Fuzzing vs SBST: Intersections & Differences

Giovani Guizzo
University College London (UCL)
London, UK
g.guizzo@ucl.ac.uk

Sebastiano Panichella
Zurich University of Applied Sciences
Zurich, Switzerland
panc@zhaw.ch

ABSTRACT

Search-Based Software Testing (SBST) is the application of SBSE for solving hard software testing problems. SBST has been the subject of discussion of our International SBST Workshop, co-located with the International Conference on Software Engineering (ICSE). In 2022 we hosted the 15th edition of our SBST Workshop, which brought together researchers and industrial practitioners to encourage the use of search-based and fuzz testing techniques and tools for addressing software engineering-specific challenges. In this 2022 edition, SBST held, among other exciting events, a discussion panel on the similarities and differences between SBST and Fuzzing. As it implies, the goal of the panel was to have a common ground for discussion on the main similarities and differences between the Fuzzing and SBST fields, focusing on how both communities can collaborate to advance the state-of-the-art on automated testing. This strong panel composed of researchers from both academia and industry was the highlight of SBST'22 and allowed the chairs of the workshop to make substantial changes for 2023. In this paper, we present the main take away messages from that seminal panel and highlight exciting new challenges in the field.

1. INTRODUCTION

Search-Based Software Testing (SBST) consists in the application of optimization algorithms to implement cost-effective test generation techniques, among other testing-related tasks [3]. SBST has been the subject of discussion of our International SBST Workshop, co-located with the International Conference on Software Engineering (ICSE). Some of the specific topics explored in the workshop include Multi-Objective Solutions, Search-Based Optimization, Test Oracle, SBST for Web and Mobile Software, Human Aspects and Integration into Real Test Environments, Enriching SBST with Machine Learning, Application of SBST in Industrial Practice and the Classroom, and Synergies of SBST and other SE areas.

In light of recent software testing advances, we also decided to include Fuzzing [6] in our SBST call for papers. The potential of SBST and Fuzzing approaches to be applied to a vast number of open and complex testing problems (e.g., autonomous driving functions or components [1, 5]) is increasingly evident.

Hence, in our 2022 edition of the SBST Workshop, we put forward the first efforts on discussing the combination of Fuzzing and SBST approaches by hosting a discussion panel entitled "**Fuzzing vs SBST - Intersections and Differences**". This panel was held with leading researchers in the field: **Andreas Zeller**, **Annibale Panichella**, **Lionel Briand**, **Marcel Böhme**, **Mark Harman**, **Myra Cohen**, and **Paolo Tonella**. The goal was to have a ground for discussion on the main similarities and differences between the Fuzzing and SBST fields. This seminal panel composed of researchers from both academia and industry was the highlight of SBST'22 and allowed the chairs of the workshop to make substantial changes for 2023.

The objective of this paper is to summarize the take away messages of this seminal panel. More details on the SBST 2022 workshop can be found at `https://sbst22.github.io/`, including the recordings of all presentations, keynotes, tutorial, and discussion panel. If you would like to watch the discussion panel recording in its entirety, we refer you to the link: `https://sbst22.github.io/panel/`.

2. SBST 2022

The 15th edition of SBST was co-located with ICSE 2022 and was held in virtual mode; 115 distinct participants attended the workshop during the day. From 2008 to 2013, SBST was co-located with the Intl. Conference on Software Testing, Verification and Validation (ICST). Since 2014, SBST has been co-located with ICSE. The 16th edition of the SBST workshop (renamed as Search-Based and Fuzz Testing, as explained later in the paper) will be co-located with ICSE 2023 (see `https://sbft23.github.io/`).

SBST 2022 consisted of one-day workshop, before the main conference on May 9, with the final program of the workshop available online: `https://sbst22.github.io/program/`. The starting point for the SBST discussions were the following elements:

Keynote. Paolo Tonella, from the University of Lugano, presented a keynote on "Deep Learning Testing".

Paper Sessions. We hosted four full and short paper presentations. The list of accepted papers and their recorded presentations can be seen on our website.

Tutorial. **Rahul Gopinath**, from the University of Sydney, Australia, presented a tutorial on "Learning and Refining Input Grammars for Effective Fuzzing".

Tool Competitions. As for previous years, SBST hosted two Tool Competitions: A Java testing tool competition and a Cyber-physical systems testing tool competition (`https://sbst22.github.io/tools/`). This edition of the workshop attracted thirteen tool papers.

Discussion panel on SBST vs Fuzzing. The star attraction of our 15th workshop edition. Composed by a set of leading researchers, this panel was a stage for a rich discussion on the challenges faced when automating testing tasks and how both communities handle those challenges. You can find the full panel recording in this link: `https://sbst22.github.io/panel/`.

The next section summarises the main take away messages we gathered from the discussion panel.

3. DISCUSSION PANEL: FUZZING VS SBST – INTERSECTIONS & DIFFERENCES

Although SBST 2022 was full of insightful ideas and exciting research, it is evident that the main attraction was the discussion panel on SBST and Fuzzing. The discussion panel entitled "**Fuzzing vs SBST – Intersections and Differences**" had as main goal to provide a common ground for discussion on the main similarities and differences between the Fuzzing [2, 6] and SBST fields [3]. Both have similar goals and apply similar techniques: automatically testing the software through heuristics to reveal faults. However, they differ in some aspects such as the scope of the testing (i.e., system testing vs unit testing), guiding technique (i.e., coverage vs fitness functions).

The panel was comprised by notorious software engineering researchers (in alphabetical order): **Andreas Zeller** (CISPA Helmholtz Center for Information Security & Saarland University); **Annibale Panichella** (Delft University of Technology); **Lionel Briand** (University of Ottawa & University of Luxembourg); **Marcel Böhme** (Max Planck Institute for Security and Privacy); **Mark Harman** (Meta Platforms Inc. & University College London); **Myra Cohen** (Iowa State University); and **Paolo Tonella** (Università della Svizzera Italiana). Their expertise was demonstrated in a two hour long discussion, which proved to be so impacting that it changed the future of the SBST workshop and, arguably, the future of automated software testing as a whole.

As follows, we summarise the main lessons learned from the discussion panel, as perceived by the authors of this paper. We also lay down a few challenges that can be tackled by both communities, perhaps known challenges, but nevertheless still relevant ones.

3.1 The SBST and Fuzzing Overlapping Insights

It is difficult to completely separate Fuzzing from SBST approaches: Similar strategies are often used to solve the same set of problems and with similar goals. The panelists have rightfully pointed out that it is hard to distinguish where the overlaps end with Fuzzing and SBST when the terms are obfuscated. In reality, what differs Fuzzing from SBST is the scope of the testing effort (system testing for Fuzzing and unit testing for SBST), but aside from that, the differences are most of time only perceived because of different terms adopted by both communities. In the end, the goal is the same: automatically reveal faults in the Software Under Test (SUT). The techniques used by both communities also share similar behaviour: intelligent search for meaningful test data.

Lack of knowledge sharing and known pitfalls: It is not unusual for both fields to apply the same techniques, achieve similar milestones, and sometimes fall into the same pitfalls when techniques prove to be sub-optimal. For example, one of the main perceived problems with SBST is the lack of scalability of the techniques due to costly solution evaluations. Fuzzing, on the other hand, has as one of its main advantages the incredible scalability of generating millions (if not billions) of inputs in the same time window an SBST technique takes to achieve a decent coverage milestone. Conversely, Fuzzing struggles with diversity of inputs, i.e., when an input makes the SUT crash, the Fuzzing technique may find itself generating the same input because it was successful in the past. In SBST, maintaining the diversity of inputs and avoiding local optima is something that has been investigated since its conception. However, because both communities do not collaborate as often as they should, such knowledge about what is adequate and what is not is seldom shared. This is not a major problem that can jeopardize scientific advances on the state-of-the-art, but trying to "reinvent the wheel" was never efficient.

The SBST and Fuzzing communities should walk together: This is the main take away message from the discussion panel (from Lionel Briand). A more tight collaboration between both communities is not only desirable, but rather a necessity to achieve a smooth, efficient, and more exciting scientific advancement. Both communities should walk together and join efforts to effectively and efficiently learn from each other. One of the most prominent results of such collaboration is the standardisation of terms in both communities, i.e., while one community calls a given term "A1", the other calls it "1A": they are essentially the same, but the search engines will definitely lose their marbles. To achieve this, it would be useful to organize common events, organize seminars, and have hybrid PC members. Such joint collaboration is long overdue.

3.2 Open Challenges

Scalability, the testing Achilles' tendon: When automating testing, be it with SBST or Fuzzing, one of the biggest concerns is scalability. It is not unknown that testing comprises most of the software development cost, and this high cost is not different for its automation. As mentioned, some techniques (we are looking at you Machine Learning) may struggle with the incurred overhead. Not only that, the entire process can be a burden to the engineer and cause the cost of the testing task to rapidly increase. We need to consider execution costs and attributes such as scalability of proposed tools (does an approach scale in a particular context?) when designing new techniques. The usage of "surrogate models" is a topic that could be investigated by both SBST and Fuzzing communities to test complex systems.

The lingering oracle problem: When Mark Harman (literally) brought his white elephant to our online room, called it "Oracle", and said we should address it, there was a single reaction from all participants: they all nodded (with visible pain). Given an input, is the generated output the correct one? Answering that question requires an oracle, the ever so elusive oracle. What we learnt during the panel discussion is that SBST still struggles with attaining a good oracle, while Fuzzing uses the answer to "has the program crashed?" as their main source. There is something beautiful about reducing the oracle problem to a simple "has it crashed or not?" question; it is undoubtedly efficient, but is it enough? More complex and continuous oracles may give us a smoother guidance to effective inputs, but the cost behind such complex oracles is still a problem. We hope that, by bringing SBST and Fuzzing together, we can finally find a middle ground where an enhanced simple oracle is all we need.

Flaky tests: Not surprisingly, panelists from both SBST and Fuzzing fields showed increased concern with flaky tests, i.e., tests that, although unchanged, one cannot predict their behaviour. Flaky tests introduce all sorts of uncertainty to scientific experiments and practical applications alike. Much like the oracle problem, flakiness is costly and generally requires some human effort. This was the only problem brought up during the discussion panel for which the panelists had no clear direction on how to solve it. Flakiness remains an open challenge, perhaps the hardest.

Other challenges and solutions: Human-into-the-loop testing could be an important challenging aspect to consider in the evaluation of SBST and Fuzzing techniques. Perhaps involving humans in the loop is not a challenge, but rather a solution. Who is it to say? Security testing was also brought up during the discussion. Although Fuzzing focuses on this aspect more commonly than SBST, it is still a great deal of importance in the testing community and for sure an open challenge.

4. FUTURE PLANS

As previously mentioned, the results of the rich and insightful discussion panel of SBST 2022 will echo through years to come. In light of this, we decided that the SBST workshop should focus more and more on bringing together international researchers and practitioners from both fields. The idea is to to collaborate, share experiences, provide directions for future research, and encourage the use of SBST and Fuzzing techniques for supporting developers and project managers in all aspects of the software development cycle.

Considering the inputs of the SBST 2022 panel, the general agreement of world reckoned testing experts and the SBST workshop Steering Committee is that SBST and Fuzzing overlap considerably and are converging towards a single research stream with similar goals. This observation has led to the decision (approved by the SBST workshop Steering Committee) to rename the SBST workshop as Search-Based and Fuzz Testing (SBFT). SBFT 2023 (`https://sbft23.github.io/`), similarly to the structure of the last SBST edition, will welcome standard research submissions (i.e., full and short papers), tool competitions, and keynote talks on SBST and Fuzzing alike.

The SBFT 2023 workshop will host three different Tool Competitions, as described at `https://sbft23.github.io/tools/`: the already consolidated Java Unit Testing Competition, the second Cyber-Physical Systems (CPS) Testing Competition, and a third, new, and exciting competition, called Fuzzing Competition. The latter will focus on using FuzzBench [4] as a benchmarking tool for the fuzzers. The 12th edition of the competition will be organized by **Abhishek Arya** (Google Inc.), **Dongge Liu** (Google Inc.), **Gunel Jahangirova** (King's College London), **Jarkko Peltomaki** (Abo Akademi University), **Johnatan Metzman** (Google Inc.), **Marcel Böhme** (Max Planck Institute for Security and Privacy), **Matteo Biagiola** (Università Della Svizzera Italiana), **Oliver Chang** (Google Inc.), **Stefan Klikovits** (Johannes Kepler University Linz), **Valerio Terragni** (University of Auckland), and **Vincenzo Riccio** (Università della Svizzera Italiana).

Complementary, for the 16th edition of SBFT workshop, **Prof. Lionel Briand**, having shared appointments between (1) The University of Ottawa, Canada and (2) The SnT centre for Security, Reliability, and Trust, University of Luxembourg, already accepted our invitations to give a keynote talk. We are confident that this Keynote will be great additions to an attractive workshop program. Finally, we plan to hold an one-hour-long discussion panel, entitled "Testing & Security for Cyber-physical systems (CPS)". We will invite at least six panelists in the areas of SBFT, we believe that this strong panel will be one of the highlights of SBFT'23.

5. REFERENCES

[1] Raja Ben Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. Testing autonomous cars for feature interaction failures using many-objective search. In *ASE*, pages 143–154, 2018.

[2] Valentin Jean Marie Manès, HyungSeok Han, Choongwoo Han, Sang Kil Cha, Manuel Egele, Edward J Schwartz, and Maverick Woo. The art, science, and engineering of fuzzing: A survey. *Transactions on Software Engineering*, 2019.

[3] Philip McMinn. Search-based software test data generation: A survey. *Software Testing, Verification and Reliability*, 14(2):105–156, June 2004.

[4] Jonathan Metzman, László Szekeres, Laurent Maurice Romain Simon, Read Trevelin Sprabery, and Abhishek Arya. FuzzBench: An Open Fuzzer Benchmarking Platform and Service. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2021, page 1393–1403, New York, NY, USA, 2021. Association for Computing Machinery.

[5] Andreea-Ina Radu and Flavio D. Garcia. Grey-box analysis and fuzzing of automotive electronic components via control-flow graph extraction. In Björn Brücher, Oliver Wasenmüller, Mario Fritz, Hans-Joachim Hof, and Christoph Krauß, editors, *CSCS '20: Computer Science in Cars Symposium, Feldkirchen, Germany, December 2, 2020*, pages 9:1–9:11. ACM, 2020.

[6] Andreas Zeller, Rahul Gopinath, Marcel Böhme, Gordon Fraser, and Christian Holler. *The Fuzzing Book*. CISPA Helmholtz Center for Information Security, 2021. Retrieved 2021-10-26 15:30:20+02:00.