Reference : http://blog.caplin.com/2012/07/23/mutation-testing-killed-or-survived/

Example

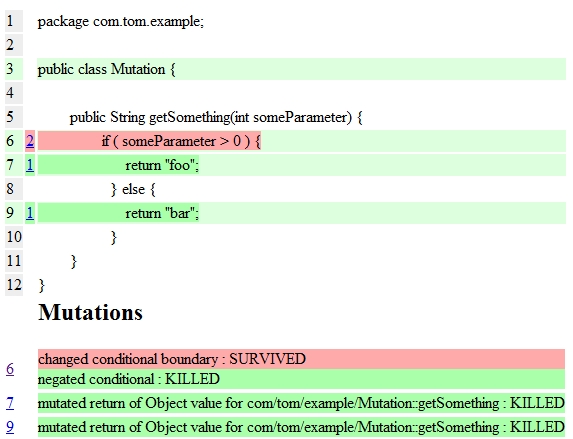
Let’s have a look at a simple example I have prepared using [**PIT Mutation Testing.**](http://pitest.org/)

public String getSomething(int someParameter) {  
if ( someParameter > 0 ) {  
return "foo";  
} else {  
return "bar";  
}  
}  
  
  
@Test  
public void testOne() {  
assertEquals("foo", mutationClass.getSomething(1));  
}

@Test  
public void testMinusOne() {  
assertEquals(“bar”, mutationClass.getSomething(-1));  
}

If we now run the unit tests we will get a line and branch coverage of 100%.

However if we run the tests with PIT we can see that there is a test that survives a mutation.

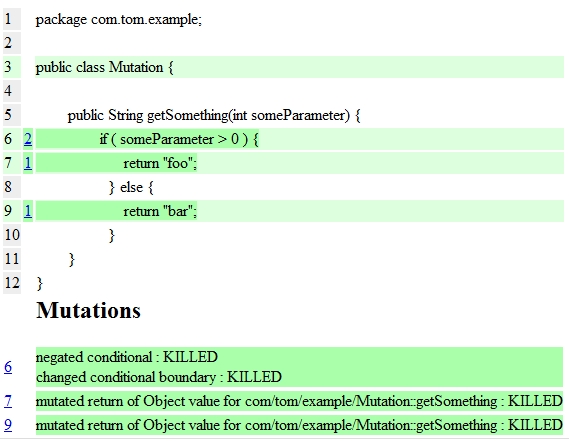


From the result we can conclude the following. PIT mutated the code and changed the if-statement to “someParameter >= 0″. As our tests only test 1 and -1 we are missing to test the value 0.

So let’s add another test.

@Test  
public void testZero() {  
assertEquals("bar", mutationClass.getSomething(0));  
}

The result of the second run is:



Rather than adding another test I could have just replaced the -1 with 0.

Conclusion

I think mutation testing is an interesting technique to explore the behaviour of the application under test and the existing tests in connection with small code changes.  Whether you do it manually by changing some statements and variables or with an existing tool. Sometimes a developer makes a tiny change by replacing  ”>” with”>=”, runs the tests and everything passes although the tests might not be designed for this change.

A line and branch coverage of 100% does not say anything about the quality of the tests. Getting feedback about the quality of the tests in addition to the coverage is absolutely vital. The example shows that choosing arbitrary values will result in passing tests but with a risk to future changes. Applying a [**boundary-value analysis**](http://en.wikipedia.org/wiki/Boundary-value_analysis) for example would be a first step to improve the quality of the tests. And that is exactly what the mutated code illustrated.