The Interplay Between Automatic Generation and Human Exploration in Developer-Centric Test Amplification

Carolin Brandt
Delft University of Technology
c.e.brandt@tudelft.nl

***Abstract*—This presentation introduces our explorative study on generating amplified test cases that developers can take over right into their maintained test suite.**

***Index Terms*—Software Testing, Test Amplification, Test Exploration, Test Generation, Developer-Centric Design**

I. Introduction

Software testing is one of the central practices of software engineering. When checking whether our software is behaving in the way we intend, our first line of attack is *developer testing*: Scripts that developers write to validate the behavior of the code they write [1]. Many developers test their software less than they would want to because it requires a high effort with little direct value, keeping them from developing new features. To ease the burden of testing, automatically generating test cases for software has been an active research topic for many years. While current tools can generate powerful regression or crash-reproducing test cases, these tests are often kept separately from the maintained test suite as they are not up to par with manually written test cases. Furthermore, they are often inadequate for key use cases of developer tests such as documentation [2] or fault localization [3]. To truly alleviate developer's efforts of writing test cases, I conjecture that we need to generate test cases that developers accept and add into their codebase as if they wrote the tests themselves. I hypothesize, that, if we provide valuable test cases to developers, we can also start asking for something in return. The developer's knowledge about the software and its domain could help our test generation tools with aspects that are to date hard to solve, such as the oracle problem or prioritization. This is why I am working on automatically generating test cases for and with developers.

In this presentation, I will present the first study of my PhD [4], in which we develop a prototypical test amplification tool that is centered around software developers. We extend the state-of-the-art test amplification tool DSpot [5] to generate shorter, easier to understand test cases. To facilitate the interaction between the developer and the automatic test amplification, we create the test exploration tool and IntelliJ plugin *TestCube*[1]. Figure 1 introduces the interaction we envision. With these prototypes, we investigate the important aspects and open challenges of developer-centric test amplification in 16 semi-structured interviews with software developers. From our interviews, we gather 52 observations that we summarize into 23 result categories and give two key recommendations on how future tool designers can make their tools better suited for developer-centric test amplification. My presentation will give an overview of our study and discuss selected results that inspire our research efforts today and in the coming years.

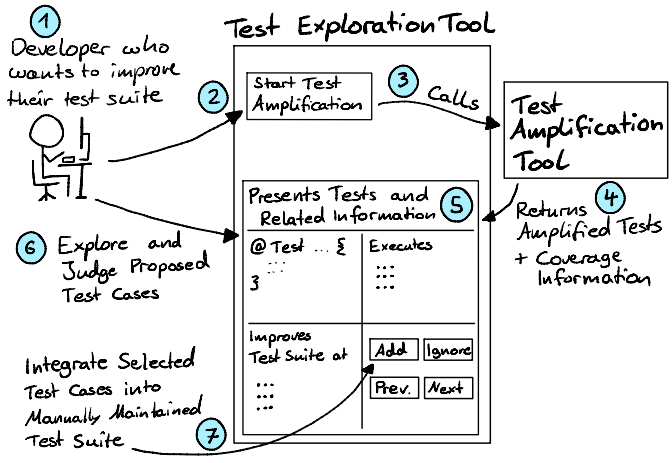

Fig. 1. Overview of test amplification with the help of a test exploration tool.

This research was funded by the Dutch science foundation NWO through the Vici "TestShift" grant (No. VI.C.182.032)

[1] https://github.com/TestShiftProject/test-cube

References

[1] G. Meszaros, *XUnit Test Patterns: Refactoring Test Code*. Pearson Education, 2007.
[2] K. L. Beck, *Test-Driven Development - By Example*, ser. The Addison-Wesley signature series. Addison-Wesley, 2003.
[3] S. Panichella, A. Panichella, M. Beller, A. Zaidman, and H. C. Gall, "The impact of test case summaries on bug fixing performance: An empirical investigation," in *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*. ACM, 2016, pp. 547–558.
[4] C. Brandt and A. Zaidman, "Developer-centric test amplification: The interplay between automatic generation and human exploration," 2021.
[5] B. Danglot, O. L. Vera-Pérez, B. Baudry, and M. Monperrus, "Automatic test improvement with DSpot: A study with ten mature open-source projects," *Empirical Software Engineering*, vol. 24, no. 4, pp. 2603–2635, 2019.