Are automatically generated test suites “good”?

Simon Larsén Philippa Örnell

March 24, 2019

1. What makes a good test suite?

2. Automated test generation

3. Summary and references

Section 1

What makes a good test suite?

Code coverage

- Purpose
  - Check which parts of production code have been executed
- Pros
  - Automated
  - Fast
  - Percentages are nice
  - Code not covered $\implies$ code not tested
- Cons
  - Code covered $\not\Leftarrow$ code tested
  - False sense of security?

Mutation testing

- Purpose
  - Check if tests observe incorrect states
- Pros
  - Indicates what has not been tested properly
  - Automated
- Cons
  - Traditional mutation testing takes a *long* time [1]
  - Equivalent mutants (8-9% [2], [3])

Maintainability

- Performance measures exclusively focused on *now*
- Software maintenance costs typically exceed 50% of total development cost [4]
- Test maintenance can be more costly than production code maintenance [5], [6]
- Performance now $\not\Rightarrow$ performance tomorrow
  - ABB test suite started at 90% coverage
  - Ten years later: 10% coverage, rarely even run [7]

A maintainable test case

- DevOps is heavily focused around software as a living thing

> *"[. . .], a good test case should not only be sensitive to deviations from the intended behavior, but should also be maintainable in its own right;* ***it should be easy to understand so that it can be readily adapted to changes in the rest of the code base as it evolves.****" [8]*

Section 2

Automated test generation

Fibo.java

```java
public class Fibo {
    private long current;
    private long next;

    public Fibo() {
        current = 0;
        next = 1;
    }

    public long next() {
        long previous = current;
        current = next;
        next = previous + current;
        return previous;
    }
}
```

EVOSUITE generated test

```
@Test(timeout = 4000)
public void test0()  throws Throwable  {
  Fibo fibo0 = new Fibo();
  long long0 = fibo0.next();
  assertEquals(0L, long0);

  long long1 = fibo0.next();
  assertEquals(1L, long1);
}
```

Clean test?

1. Tests one thing?
2. Good test name?
3. Clear structure (e.g. AAA)?

Not a very good test suite

- Assumes current implementation is correct
  - To us, testing should be about contesting correctness
- Test scores high on performance (full coverage, 71% mutation score)
  - But would pass a function generating 0, 1, 2. . .
- Test has no obvious purpose
  - Harder for human testers to understand and thus maintain [8]

Section 3

Summary and references

Takeaways

- Performance is *hard* to measure in a general way
- High performance $\not\Rightarrow$ good test suite
  - Maintainability is also important
- AGTs do what they are designed to do well
  - But maybe white-box performance criteria is insufficient

References I

[1] R. Niedermayr, E. Juergens, and S. Wagner, "Will my tests tell me if i break this code?" in *Proceedings of the international workshop on continuous software evolution and delivery*, 2016, pp. 23–29.

[2] A. J. Offutt and J. Pan, "Automatically detecting equivalent mutants and infeasible paths," *Software testing, verification and reliability*, vol. 7, no. 3, pp. 165–192, 1997.

[3] M. Bybro and S. Arnborg, "A mutation testing tool for java programs," *Master's thesis, Stockholm University, Stockholm, Sweden*, 2003.

[4] B. Hunt, B. Turner, and K. McRitchie, "Software maintenance implications on cost and schedule," in *2008 ieee aerospace conference*, 2008, pp. 1–6.

References II

[5] A. Labuschagne, L. Inozemtseva, and R. Holmes, “Measuring the cost of regression testing in practice: A study of java projects using continuous integration,” in *Proceedings of the 2017 11th joint meeting on foundations of software engineering*, 2017, pp. 821–830.

[6] E. Alégroth, R. Feldt, and P. Kolström, “Maintenance of automated test suites in industry: An empirical study on visual gui testing,” *Information and Software Technology*, vol. 73, pp. 66–80, 2016.

[7] B. Robinson, M. D. Ernst, J. H. Perkins, V. Augustine, and N. Li, “Scaling up automated test generation: Automatically generating maintainable regression unit tests for programs,” in *2011 26th ieee/acm international conference on automated software engineering (ase 2011)*, 2011, pp. 23–32.

References III

[8] S. Shamshiri, J. M. Rojas, J. P. Galeotti, N. Walkinshaw, and G. Fraser, “How do automatically generated unit tests influence software maintenance?” in *2018 ieee 11th international conference on software testing, verification and validation (icst)*, 2018, pp. 250–261.