32. Agiler Stammtisch Frankfurt

Testing-Darwinismus

Code Defekte mit Mutation Testing aufdecken

Qaiser Abbasi

codecentric

Today...

- Who is who?
- Mutation Testing
- Lean Coffee Panel
- Pizza :)
- Feedback

A long time ago in a galaxy far, far away...

The untested side effect

```
public static String foo(boolean b) {
  if ( b ) {
    performVitallyImportantBusinessFunction();
    return "OK";
  }

  return "FAIL";
}

@Test
public void shouldFailWhenGivenFalse() {
  assertEquals("FAIL", foo(false));
}

@Test
public void shouldBeOkWhenGivenTrue() {
  assertEquals("OK", foo(true));
}
```

The missing boundary test

```
public static String foo(int i) {
  if ( i >= 0 ) {
      return "foo";
  } else {
      return "bar";
  }
}

@Test
public void shouldReturnBarWhenGiven1() {
  assertEquals("bar", foo(1));
}

@Test
public void shouldReturnFooWhenGivenMinus1() {
  assertEquals("foo", foo(-1));
}
```

The myopic mockist

```
public static String foo(Collaborator c, boolean b) {
  if ( b ) {
      return c.performAction();
  }

  return "FOO";
}

@Test
public void shouldPerformActionWhenGivenTrue() {
  foo(mockCollaborator, true);
  verify(mockCollaborator).performAction();
}

@Test
public void shouldNotPerformActionWhenGivenFalse() {
  foo(mockCollaborator, false);
  verify(never(), mockCollaborator).performAction();
}
```

Try to describe the very nature of the problem here

How do we ensure that our test suite copes well with the relevant stuff?

Mutation Testing

Idea: Mutate the code and the tests should *fail*

Pitest – „Real world mutation testing“

pitest.org

- Most popular framework for Java
- Active development & support
- Works with Ant, Maven, Gradle and others
- Human-readable reports containing line and mutation coverage information

Mutators

Apply mutation operators to the byte code

Conditionals Boundary Mutator

```
if ( i >= 0 ) {
    return "foo";
} else {
    return "bar";
}
```

mutated into

```
if ( i > 0 ) {
    return "foo";
} else {
    return "bar";
}
```

Mutants

Java classes containing the injected fault

Available Mutators

http://pitest.org/quickstart/mutators/

- Conditionals Boundary Mutator
- Increments Mutator
- Invert Negatives Mutator
- Math Mutator
- Return Values Mutator
- Void Method Calls Mutator
- ...

Return Values Mutator

```
public Object foo() {
  return new Object();
}
```

mutated into

```
public Object foo() {
  new Object();
  return null;
}
```

Void Method Calls Mutator

```
public void someVoidMethod(int i) {
  // serious business logic
}

public int foo() {
  int i = 5;
  doSomething(i);
  return i;
}
```

mutated into

```
public void someVoidMethod(int i) {
  // serious business logic
}

public int foo() {
  int i = 5;
  return i;
}
```

Running The Tests

Attack the suspicious ones!

Summing up

- Traditional test coverage only addresses which code is executed
- By contrast, Mutation testing detects whether the code is meaningfully tested

Thank you for your attention!

Any questions left?

Slides @github.com/qabbasi