TSE: Project Proposal

Shirish Singh, Aria Jiang, Yiqiao Liu

March 2021

1 Project Description

The project aims to build a tool that helps find code clones across multi-file projects. It will be based on code understanding through behavioral code similarity (Type-4 clones). A motivating scenario for such a tool is that if a developer wants to understand a piece of code that s/he has encountered for the first time, it would be helpful to find similar and familiar code that the developer might have developed before. The tool can find behavioral similarity between the new code and code the developer is familiar with. More broadly, the applications of behavioral similarity are manifold: a) Program understanding for first-time developers, b) code search, c) code refactoring, and so on. In this work, we will use test cases to measure similarity. At the same time, we will attempt to answer the following research questions:

- **RQ1**: How many test cases do we need to use (generate) for finding high confidence code clones (Preponderance of evidence)?
- **RQ2**: How to generate optimal test cases that can cover the functionality of the program?

2 Our Plan

We have looked into SLACC [3]. SLACC's application has been limited to the Google CodeJam dataset, which consists of individual file-based programs and trivial input. For our project, we are planning to build a real-time code similarity tool that can be used to find similar functions in real projects that contain multiple files. Figure 1 shows the high level overview of our system. Unlike Google CodeJam, real-world projects have associated test cases that can help with behavioral similarity. We might modify SLACC or build upon a part of it to build the proposed tool.

Besides SLACC, we plan to use the 106 student submissions [1] from the Advanced Software Engineering course that took place in Fall-2020 as dataset for the project.

[1] https://github.com/jxm033f/4156-PublicAssignment/network/members

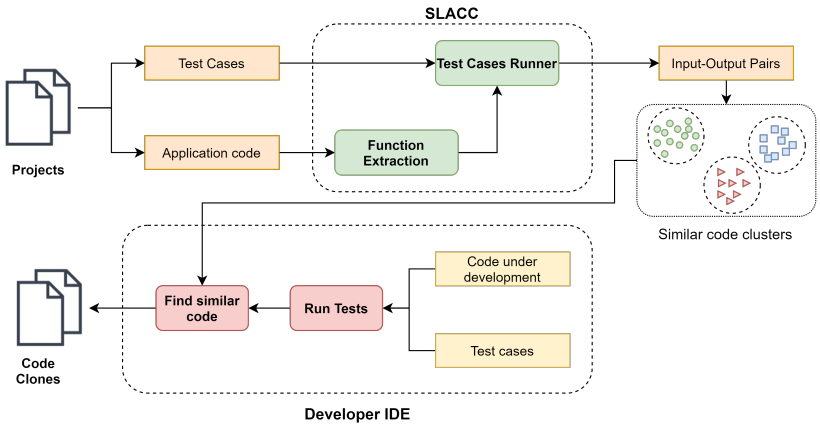

Figure 1: High level overview

In terms of milestones we hope to achieve, please find details in the table below.

Milestones	Tasks
First Progress Report	Process dataset, test function extraction and test case runner in SLACC
Second Progress Report	Modify SLACC to build tool for dataset
Project Demo	Test tool

3 How the project is relevant to this course

The project is relevant to our course because such a tool has ample applications in advancing the field of software engineering. Additionally, the topic of code understanding, and more specifically code understanding through testing is an important area within the field of software engineering.

4 Why we are interested in doing this project

This project furthers what existing state-of-the-art testing-based code understanding tool (SLACC) does by building a tool that works for multi-file projects. Therefore this project addresses an important research question and can advance our understanding of the topic.

Additionally, this project has very practical potential applications that could boost developer productivity while minimizing stress. We have included a detailed motivating potential application below.

Imagine your next job as a software engineer at a startup. You are tasked with implementing a feature for a product. To be able to implement the feature, you need to first isolate the locations for changes to be added in the codebase,

which in turn requires you to understand some parts of the codebase. If the code is not well documented (this problem is prominent in the literature [[1, 2]]), you will spend a lot of time understanding what is going on in the code. However, if you had a nice IDE plugin that could find similar and well-documented code snippets for every piece of code that you review, you will save a lot of time.

5 If the project is related to your midterm paper

This project is related to Shirish's midterm paper. It is somewhat related to Aria and Yiqiao's midterm paper in that the topic is in code understanding and test case generation. However, the approach is a bit different in that we are taking a testing-based approach rather than a machine-learning-based approach.

References

[1] L. C. Briand. Software documentation: how much is enough? In *Seventh European Conference onSoftware Maintenance and Reengineering, 2003. Proceedings.*, pages 13–15, 2003.

[2] Vikas S Chomal and Jatinderkumar R Saini. Significance of software documentation in software development process. *International Journal of Engineering Innovations and Research*, 3(4):410, 2014.

[3] George Mathew, Chris Parnin, and Kathryn T Stolee. SLACC: Simion-Based Language Agnostic Code Clones. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, page 210–221, New York, NY, USA, 2020. Association for Computing Machinery.