1. Revisiting test cases to boost generate-and-validate program repair

Jingtang Zhang (1); Kui Liu (1, 2); Dongsun Kim (3); Li Li (4); Zhe Liu (1); Klein, J. (5); Bissyande, T.F. (5)
Source: *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, p 35-46, 2021;
ISBN-13: 978-1-6654-2882-8; **DOI:** 10.1109/ICSME52107.2021.00010; **Conference:** 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), 27 Sept.-1 Oct. 2021, Luxembourg City, Luxembourg;
Publisher: IEEE, Piscataway, NJ, USA
Author affiliation: (1) Nanjing University of Aeronautics and Astronautics, China (2) State Key laboratory of Mathematical Engineering and Advanced Computing, State Key Laboratory of Mathematical Engineering and Advanced Computing, China (3) Kyungpook National University, Korea, Republic of (4) Monash University, Melbourne, VIC, Australia (5) University of Luxembourg, Interdisciplinary Centre for Security, Reliability and Trust, Luxembourg
Abstract: Fault localization produces bug positions as the basic input for many automated program repair (APR) systems. Given that test cases are the common means that automatic fault localization techniques leverage, we investigate the impact of their characteristics (in terms of quality and quantity) on APR. In particular, we analyze the statements that appear in crash stack traces when test cases fail (note that stack traces are available when an ordinary test case fails since its verdict is often made by assertions that produce errors such as AssertError in Java and JUnit), and explore the possibility of using some relevant crash information to enhance fault localization; this ultimately improves the effectiveness of APR tools. Our study reveals that the considered state-of-the-art APR systems achieve the best performance when fixing bugs associated with boolean type expected values (e.g., assertTrue ()) or assertFalse(). In contrast, they achieve their worst performance when addressing bugs related to null check assertions. Meanwhile, null check bugs as well as the bugs associated with boolean and string type expected values are still the main challenge that should be addressed by the future APR. For exception throwing bugs, existing APR systems present the best performance on fixing NullPointerException bugs, while the tough task of them is to resolve the bugs throwing developer-defined exceptions. The information in stack traces after executing the bug-triggering test cases can be used to effectively improve the performance on fault localization and program repair. (0 refs)
Inspec controlled terms: Java - program debugging - program diagnostics - program testing - public domain software - software fault tolerance - software maintenance
Uncontrolled terms: APR tools - state-of-the-art APR systems - null check assertions - null check bugs - boolean - string type expected values - future APR - exception throwing bugs - stack traces - bug-triggering test cases - generate-and-validate program repair - bug positions - automated program repair systems - automatic fault localization techniques leverage - crash stack - ordinary test case - relevant crash information - NullPointerException bugs
Classification Code: C6150G Diagnostic, testing, debugging and evaluating systems - C6110B Software engineering techniques - C6110J Object-oriented programming
IPC Code: G06F9/44 - G06F11/36
Treatment: Practical (PRA)
Database: Inspec
Data Provider: Engineering Village