SSBSE'21 Tutorial: Search-Based System Testing with EvoMaster *

Andrea Arcuri[1,2][0000−0003−0799−2930]

[1] Kristiania University College, Norway
[2] Oslo Metropolitan University, Norway
andrea.arcuri@kristiania.no

Abstract. In this tutorial, I will show how to use the open-source EvoMaster tool to generate system-level test cases for different web services (e.g., RESTful APIs written in Java). The generated test cases can then be used to enable different kinds of research endeavors. I will also give an overview of the source code and architecture of EvoMaster, to help researchers that need to extend it for their research work.

Keywords: REST API Testing · SBST · Fuzzing.

1 Overview

EvoMaster [1, 2] is an open-source tool [8], under development since 2016. It is currently hosted on GitHub[3] with releases stored on Zenodo as well (e.g., [9]).

EvoMaster aims at system-level test case generation for web and enterprise applications, using evolutionary techniques, such as the MIO algorithm [3]. It currently targets REST APIs [4], but can be extended for other domains (e.g., support for GraphQL APIs is in beta-version at the time of this writing). EvoMaster supports both white-box and black-box testing [5]. Black-box testing is more general, and can be applied on any API regardless of the programming language they are written in. However, it lacks the advanced search-based heuristics like *testability transformations* [7] that can be used in the white-box version, as well as analyzing all interactions with SQL databases to improve the test generation even further [6].

The white-box testing is currently aimed at APIs running on the JVM, like the ones written in Java and Kotlin, where code-level search-based heuristics (e.g., the *branch distance*) are computed via bytecode instrumentation (which is fully automated). Furthermore, support for NodeJS (e.g., JavaScript) and .Net (e.g., C#) are currently under development.

EvoMaster is divided in two main components: a *core* and a *driver*. The *core* is written in Kotlin, and it contains the implementations of different search algorithms (e.g., MIO [3]), and all the code to evaluate the fitness function via

* This work is supported by the Research Council of Norway (project on Evolutionary Enterprise Testing, grant agreement No 274385).

[3] https://github.com/EMResearch/EvoMaster

HTTP calls toward the system under test (SUT). It also includes the code to output the evolved test cases in different formats, such as JUnit in either Java or Kotlin, Jest for JavaScript and XUnit for C#. The core is released as an executable jar file, where we provide as well installers for the main operating systems[4] (e.g., MSI for Windows). On the other hand, the *driver* is responsible to start/stop/reset the SUT, as well as doing all the instrumentations needed to compute code-level search-based heuristics. For SUTs running on the JVM, the *driver* is written in Java, and released on Maven Central[5].

The *core* and the *driver* will run on two separated processes, where the *driver* exposes its functionality via a REST API. This architectural decision was purposely made to be able to support further programming languages, besides the original Java (e.g., JavaScript and C#).

This tutorial at SSBSE'21 is aimed mainly at researchers, and not practitioners in industry. The goal of this tutorial is to show how to use EvoMaster to generate test cases for REST APIs, which could be used in different research contexts. I will also go through and discuss some of the source-code of EvoMaster, for researchers that want to extend it to investigate different research questions.

References

1. Arcuri, A.: RESTful API Automated Test Case Generation. In: IEEE International Conference on Software Quality, Reliability and Security (QRS). pp. 9–20. IEEE (2017)
2. Arcuri, A.: EvoMaster: Evolutionary Multi-context Automated System Test Generation. In: IEEE International Conference on Software Testing, Verification and Validation (ICST). pp. 394–397. IEEE (2018)
3. Arcuri, A.: Test suite generation with the Many Independent Objective (MIO) algorithm. Information and Software Technology **104**, 195–206 (2018)
4. Arcuri, A.: RESTful API Automated Test Case Generation with EvoMaster. ACM Transactions on Software Engineering and Methodology (TOSEM) **28**(1), 3 (2019)
5. Arcuri, A.: Automated black-and white-box testing of restful apis with evomaster. IEEE Software **38**(3), 72–78 (2020)
6. Arcuri, A., Galeotti, J.P.: Handling sql databases in automated system test generation. ACM Transactions on Software Engineering and Methodology (TOSEM) **29**(4), 1–31 (2020)
7. Arcuri, A., Galeotti, J.P.: Testability transformations for existing apis. In: 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST). pp. 153–163. IEEE (2020)
8. Arcuri, A., Galeotti, J.P., Marculescu, B., Zhang, M.: Evomaster: A search-based system test generation tool. Journal of Open Source Software **6**(57), 2153 (2021)
9. Arcuri, A., Galeotti, J.P., Marculescu, B., Zhang, M., Belhadi, A., Golmohammadi, A.: EvoMaster: A Search-Based System Test Generation Tool (Jun 2021). https://doi.org/10.5281/zenodo.4896807, https://doi.org/10.5281/zenodo.4896807

[4] https://github.com/EMResearch/EvoMaster/releases
[5] https://search.maven.org/artifact/org.evomaster/evomaster