Understandable Test Generation Through Capture/Replay and LLMs

Deljouyi, Amirhossein

DOI
10.1145/3639478.3639789

Publication date
2024

Document Version
Final published version

Published in
Proceedings - 2024 ACM/IEEE 46th International Conference on Software Engineering

Understandable Test Generation Through Capture/Replay and LLMs

Amirhossein Deljouyi*
a.deljouyi@tudelft.nl
Delft University of Technology
Delft, Netherlands

ABSTRACT

Automatic unit test generators, particularly search-based software testing (SBST) tools such as EvoSuite, efficiently generate unit test suites with acceptable coverage. Although this removes the burden of writing unit tests from developers, these generated tests often pose challenges in terms of comprehension for developers. In my doctoral research, I aim to investigate strategies to address the issue of comprehensibility in generated test cases and improve the test suite in terms of effectiveness. To achieve this, I introduce four projects leveraging Capture/Replay and Large Language Model (LLM) techniques.

Capture/Replay carves information from End-to-End (E2E) tests, enabling the generation of unit tests containing meaningful test scenarios and actual test data. Moreover, the growing capabilities of large language models (LLMs) in language analysis and transformation play a significant role in improving readability in general. Our proposed approach involves leveraging E2E test scenario extraction alongside an LLM-guided approach to enhance test case understandability, augment coverage, and establish comprehensive mock and test oracles.

In this research, we endeavor to conduct both a quantitative analysis and a user evaluation of the quality of the generated tests in terms of executability, coverage, and understandability.

CCS CONCEPTS

• **Software and its engineering** → Software testing and debugging.

KEYWORDS

Automatic Test Generation, Carving and Replaying, Large Language Models, Readability, Understandability, Unit Testing

ACM Reference Format:
Amirhossein Deljouyi. 2024. Understandable Test Generation Through Capture/Replay and LLMs. 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, 3 pages. https://doi.org/10.1145/3639478.3639789

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

ACM ISBN 979-8-4007-0502-1/24/04.
https://doi.org/10.1145/3639478.3639789

1 PROBLEM STATEMENT

In today's software-dominated world, software reliability and accuracy hold immense importance [18]. Consequently, software quality assurance has become an indispensable asset for software engineers. Automated testing in the form of unit tests has become a crucial element in ensuring high-quality software [6]. However, despite the widely acknowledged significance of testing, writing tests is seen as a tedious and time-consuming task [3, 7]. To alleviate this burden on developers and testers, the research community has devoted considerable effort to developing and evaluating automatic test generation approaches [1, 5, 12, 16].
Among the notable test generators are Randoop [22] and EvoSuite [12]. EvoSuite, for example, is a search-based test generator that employs genetic algorithms to construct a test suite [15], which has demonstrated convincing results in terms of coverage [14, 25]. There are, however, limitations in the quality of the test cases generated based on industrial case studies [2, 4, 13, 17, 23, 24, 28].

These limitations encompass challenges in (1) comprehending the generated test cases, and (2) generating tests for complex scenarios, which need complex test data or specific mock objects [4]. One significant limitation revolves around the understandability of the generated test cases, which involves various facets such as meaningful test data, proper assertions, well-defined mock objects, descriptive identifiers, lucid test names, as well as informative comments and summaries. In addition, the difficulty in following the scenario depicted in the test case and the ambiguity surrounding the test data significantly hamper this clarity [2, 8].

> While search-based unit test generators achieve reasonable test coverage, they fall short in generating understandable tests and struggle with generating tests for complex scenarios.

In this research, my focus is on enhancing the comprehensibility of the generated unit tests and having effective tests that include complex scenarios.

2 RESEARCH HYPOTHESIS

My hypothesis is that the E2E tests can provide a basis for enhancing the test suite at the unit test level with realistic and domain-specific test scenarios, and that Large Language Models (LLMs) can make generated test cases more like human-written tests. In addition, traditional search-based approaches excel at boosting coverage and generating highly executable tests.

I aim to leverage the strengths of both approaches, Capture/Replay and LLMs, in search-based test generators to improve the understandability of the generated test cases while achieving high coverage all at once.

A. Capture/Replay
The test suite includes a variety of types of tests besides unit testing, including End-to-End (E2E) testing [30]. The capture/replay approach captures fine-grained execution information during End-to-End (E2E) testing such as the order of method calls and the actual inputs, and subsequently replays them [11, 33]. Capture/Replay technique has been used to capture dynamic information for generating tests for regression testing [11] or reproduce a crash [10]. However, they have not been used in purpose of enhancing the understandability of generated tests. It is my hypothesis that this approach holds considerable potential to be used in the test generation process to have meaningful test scenarios, containing real and complex test data and effective mock objects.

B. Large Language Models (LLMs)
The realm of Natural Language Processing (NLP) offers a variety of techniques for test generation and optimization. These include traditional NLP methods [34], Deep Learning approaches [26], and the increasingly popular use of Large Language Models (LLMs). These methods are particularly adept at handling text-based tasks, with significant success in tasks like generating identifier names and crafting informative comments and summaries [21, 26]. Recent advancements in this domain have notably leaned towards deploying contemporary techniques, particularly focusing on LLMs [20, 21, 27, 31, 32]. These approaches involve fine-tuning pre-existing models, specifically tailored for test generation. Additionally, LLMs can enhance coverage when combined with Search-based algorithms [19]. It is my hypothesis that when we combine LLMs with search-based algorithms, we will be able to not only improve the code coverage, but we can also improve the understandability of the generated test cases.

3 THE EXPECTED CONTRIBUTIONS

My Ph.D. research centers around enhancing automated test case generation by integrating capture/replay and LLMs with search-based algorithms. The primary objective is to generate test cases that are not only executable and cover a wide range of corner cases but are also easily understandable by developers. Specifically, my investigation is steered by the following research questions:

- **RQ_1** **Feasibility:** Can the proposed approaches successfully generate executable unit tests?
- **RQ_2** **Understandability Evaluation:** How do the tests generated by the proposed approaches compare to state-of-the-art test generators in terms of understandability?
- **RQ_3** **Coverage Evaluation:** How do the proposed approaches perform in terms of test coverage compared to the current leading test generators?
- **RQ_4** **Developers' Perspectives:** How do developers perceive the understandability of generated tests compared to those written manually?

In order to address these questions, I have outlined four key research projects:

- ***RP*$_1$** Generating understandable unit tests through E2E tests.
- ***RP*$_2$** Generating understandable unit tests with high-coverage through a combination of Search-Based algorithms and LLMs.
- ***RP*$_3$** Seeding RP_2 with E2E Tests Scenario Carving.
- ***RP*$_4$** An empirical study on test comprehension and a comparison of different approaches from the developer's perspective.

I started with ***RP*$_1$**, where we introduced the *MicroTestCover* approach that generates unit tests starting from manual or scripted end-to-end tests. This led to test cases containing meaningful test scenarios and containing actual test data. The results of this study have been published in the following paper [9]:

> **Generating Understandable Unit Tests through End-to-End Test Scenario Carving**
> Amirhossein Deljouyi, Andy Zaidman. In *Proceedings of the 23rd IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM 2023)*, pp. 107-118.

In ***RP*$_2$**, our next step, I aim to expand my inquiry by integrating LLMs with a search-based test generator, EvoSuite, striving to achieve a higher level of comprehensibility and coverage. Progressing to ***RP*$_3$**, we plan to combine dynamic information and the carved tests from E2E testing and the approach outlined in RP_2. My hypothesis is that providing Search-Based algorithms and LLMs with realistic and domain-specific test scenarios will lead to the creation of more contextually relevant and understandable tests.

Finally, in ***RP*$_4$**, I aim to gather insights from developers/testers about the understandability of the generated test cases by the proposed approaches. Conducting an empirical human study, which is a notable gap, is crucial to determining comprehension of the generated tests by state-of-the-art approaches from the point of view of developers.

Research Impact, Who-What-How [29]: Software engineers and the research community will benefit from this research, particularly through tools for generating understandable unit tests and providing critical insights into the impact of various automated test generation approaches on test case comprehension. For instance, consider a common scenario in software development where a system is rapidly evolving and primarily relies on E2E tests. Software engineers can generate understandable unit tests with the innovations in this research, enabling faster and more precise fault localization. In addition to speeding up testing, understandable tests are easier to modify, update, and reuse.

4 EVALUATION PLAN

In ***RP*$_1$**, we conducted an exploratory case study involving four software systems to assess the feasibility of *MicroTestCarver*. Additionally, a user study with 20 participants was carried out to compare the understandability of *MicroTestCarver*-generated tests with EvoSuite-generated and manually-written test cases.

For ***RP*$_2$** and ***RP*$_3$**, our focus lies in evaluating the proposed approaches through both case studies and human studies. The case study will measure the approaches in terms of coverage, performance-efficiency/cost, and mutation score. Simultaneously, the human study aims to gauge the understandability of these approaches.

***RP*$_4$** entails a comprehensive human study to grasp developers' perspectives regarding the generated test cases.

ACKNOWLEDGMENTS

This research is done at Delft University of Technology under the supervison of Professor Andy Zaidman.

REFERENCES

[1] Shaukat Ali, Lionel C. Briand, Hadi Hemmati, and Rajwinder Kaur Panesar-Walawege. 2010. A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation. *IEEE Trans. Software Eng.* 36, 6 (2010), 742–762.

[2] M. Moein Almasi, Hadi Hemmati, Gordon Fraser, Andrea Arcuri, and Janis Benefelds. 2017. An Industrial Evaluation of Unit Test Generation: Finding Real Faults in a Financial Application. In *Int'l Conf. on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*. IEEE, 263–272.

[3] Maurício Finavaro Aniche, Christoph Treude, and Andy Zaidman. 2022. How Developers Engineer Test Cases: An Observational Study. *IEEE Trans. Software Eng.* 48, 12 (2022), 4925–4946.

[4] Andrea Arcuri. 2018. An experience report on applying software testing academic results in industry: we need usable automated test generation. *Empirical Software Engineering* 23, 4 (2018), 1959–1981.

[5] Luciano Baresi and Matteo Miraz. 2010. TestFul: automatic unit-test generation for Java classes. In *32nd IEEE/ACM International Conference on Software Engineering (ICSE)*. ACM, 281–284.

[6] Kent L. Beck. 2003. *Test-Driven Development - By Example*. Addison-Wesley.

[7] Moritz Beller, Georgios Gousios, Annibale Panichella, Sebastian Proksch, Sven Amann, and Andy Zaidman. 2019. Developer Testing in the IDE: Patterns, Beliefs, and Behavior. *IEEE Trans. Software Eng.* 45, 3 (2019), 261–284.

[8] Carolin E. Brandt and Andy Zaidman. 2022. Developer-centric test amplification. *Empir. Softw. Eng.* 27, 4 (2022), 96.

[9] A Deljouyi and AE Zaidman. 2023. Generating Understandable Unit Tests through End-to-End Test Scenario Carving. In *Proceedings of the 23rd IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM)*. 107–118.

[10] Pouria Derakhshanfar, Xavier Devroey, Gilles Perrouin, Andy Zaidman, and Arie van Deursen. 2020. Search-based crash reproduction using behavioural model seeding. *Softw. Test. Verification Reliab.* 30, 3 (2020).

[11] Sebastian Elbaum, Hui Nee Chin, Matthew B. Dwyer, and Jonathan Dokulil. 2006. Carving Differential Unit Test Cases from System Test Cases. In *Proc. Int'l Symposium on Foundations of Software Engineering (FSE)*. ACM, 253–264.

[12] Gordon Fraser and Andrea Arcuri. 2011. EvoSuite: Automatic Test Suite Generation for Object-Oriented Software. In *Proc. Joint Meeting Symp. Foundations of Software Engineering and the European Softw. Eng. Conf. (ESEC/FSE)*. ACM, 416–419.

[13] Gordon Fraser and Andrea Arcuri. 2013. EvoSuite: On the Challenges of Test Case Generation in the Real World. In *International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 362–369.

[14] Gordon Fraser and Andrea Arcuri. 2013. Whole Test Suite Generation. *IEEE Transactions on Software Engineering* 39, 2 (2013), 276–291.

[15] Gordon Fraser and Andrea Arcuri. 2015. Achieving scalable mutation-based generation of whole test suites. *Empirical Software Engineering* 20, 3 (2015), 783–812.

[16] Gordon Fraser, Matt Staats, Phil McMinn, Andrea Arcuri, and Frank Padberg. 2015. Does Automated Unit Test Generation Really Help Software Testers? A Controlled Empirical Study. *ACM Trans. Softw. Eng. Methodol.* 24, 4 (2015), 23:1–23:49.

[17] Giovanni Grano, Fabio Palomba, Dario Di Nucci, Andrea De Lucia, and Harald C. Gall. 2019. Scented since the beginning: On the diffuseness of test smells in automatically generated test code. *Journal of Systems and Software* 156 (2019), 312–327.

[18] Amy J. Ko, Bryan Dosono, and Neeraja Duriseti. 2014. Thirty years of software problems in the news. In *Proc. Int'l Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. ACM, 32–39.

[19] Caroline Lemieux, Jeevana Priya Inala, Shuvendu K. Lahiri, and Siddhartha Sen. 2023. CodaMosa: Escaping Coverage Plateaus in Test Generation with Pre-trained Large Language Models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 919–931.

[20] Vadim Liventsev, Anastasiia Grishina, Aki Härmä, and Leon Moonen. 2023. Fully Autonomous Programming with Large Language Models. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM.

[21] A. Mastropaolo, S. Scalabrino, N. Cooper, D. N. Palacio, D. Poshyvanyk, R. Oliveto, and G. Bavota. 2021. Studying the usage of text-to-text transfer transformer to support code-related tasks. *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)* (2021).

[22] Carlos Pacheco and Michael D. Ernst. 2007. Randoop: Feedback-Directed Random Testing for Java. In *Conf. on Object-Oriented Programming Systems and Applications (OOPSLA-Companion)*. ACM, 815–816.

[23] Fabio Palomba, Dario Di Nucci, Annibale Panichella, Rocco Oliveto, and Andrea De Lucia. 2016. On the Diffusion of Test Smells in Automatically Generated Test Code: An Empirical Study. In *2016 IEEE/ACM 9th International Workshop on Search-Based Software Testing (SBST)*. 5–14.

[24] Fabio Palomba, Annibale Panichella, Andy Zaidman, Rocco Oliveto, and Andrea De Lucia. 2016. Automatic Test Case Generation: What If Test Code Quality Matters?. In *Proceedings of the 25th International Symposium on Software Testing and Analysis (ISSTA)*. ACM, 130–141.

[25] Annibale Panichella, Fitsum Meshesha Kifetew, and Paolo Tonella. 2018. Automated Test Case Generation as a Many-Objective Optimisation Problem with Dynamic Selection of the Targets. *IEEE Trans. Software Eng.* 44 (2018), 122–158.

[26] Devjeet Roy, Ziyi Zhang, Maggie Ma, Venera Arnaoudova, Annibale Panichella, Sebastiano Panichella, Danielle Gonzalez, and Mehdi Mirakhorli. 2020. DeepTC-Enhancer: Improving the readability of automatically generated tests. In *Proc. Int'l Conf. on Automated Software Engineering (ASE)*. 287–298.

[27] Max Schäfer, Sarah Nadi, Aryaz Eghbali, and Frank Tip. 2023. An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation. arXiv:2302.06527 [cs.SE]

[28] Sina Shamshiri, Rene Just, Jose 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. In *International Conference on Automated Software Engineering (ASE)*. IEEE, 201—-211.

[29] Margaret-Anne Storey, Neil A Ernst, Courtney Williams, and Eirini Kalliamvakou. 2020. The who, what, how of software engineering research: a socio-technical framework. *Empirical Software Engineering* 25 (2020), 4097–4129.

[30] Ham Vocke. 2018. The Practical Test Pyramid. https://martinfowler.com/articles/practical-test-pyramid.html

[31] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. 2023. Software Testing with Large Language Model: Survey, Landscape, and Vision. arXiv:2307.07221 [cs.SE]

[32] Shengcheng Yu, Chunrong Fang, Yuchen Ling, Chentian Wu, and Zhenyu Chen. 2023. LLM for Test Script Generation and Migration: Challenges, Capabilities, and Opportunities. arXiv:2309.13574 [cs.SE]

[33] Andreas Zeller, Rahul Gopinath, Marcel Böhme, Gordon Fraser, and Christian Holler. 2021. *The Fuzzing Book*. CISPA Helmholtz Center for Information Security.

[34] Benwen Zhang, Emily Hill, and James Clause. 2016. Towards Automatically Generating Descriptive Names for Unit Tests. In *Proc. Int'l Conf. on Automated Software Engineering (ASE)*. ACM, 625–636.