The Journal of Open Source Software

EvoMaster: A Search-Based System Test Generation Tool

Andrea Arcuri[1], **Juan Pablo Galeotti**[2], **Bogdan Marculescu**[1], **and Man Zhang**[1]

1 Kristiania University College, Department of Technology, Oslo, Norway **2** FCEyN-UBA, and ICC, CONICET-UBA, Depto. de Computaci'on, Buenos Aires, Argentina

DOI: 10.21105/joss.02153

Software

- Review
- Repository
- Archive

Editor: George K. Thiruvathukal

Reviewers:

- @mado89
- @s0nata
- @UTH-Tuan

Submitted: 07 January 2020
Published: 03 January 2021

Statement of Need

Testing web/enterprise applications is complex and expensive when done manually. Often, software testing takes up to half of the development time and cost for a system. So much testing is needed because the cost of software failure is simply too large: for example, in 2017, 304 software failures (reported in the media) impacted 3.6 billion people and $1.7 trillion in assets worldwide (Tricentis, 2017). Unfortunately, due to its high cost, software testing is often left incomplete, and only applied partially.

To address this problem, in *Software Engineering* (SE) research a lot of effort has been spent in trying to design and implement novel techniques aimed at automating several different tasks, where software testing is among the most studied tasks. *Search-Based Software Testing* (SBST) (Harman et al., 2012) casts the problem of software testing as an optimization problem, aimed at, for example, maximizing code coverage and fault detection.

Our SBST tool called `EvoMaster` addresses these challenges by using evolutionary techniques to automatically generate test cases. It currently focuses on RESTful web services, which are the pillars of modern web and enterprise applications (Fielding, 2000)(Allamaraju, 2010).

The `EvoMaster` tool is aimed at:

- practitioners in industry that want to automatically test their software.
- researchers that need generated test cases for their studies.

Tool Summary

`EvoMaster` (Arcuri, 2018a) is a SBST tool that automatically *generates* system-level test cases. Internally, it uses an *Evolutionary Algorithm* and *Dynamic Program Analysis* to be able to generate effective test cases. The approach is to *evolve* test cases from an initial population of random ones, using code coverage and fault detection as fitness function.

Key features:

- At the moment, *EvoMaster* targets RESTful APIs compiled to JVM **8** and **11** bytecode.
- The APIs must provide a schema in *OpenAPI/Swagger* format (either *v2* or *v3*).
- The tool generates JUnit (version 4 or 5) tests, written in either Java or Kotlin.

- Fault detection: *EvoMaster* can generate tests cases that reveal faults/bugs in the tested applications. Different heuristics are employed, like checking for 500 status codes and mismatches from the API schemas.
- Self-contained tests: the generated tests do start/stop the application, binding to an ephemeral port. This means that the generated tests can be used for *regression testing* (e.g., added to the *Git* repository of the application, and run with any build tool such as *Maven* and *Gradle*).
- Advanced *whitebox* heuristics: *EvoMaster* analyses the bytecode of the tested applications, and uses several heuristics such as *testability transformations* and *taint analysis* to be able to generate more effective test cases.
- SQL handling: *EvoMaster* can intercept and analyse all communications done with SQL databases, and use such information to generate higher code coverage test cases. Furthermore, it can generate data directly into the databases, and have such initialization automatically added in the generated tests. At the moment, *EvoMaster* supports *H2* and *Postgres* databases.
- *Blackbox* testing mode: can run on any API (regardless of its programming language), as long as an *OpenAPI* schema is provided. However, results will be worse than *whitebox* testing (e.g., due to lack of bytecode analysis).

Published Results

When addressing the testing of real-world web/enterprise applications, there are many challenges. The tested code can for example have complex execution flows, where the boolean predicates in *if* and *loop* statements depend on specific input data. Furthermore, the execution flow could depend on interactions with external entities, such as databases, GUIs and remote web services. The search space of all possible test inputs is huge, where only a tiny subset lead to maximize code coverage and detect faults.

To face and overcome those challenges, *EvoMaster* has been used to experiment with several novel techniques. These techniques are now integrated in *EvoMaster*, where their best settings (based on empirical studies) are on by default.

This research work led to several publications: novel search algorithms such as *MIO* (Arcuri, 2017a)(Arcuri, 2018b), addressing the white-box testing of RESTful APIs (Arcuri, 2017b)(Arcuri, 2019), resource-dependency handling (Zhang et al., 2019), accesses to SQL databases (Arcuri & Galeotti, 2019), and novel *testability transformations* (Arcuri & Galeotti, 2020).

Related Work

In the recent years, different techniques have been proposed for *black-box* testing of RESTful APIs (e.g., (Atlidakis et al., 2019)(Karlsson et al., 2020) (Viglianisi et al., 2020)(Ed-douibi et al., 2018)). Those present different variants of random testing, enhanced with heuristics based on the information provided in the API schemas. As those techniques are black-box, they do not access the source-code/bytecode of the tested APIs, and so cannot exploit any such information to improve the generation of test cases.

At the time of this writing, *EvoMaster* appears to be the only tool that can do both *black-box* and *white-box* testing, and that is also released as open-source. Supporting black-box testing is important, as it is easier to setup and use. However, white-box testing leads to better

results (e.g., higher code coverage and fault detection), as it can exploit more information on the system under test.

Acknowledgements

We thank Annibale Panichella for providing a review and fix of our implementation of his MOSA algorithm. We also want to thank Agustina Aldasoro for her contributed bug fixes. This work is funded by the Research Council of Norway (project on Evolutionary Enterprise Testing, grant agreement No 274385), and partially by UBACYT-2018 20020170200249BA, PICT-2015-2741.

References

Allamaraju, S. (2010). *Restful web services cookbook: Solutions for improving scalability and simplicity*. " O'Reilly Media, Inc.".

Arcuri, A. (2018a). EvoMaster: Evolutionary Multi-context Automated System Test Generation. *IEEE International Conference on Software Testing, Verification and Validation (ICST)*, 394–397. https://doi.org/10.1109/icst.2018.00046

Arcuri, A. (2017a). Many Independent Objective (MIO) Algorithm for Test Suite Generation. *International Symposium on Search Based Software Engineering (SSBSE)*, 3–17. https://doi.org/10.1007/978-3-319-66299-2_1

Arcuri, A. (2017b). RESTful API Automated Test Case Generation. *IEEE International Conference on Software Quality, Reliability and Security (QRS)*, 9–20. https://doi.org/10.1109/qrs.2017.11

Arcuri, A. (2018b). Test suite generation with the Many Independent Objective (MIO) algorithm. *Information and Software Technology*, *104*, 195–206. https://doi.org/10.1016/j.infsof.2018.05.003

Arcuri, A. (2019). RESTful API Automated Test Case Generation with EvoMaster. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *28*(1), 3. https://doi.org/10.1145/3293455

Arcuri, A., & Galeotti, J. P. (2019). SQL data generation to enhance Search-Based System Testing. *Genetic and Evolutionary Computation Conference (GECCO)*, 1390–1398. https://doi.org/10.1145/3321707.3321732

Arcuri, A., & Galeotti, J. P. (2020). Testability Transformations For Existing APIs. *IEEE International Conference on Software Testing, Verification and Validation (ICST)*. https://doi.org/10.1109/ICST46399.2020.00025

Atlidakis, V., Godefroid, P., & Polishchuk, M. (2019). RESTler: Stateful REST API Fuzzing. *Proceedings of the 41st International Conference on Software Engineering*, 748–758. https://doi.org/10.1109/ICSE.2019.00083

Ed-douibi, H., Cánovas Izquierdo, J. L., & Cabot, J. (2018). Automatic Generation of Test Cases for REST APIs: A Specification-Based Approach. *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*, 181–190. https://doi.org/10.1109/EDOC.2018.00031

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* [PhD thesis]. University of California, Irvine.

Harman, M., Mansouri, S. A., & Zhang, Y. (2012). Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys (CSUR)*, *45*(1), 11. https://doi.org/10.1145/2379776.2379787

Karlsson, S., Causevic, A., & Sundmark, D. (2020). QuickREST: Property-based Test Generation of OpenAPI Described RESTful APIs. *IEEE International Conference on Software Testing, Verification and Validation (ICST)*. https://doi.org/10.1109/ICST46399.2020.00023

Tricentis. (2017). *Software Fail Watch 5th Edition*. https://www.tricentis.com/resources/software-fail-watch-5th-edition/.

Viglianisi, E., Dallago, M., & Ceccato, M. (2020). RESTTESTGEN: Automated Black-Box Testing of RESTful APIs. *IEEE International Conference on Software Testing, Verification and Validation (ICST)*. https://doi.org/10.1109/ICST46399.2020.00024

Zhang, M., Marculescu, B., & Arcuri, A. (2019). Resource-based test case generation for RESTful web services. *Genetic and Evolutionary Computation Conference (GECCO)*, 1426–1434. https://doi.org/10.1145/3321707.3321815