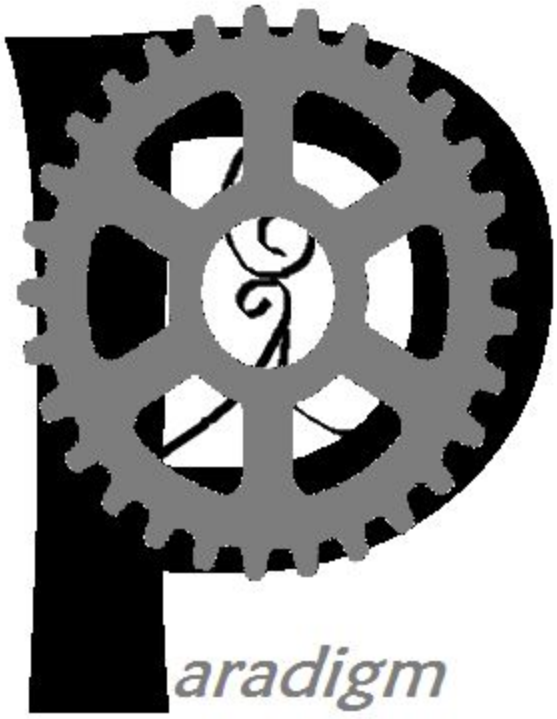
Test Suite
Final Report

v.1.0



Team Members:
Cavaughn Browne
Christian Norfleet
Damien Moeller
Aimee Phillips

1. Introduction

The purpose of this document is to serve as a post-mortem for the project. This document describes and analyzes the problems encountered during the project and deviations from the original specified requirements. It also describes the parts of the project the Paradigm group members were unsatisfied with and hoped to improve if more time permitted as well as the lessons learned during the development of the Test Suite

2. Overview of project

Students often have problems with testing their programs. They currently submit programs without proper testing due to lack of knowledge of testing methods and tool-sets. Although the program might seem functional, the students may overlook possible faults and develop bad habits. So, the Test Suite application will aid said students by parsing their source code and generating applicable test cases and test drivers as well as generating test reports for use and review.

3. Project Details

3.1. Problems Encountered

During the course of development, we encountered a few problems. In the beginning, all of our team members were not familiar with the language in which the application would be coded. So, using the pair programming technique, an inexperienced member had to be assisted by a more experienced member as they performed designated tasks, Moreover, the team had to figure what goals would and would not be feasible to achieve at the end of the project. So, we had an unclear start at first, however, by having meetings and discussions with the customer, we were able to clear up misunderstandings and develop a clearer idea of the system that needed to be developed.

3.2. Deviations from specifications

During the course of development, our team had different ideas and problems which solutions caused some deviations from the original specified requirements. For boundary testing the application does not simply analyze the source file for all the

information. The user can now specify the lower and upper limits for an input through the Graphical User Interface (GUI). Therefore, the GUI needed to be changed to accommodate the new functionality. In addition, in the requirements, it was stated that the application would be coded in C# alone. However, the team decided on using a Python script to parse the source files instead of C#. The C# application would run the script when needed using IronPython. Lastly, the user can open an electronic copy of the user manual in the application but it requires the user to have a default PDF reader not stated in the requirements.

3.3. Remaining Problems

The Test Suite does not accurately find inputs in c++ source code if the following conditions don't hold:

- Inputs must be declared on the same line as their type
- Inputs must use >> operator or the getline() function
- An input variable with the >> operator has to be on the same line as the operator
- Correct test drivers will only be generated if the class in the .h file has only public methods

Other minor bugs found include:

- When tabs are switched, there is a "jerky" effect, however, that doesn't affect the rest of application.
- Corresponding menu strip items aren't being disabled with their buttons.
- The testing methods do not differentiate between .h files and .cpp files e.g. (O-O testing should only work on .h files and shouldn't with .cpp).

4. Reflection

4.1. What we would do if more time permitted

If more time permitted, our team would have worked on improving and adding more functionality to the application. Some of the improvements are as follows:

- Fix remaining errors.
- Multi-Language Support - Students code assignments in more than just C++. With more language support included, the application would be more inclusive in terms of students who need to do assignments and test in languages other than C++.
- Implement Loop, Path and Functional testing methods.

- Test the application even more - Testing is very important as it becomes difficult to deal with the consequences of errors after deployment. In order to reduce as much errors as possible in the application, more testing would have been great use of our time.

4.2. What we would do different if we had it to do all over again

- Better project planning and version management - Proper project planning is usually what determines whether a project fails or not. There times during the project where better project planning could have helped us avoid some of the pitfalls and difficulties of developing the system.
- Have more in-person meetings - The team usually meets on Mondays, however, not week. A lot of times, discussion would be held over Slack and Google Hangouts. However, this created a digital barrier between the members that would not be there in an in-person meeting. This led to misunderstandings and times where activities were completed too close to the deadline.

4.3. Lessons Learned

- How to effectively work in teams
- Team management is critical
- The need for clear communication and task management
- The usefulness and importance of the methodologies learned
- Importance of testing / Testing is never done
- Expect the unexpected
    - “Plans never go as devised so it's important to have a fallback to account for the unexpected hiccups”

5. Summary

Overall, the Test Suite Application is designed solely to help early Undergraduate level computer science students test their code more thoroughly. During the project we had our problems, however, we worked through them and produced the application (with a few issues remaining not fixed) due to time constraints. Our team has learned a lot from this project and would do things differently if we had to do it over again. Unfortunately we can’t, but. our team will apply what has been learned to future projects, so it has overall been a fruitful endeavor.