Presentation Abstract: Generating Class Integration Tests Using Call Site Information

Pouria Derakhshanfar, Xavier Devroey, Annibale Panichella, Andy Zaidman and Arie van Deursen
Delft University of Technology, Delft, The Netherlands. Emails: p.derakhshanfar@tudelft.nl,
x.d.m.devroey@tudelft.nl, a.panichella@tudelft.nl, a.e.zaidman@tudelft.nl and arie.vandeursen@tudelft.nl

Search-based approaches have been applied to a variety of white-box testing activities [1], among which test case and data generation [2]. In white-box testing, most of the existing work has focused on the unit level, where the goal is to generate test cases/suites that achieve high structural (e.g., branch) coverage. Prior work has shown that search-based unit test generation can achieve high code coverage [3]–[5], detect real-bugs [6], [7], and help developers during debugging activities [8].

Despite these undeniable advantages, in recent years, researchers have investigated the imitations of the generated unit tests [7], [9], [10]. Prior studies have questioned the effectiveness of the generated unit tests with high code coverage in terms of their capability to detect real faults or to kill mutants when using mutation coverage. For example, Gay *et al.* [9] have highlighted how traditional code coverage could be a poor indicator of test effectiveness (in terms of fault detection rate and mutation score). In particular, traditional unit-level adequacy criteria measure only whether certain code elements are reached, but not *how* each element is covered. The quality of the test data and the paths from the covered element to the assertion play an essential role for better test effectiveness. As such, they have advocated the need for more reliable adequacy criteria for test case generation tools.

In this presentation, we discuss how the integration code between coupled classes can be used as guidance for the test generation process. The idea is that, by exercising the behavior of a class under test *E* (the calleE) through another class *R* (the calleR) calling its methods, *R* will handle the creation of complex parameter values and exercise valid usages of *E*. In order words, the caller R might contain integration code that (1) enables to create better test data for the callee E, and (2) allows to better validate the data returned by E.

Integration testing can be approached from many different angles [11], [12]. In our case, we focus on *class integration testing* between a caller and a callee [13]. Our idea is to complement unit test generation for a class under test by looking at its integration with other classes. To that end, we define a novel structural adequacy criterion we call *Coupled Branches Coverage* (CBC), targeting specific integration points between two classes. Coupled branches are pairs of branches $\langle r, e \rangle$, with r a branch of the caller, and e a branch of the callee, such that an integration test that exercises branch r also exercises branch e. Furthermore, we devise a search-based approach that generates integration-level test suites, based on the CBC criterion. We coin our approach CLING (for class integration testing).

According to our preliminary results, on average, CLING allows killing 10% of mutants per class that cannot be detected by unit tests generated with EVOSUITE for both the caller and the callee. The improvements in mutation score are even up to 60% for certain classes.

This research was partially funded by the EU Project STAMP ICT-16-10 No.731529.

REFERENCES

[1] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering," *ACM Computing Surveys*, vol. 45, no. 1, pp. 1–61, nov 2012.
[2] P. McMinn, "Search-based software test data generation: A survey," *STVR*, vol. 14, no. 2, pp. 105–156, 2004.
[3] M. M. Almasi, H. Hemmati, G. Fraser, A. Arcuri, and J. Benefelds, "An Industrial Evaluation of Unit Test Generation: Finding Real Faults in a Financial Application," in *ICSE-SEIP '17*. IEEE, may 2017, pp. 263–272.
[4] J. Campos, Y. Ge, G. Fraser, M. Eler, and A. Arcuri, "An empirical evaluation of evolutionary algorithms for test suite generation," in *SSBSE '17*, ser. LNCS, T. Menzies and J. Petke, Eds., vol. 10452. Springer, 2017, pp. 33–48.
[5] A. Panichella, F. M. Kifetew, and P. Tonella, "A large scale empirical comparison of state-of-the-art search-based test case generators," *IST*, vol. 104, no. June, pp. 236–256, 2018.
[6] G. Fraser and A. Arcuri, "1600 faults in 100 projects: automatically finding faults while achieving high coverage with evosuite," *EMSE*, vol. 20, no. 3, pp. 611–639, 2015.
[7] S. Shamshiri, R. Just, J. M. Rojas, G. Fraser, P. McMinn, and A. Arcuri, "Do automatically generated unit tests find real faults? An empirical study of effectiveness and challenges," *ASE '15*, pp. 201–211, 2016.
[8] M. Ceccato, A. Marchetto, L. Mariani, C. D. Nguyen, and P. Tonella, "Do automatically generated test cases make debugging easier? an experimental assessment of debugging effectiveness and efficiency," *TOSEM*, vol. 25, no. 1, pp. 5:1–5:38, Dec. 2015.
[9] G. Gay, M. Staats, M. Whalen, and M. P. Heimdahl, "The risks of coverage-directed test case generation," *TSE*, vol. 41, no. 8, pp. 803–819, 2015.
[10] A. Schwartz, D. Puckett, Y. Meng, and G. Gay, "Investigating faults missed by test suites achieving high code coverage," *JSS*, vol. 144, pp. 106–120, 2018.
[11] Z. Jin and A. J. Offutt, "Coupling-based criteria for integration testing," *STVR*, vol. 8, no. 3, pp. 133–154, sep 1998.
[12] A. Offutt, A. Abdurazik, and R. Alexander, "An analysis tool for coupling-based integration testing," in *ICECCS '00*. IEEE, 2000, pp. 172–178.
[13] A. Scott, "Building object applications that work, your step-by-step handbook for developing robust systems using object technology," 1997.