4-2024

Extracting relevant test inputs from bug reports for automatic test case generation

Wendkuuni C. Ouédraogo

Laura Plein

Kader Kaboré

Andrew Habib

Jacques Klein

See next page for additional authors

Extracting Relevant Test Inputs from Bug Reports for Automatic Test Case Generation

Wendkûuni C. Ouédraogo, Laura Plein, Kader Kaboré, Andrew Habib, Jacques Klein
firstname.lastname@uni.lu
University of Luxembourg

David Lo
davidlo@smu.edu.sg
Singapore Management University

Tegawendé F. Bissyandé
tegawende.bissyande@uni.lu

ACM Reference Format:
Wendkûuni C. Ouédraogo, Laura Plein, Kader Kaboré, Andrew Habib, Jacques Klein, David Lo, and Tegawendé F. Bissyandé. 2024. Extracting Relevant Test Inputs from Bug Reports for Automatic Test Case Generation. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3639478.3643537

1 INTRODUCTION

The pursuit of automating software test case generation, particularly for unit tests, has become increasingly important due to the labor-intensive nature of manual test generation [6]. However, a significant challenge in this domain is the inability of automated approaches to generate relevant inputs, which compromises the efficacy of the tests [6].

In this study, we address the critical issue of enhancing the quality of automated test case generation. We demonstrate the presence of valuable relevant inputs within bug reports, showcasing their potential for improving software testing. To harness these inputs effectively, we introduce BRMiner, a novel tool designed for the extraction of relevant input values from bug reports. Our approach includes the modification of EvoSuite, a prominent automated test case generation tool, enabling it to incorporate these extracted inputs. Through systematic evaluation using the Defects4J benchmark, we assess the impact of BRMiner inputs on test adequacy and effectiveness, focusing on code coverage and bug detection. This study not only identifies the relevance of bug report inputs but also offers a practical solution for leveraging them to enhance automated test case generation in real-world software projects.

In the realm of automated test case generation, methods like Dynamic Symbolic Execution (DSE) [2] and Search-Based Software Testing (SBST) have been prevalent [3]. Despite their strengths, these techniques often struggle with generating contextually appropriate and realistic inputs [6]. This study, therefore, emphasizes the untapped potential of bug reports as a source of such inputs. Bug reports, rich in valid, human-readable inputs, are particularly beneficial for enhancing test coverage and detecting bugs.

ICSE-Companion '24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0502-1/24/04…$15.00
https://doi.org/10.1145/3639478.3643537

BRMiner, automates the extraction of relevant test inputs from bug reports, significantly enhancing the efficiency of test case generation. This is achieved by incorporating these inputs into EvoSuite, a leading SBST tool. The study showcases the advantages of integrating a feature in EvoSuite for external inputs, particularly from bug reports, to improve its efficacy in conjunction with DSE.

Related research in automatic test case generation provides context to our work. TestMiner [6], unlike BRMiner, extracts literals from existing tests for domain-specific values, and approaches like K-Config [4] and LeRe [7], focusing on compiler testing using bug report information, diverge from our approach. PerfLearner [1], which uses bug reports for extracting execution commands for performance bugs, also differs from BRMiner's focus on bug detection.

2 BRMINER

2.1 Usage Scenario

In a typical scenario, developers like Bob encounter software issues that are documented in bug reports. These reports are more than just records of software flaws; they are repositories of real-world inputs, crucial for effective testing. For instance, if a bug report notes that a crash occurs when a variable is set to a particular value, this insight can be instrumental in guiding testing efforts. BRMiner capitalizes on this concept, systematically extracting these inputs from bug reports to improve the debugging and testing procedures.

2.2 Overview

BRMiner extracts relevant inputs from bug reports (GitHub[1] and Jira[2]) by parsing them using Infozilla[3]). It separates code snippets from natural language text, where inputs can be embedded. BRMiner utilizes two methods for input extraction: directly from source code using Javalang[4], a Java source code parser, and from natural language text using regular expressions in bug reports.

BRMiner extracted inputs are then fed into automated test generators, such as EvoSuite, to improve test quality and software reliability. This approach bridges real-world scenarios with automated testing, enhancing issue detection and resolution. Figure 1 outlines the key steps of BRMINER for input extraction from bug reports for EvoSuite.

[1]https://github.com/sigmavirus24/github3.py
[2]https://github.com/pycontribs/jira
[3]https://github.com/nicbet/infozilla
[4]https://github.com/c2nes/javalang

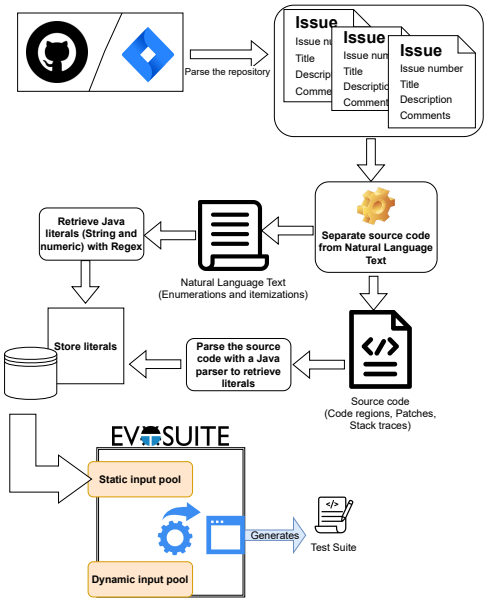

Figure 1: Overview of BRMiner, an automatic approach to extract potential test inputs from bug reports for EvoSuite.

3 EXPERIMENTAL SETUP AND RESULTS

Our empirical study evaluated BRMiner's effectiveness in extracting relevant bug report inputs for improving automated test generation. Using Defects4J 2.0.0 and EvoSuite 1.2.1, we modified EvoSuite to enable it to use BRMiner inputs in its static input pool as seed inputs, in addition to those from the dynamic pool for generating test cases. By this process we can transform inputs from BRMiner into test cases. Realistic testing scenarios were simulated with a three-minute time budget per test case and a focus on coverage criteria.

To conduct bug detection and code coverage experiments, we employed a Regression testing approach that closely resembled the one detailed in a previous study [5]. The experiments, conducted over five iterations on robust hardware (AMD EPYC 7552 48-Core Processor, 3.2 GHz, 640 GB RAM), provided a valid assessment of BRMiner's capabilities in enhancing automated test generation. In the context of these experiments, we then addressed three main research questions.

RQ1: How effectively are relevant inputs integrated into manually written test cases, and what is the rate of relevance for BRMiner-extracted inputs in comparison?

In our assessment of the presence of bug report inputs in manually written test cases and the accuracy of BRMiner in extracting relevant inputs, our analysis revealed that 14.37% of inputs in these test cases matched those found in bug reports, validating the significance of utilizing bug reports as a resource for test input extraction. Furthermore, BRMiner's extraction rate reached 68.7% when using Regex and 50.2% when using Javalang, highlighting BRMiner's ability to both quantitatively and qualitatively extract relevant inputs.

RQ2: Do tests generated using all the relevant inputs extracted by BRMiner exhibit a higher bug detection rate compared to tests generated without utilizing these inputs?

The goal was to evaluate the impact of BRMiner inputs on the bug detection capability of test cases. The results demonstrated that incorporating BRMiner inputs resulted in the detection of 45 unique bugs, surpassing the baseline approach, and thus proving the effectiveness of these inputs in bug detection.

RQ3: Does the utilization of all extracted relevant inputs by BRMiner result in higher code coverage compared to tests generated without using these inputs?

On average, branch coverage improved by approximately 1.15%, with the "Codec" project showing the most significant increase, reaching up to 2.6%. Instruction coverage saw an average improvement of about 1.17% across projects, while the "Codec" project demonstrated a notable 1.8% increase, particularly effective in specific contexts. Line coverage, on average, improved by about 1.25%, with the "JxPath" project standing out with enhancements of up to 2.24%. Even small improvements in code coverage are vital for bug detection as they enhance the quality and reliability of the software by uncovering potential defects and edge cases that lower coverage levels may miss.

4 CONCLUSION

Our research demonstrates the feasibility of extracting relevant bug report inputs to enhance automatic test case generation through BRMiner. We conducted extensive experiments, evaluating EvoSuite-eDSE with and without BRMiner's inputs for bug detection and code coverage. Promising results were achieved, with BRMiner successfully extracting 68.68% of relevant inputs using regular expressions, leading to the detection of 45 previously undetected bugs. BRMiner therefore represents an initial step in leveraging bug reports for relevant test inputs, with further potential for exploration.

5 ACKNOWLEDGEMENT

This work is supported by funding from the Fonds National de la Recherche Luxembourg (FNR) under the Aides à la Formation-Recherche (AFR) (grant agreement No. 17185670).

REFERENCES

[1] Xue Han, Tingting Yu, and David Lo. 2018. Perflearner: Learning from bug reports to understand and generate performance test frames. In *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*. 17–28.
[2] James C King. 1976. Symbolic execution and program testing. *Commun. ACM* 19, 7 (1976), 385–394.
[3] Annibale Panichella, Fitsum Meshesha Kifetew, and Paolo Tonella. 2017. Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets. *IEEE Transactions on Software Engineering* 44, 2 (2017), 122–158.
[4] Md Rafiqul Islam Rabin and Mohammad Amin Alipour. 2021. Configuring test generators using bug reports: a case study of gcc compiler and csmith. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. 1750–1758.
[5] Sina Shamshiri, René Just, José Miguel Rojas, Gordon Fraser, Phil McMinn, and Andrea Arcuri. 2015. Do automatically generated unit tests find real faults? an empirical study of effectiveness and challenges (t). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 201–211.
[6] Luca Della Toffola, Cristian-Alexandru Staicu, and Michael Pradel. 2017. Saying 'hi!' is not enough: mining inputs for effective test generation. In *Proceedings of the 32nd International Conference on Automated Software Engineering*. IEEE Computer Society, 44–49. https://doi.org/10.1109/ASE.2017.8115617
[7] Hao Zhong. 2022. Enriching compiler testing with real program from bug report. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–12.