, items);

  assertEquals(exp, res, 0.1f);
}
```

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
getPPU(float, int)		77%		50%	1	2	1	3	0	1
Calc()		100%		n/a	0	1	0	1	0	1
Total	2 of 12	83%	1 of 2	50%	1	3	1	4	0	2

The evil developer

Calc:

```
float getPPU
  (float price, int items)
{

  if (items == 0) {
    return price;
  }
  else {
    return price / items;
  }
}
```

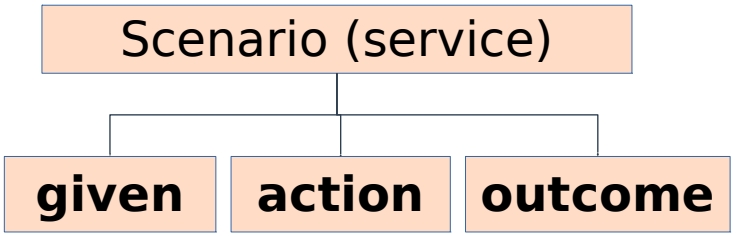
```
@Test
void testGetPPUDunno() {

  Calc c = new Calc();

  c.getPPU(0, 0);
  c.getPPU(1, 1);
}
```

missing assert

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
getPPU(float, int)		100%		100%	0	2	0	3	0	1
Calc()		100%		n/a	0	1	0	1	0	1
Total	0 of 12	100%	0 of 2	100%	0	3	0	4	0	2

Behavioral Driven Development

Arrange → GIVEN (PreCondition)
Act → WHEN (Invariant)
Assert → THEN (PostCondition)

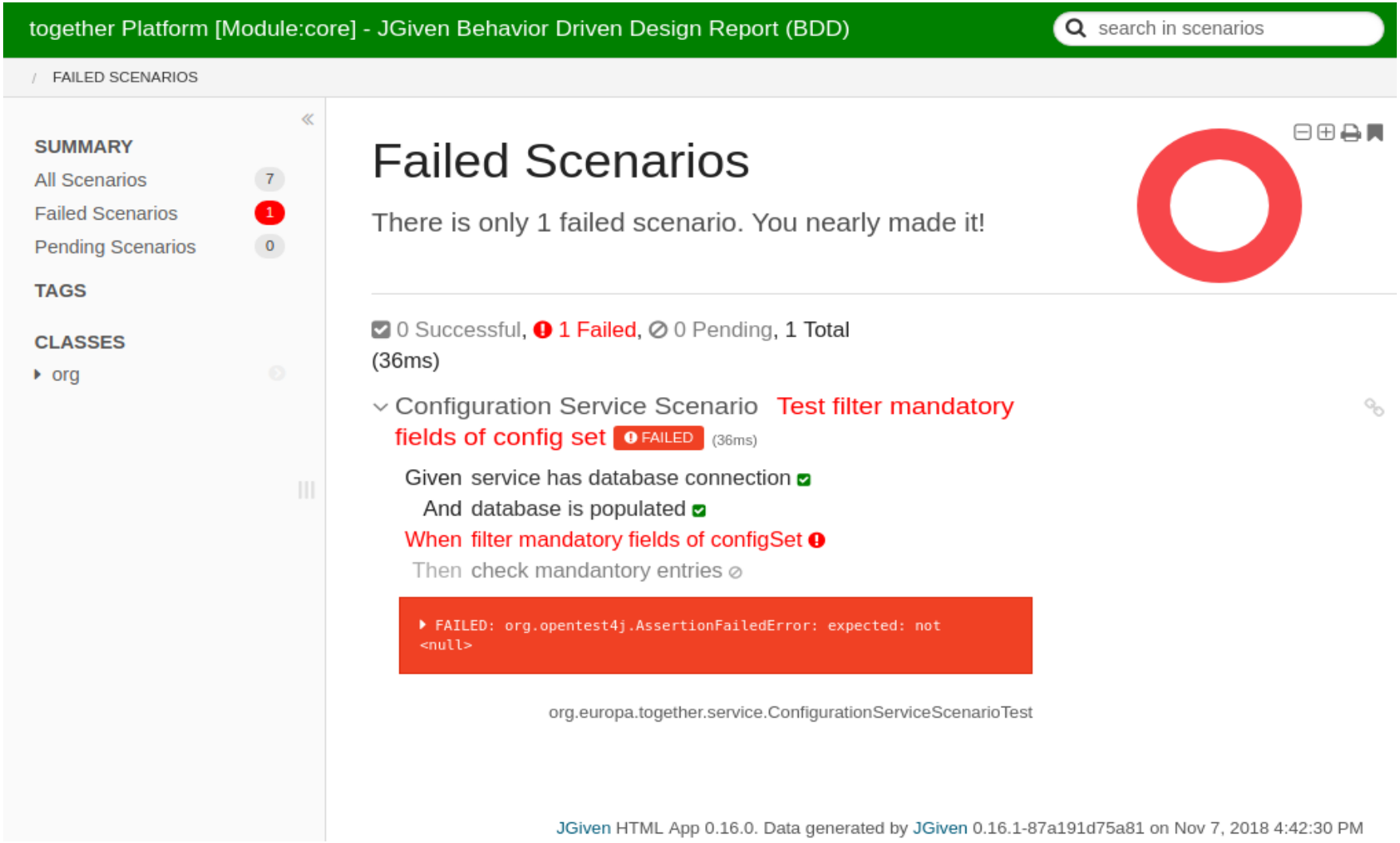


org.europa.together.service.
ConfigurationService

- resetModuleToDefault()
- filterMandatoryFieldsOfConfigSet()

```
public class ConfigurationServiceScenarioTest extends
        ScenarioTest<ConfigurationService Given, ConfigurationService Action,
ConfigurationService Outcome> {

  @Test
  void scenario_testResetModuleToDefault() {

        given(). service_has_database_connection()
              .and().database_is_populated();
        when().reset_module_to_default();
        then().check_default_entries();
```

BDD Acceptance Test Report :: JGiven



Test Expectations

> **Tests do not prove that a system is free of failures. They show or demonstrate the behavior of a system in defined conditions.**

- Unit tests are regression tests
- KISS Principle: **K**eep **i**t **s**imple, **s**tupid.
- AAA Principle: **A**rrange, **A**ct, **A**ssert
- Logging is always your friend
- Industrial standard for coverage is ~80-85%

Resources

[1] Personal Blog (Marco): https://enrebaja.wordpress.com
[2] Personal Blog (Joachim): https://kreativgeist.at
[3] Project TP-CORE: https://github.com/ElmarDott/TP-CORE
[4] Presentation: https://www.slideshare.net/elmardott/
[5] Youtube Channel: https://www.youtube.com/channel/UCBdJ0zh8xnMrQxQ4Gymy2Q

JavaMagazin 2.2019 S. 88-92
Testfalle: Eine Einführung in TDD und BDD

[A] https://dzone.com/articles/embrace-junit5
[B] https://www.javacodegeeks.com/2019/07/regression-testing-tools-techniques.html
[C] https://www.javacodegeeks.com/2019/07/tdd-misbeliefs.html
[D] https://www.yegor256.com/2018/12/11/unit-testing-anti-patterns.html
[E] https://blog.cleancoder.com/uncle-bob/2014/04/30/When-tdd-does-not-work.html
[F] https://arstechnica.com/information-technology/2013/03/how-can-i-Motivate-coworkers-to-write-unit-tests/