Analysis of Fault Localization Techniques on REST APIs

Aakash Kulkarni
dept. name of organization (of Aff.)
name of organization (of Aff.)
Corvallis, United States
kulkaraa@oregonstate.edu

Soon Song Cheok
dept. name of organization (of Aff.)
name of organization (of Aff.)
Corvallis, United States
cheoks@oregonstate.edu

***Abstract*—**

***Index Terms*—**

I. Introduction

A. Problem statement

While fault localization techniques are pivotal in reducing the time and effort required for debugging, their effectiveness in the context of REST APIs—which are crucial in today's distributed systems—is not well understood. The challenge lies in accurately identifying faults within REST APIs using automated techniques, where faults may vary widely in nature and complexity. **[[What is the problem statement that you plan to solve?]]**

B. Motivation

Effective fault localization can significantly expedite the maintenance process of REST APIs, which are integral to modern software ecosystems. Enhancing these techniques can lead to reduced downtime and improved service quality, directly impacting user experience and operational costs. Addressing the efficacy of fault localization in REST APIs is crucial for advancing automated debugging tools and practices in software engineering. **[[Why should anyone care about solving that problem?]]**

C. Relation with software engineering research

Fault localization is a well-established area of research within software engineering, focusing primarily on traditional software systems. However, the unique characteristics of REST APIs, such as their stateless nature and network dependency, introduce new challenges that are not fully addressed by existing studies. Previous works have created taxonomies of faults in REST APIs but have not extensively explored the applicability of fault localization techniques to these faults. This research seeks to bridge that gap by applying these techniques to a curated dataset of faults specific to REST APIs. **[[How is the problem related to software engineering research? What is already known about that problem space, and what is still unknown that you are interested in solving?]]**

D. Key Insight or Idea

The core of this research is the hypothesis that existing fault localization techniques can be adapted to effectively identify faults in REST APIs. By leveraging a detailed taxonomy of API faults and employing a tool like EvoMaster for automated test case generation, this project aims to systematically assess which fault categories are more amenable to automation. The feasibility of this approach within a 5-week timeframe is supported by the availability of comprehensive datasets and mature tools. **[[What is the high-level approach that you would like to explore to the solve the problem? Why you feel you can succeed with that approach in 5 weeks?]]**

E. Assumptions

[[What kind of assumptions will you need to make for your choice of solution?]]

F. Research questions

The primary research questions are formulated as follows:

- **RQ1:** Do fault localization techniques effectively localize faults in REST APIs?
- **RQ2:** What categories of faults (as per the existing taxonomy) are most effectively localized by current techniques?
- **RQ3:** Which fault localization techniques offer the highest accuracy and precision in the context of REST APIs?

[[What research question(s) will you answer?]]

G. Evaluation Dataset

For the evaluation dataset, we will mine bugs and their fixes from REST API projects that utilize Spring Boot or Jersey frameworks. This dataset will encompass various categories of faults identified in REST APIs. By analyzing repositories and commit histories, we will extract specific instances where bugs have been documented and subsequently fixed. The categorization of these faults will be aligned with an established taxonomy of API faults, ensuring that each bug is classified according to its nature and impact. This rigorous dataset compilation aims to provide a comprehensive basis for testing fault localization techniques within a controlled, relevant environment. **[[What kind of dataset will you**

evaluate your solution on? Where and how will you get that dataset?]]

H. *Evaluation metrics*

For the project, the evaluation of fault localization techniques using the dataset derived from REST APIs employing Spring Boot and Jersey frameworks will be guided by the following metrics:

Precision: This metric will measure the accuracy of the fault localization techniques in identifying only the actual faults. Precision is calculated as the ratio of true positive results (faults correctly localized) to the total number of faults that the techniques reported as localized (true positives plus false positives). This metric helps assess the exactness of a technique in pinpointing faults.

Recall: This metric will evaluate the completeness of the fault localization techniques by measuring the ratio of true positive results to the total number of actual faults present (true positives plus false negatives). High recall indicates that the technique can localize a large percentage of all existing faults.

F-Measure Score: Since precision and recall can sometimes present a trade-off, the F-measure Score will be used to find the harmonic mean of precision and recall. This measure balances both metrics and gives a single score to evaluate the overall effectiveness of a fault localization technique.

Accuracy: This metric measures the overall correctness of the fault localization techniques in identifying faults within the REST API projects. It is calculated as the ratio of correctly localized faults to the total number of faults in the dataset.

These metrics will collectively provide a thorough assessment of each fault localization technique's ability to detect and accurately categorize faults in REST APIs, contributing to a detailed understanding of their practical utility and areas for improvement. **[[How will you know that you have solved the problem successfully? In other words, how will you evaluate your solution?]]**

II. Background and Motivation

III. Related Work

[[R2Fix [1] and iFixR [2], use information retrieval-based fault localization (IRFL) that ranks suspicious program statements based on their similarity with bug reports.]]

IV. Approach

V. Evaluation

A. *Dataset*

B. *Metrics*

C. *Experiment Procedure*

D. *Results*

VI. Discussion and Threats to Validity

VII. Contributions

References

[1] C. Liu, J. Yang, L. Tan, and M. Hafiz, "R2Fix: Automatically generating bug fixes from bug reports," in *IEEE International Conference on Software Testing, Verification and Validation*, 2013, pp. 282–291.

[2] A. Koyuncu, K. Liu, T. F. Bissyandé, D. Kim, M. Monperrus, J. Klein, and Y. L. Traon, "iFixR: bug report driven program repair," in *27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2019, p. 314–325.