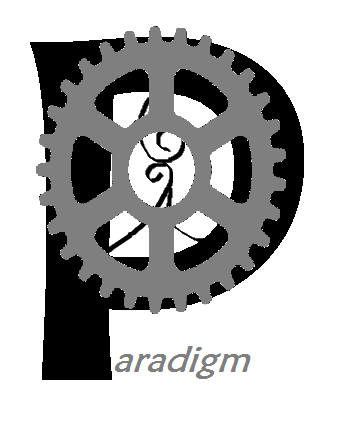
**Test Suite**

**Requirement Document**

**Christian Norfleet**

**Aimee Phillips**

**Cavaughn Browne**

**Damien Moeller**

**Table of Contents**

**1. Introduction**

**1.1. Purpose of Document**

**1.2. Main Objective**

**1.3. Specific Goals**

**1.4. Overview of Document**

**2. Users**

**2.1. Expected users**

**2.2. Use Cases**

**2.3. Scenarios**

**3. System**

**3.1. Development Environment**

**3.2. Target environment**

**3.3. Functional Requirements**

**3.3.1. Issues**

**3.3.2. Major Subsystems**

**3.3.3. Major Functions**

**3.3.4. Major Classes identified and listed**

**3.3.5. Minor System Functions**

**3.4. User Interface Specifications**

**3.5. Non-Functional Requirements**

**3.5.1. Management**

**3.5.2. Technical**

**3.5.3. Performance**

**3.5.4. Security**

**3.6. System Evolution/ Maintenance**

**4. Other Deliverables required**

**5. Risks**

**6. Additional Items**

**7. Glossary of terms used in this document**

**8. References**

**1. Introduction**

**1.1. Purpose of Document**

Students from CMPS 1044 to CMPS 3013 often have problems with testing their programs. Students 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. This program will aid said students by parsing code and generating applicable test cases and test drivers.

**1.2. Main Objective**

The main objective is to create a test suite for other students to use to test their programs with various data. This testing program will provide a user friendly GUI to produce a test report that the users can use to improve their programs. The development will transpire over a period of 4 months. This program will be open source to enable future teams to expand the feature set of functions to be implemented.

**1.3. Specific Goals**

The client requires testing program to be compatible with the C++ language. This program will be designed for the windows environment. The interface will feature open dialogue box for file submission, and will allow user to select various languages, various test methods, and levels of test.

**1.4. Overview of Document**

The rest of the items in this document will address hardware requirements, software requirements, future users, all functions and constraints, and GUI (Graphical User Interface) requirements.

**2. Users**

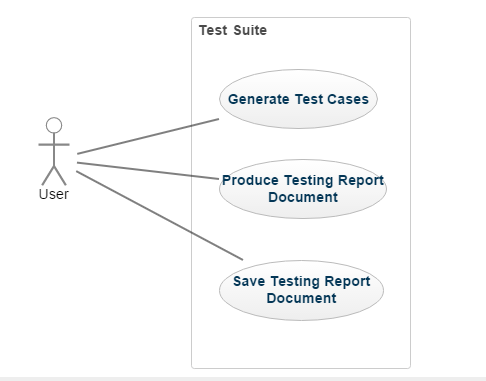
**2.1. Expected Users**

The users of this system will be Computer Science Undergraduates from the introductory Computer Science course CMPS 1044 to the CMPS 3013 Advance Structures and Algorithms.

**2.2. Use Cases**

* Generate Test Cases
* Produce Testing Report Document
* Save Testing Report Document

Use Case Diagram:



**2.3. Scenarios**

A student who needs to submit an application or program that needs to be well tested and used for a grade. So, a computer science undergraduate will launch the application and will browse for a folder or a file with source code to it. They will select options for the kind of language and type of testing they want to do. Then the application will give them a list of test cases categorized by type of testing for them to run their program on. It will allow them to enter the results they expect into the grid or table. The app will compare their results with expected to results. Then indicate if their test case passed or failed. The student may then generate and save a report of their testing and turn it into the professor if required.

**3. System**

**3.1. Development Environment**

Windows OS with Visual Studio.

**3.2. Target environment**

Recommended user system:

* Software: Windows OS, Microsoft Visual Studio
* Hardware: Personal desktop or notebook computers, CD Drive, USB Port

**3.3. Functional Requirements**

Functional requirements

* Must be able to give sets of test cases for input source file code.
* Must be able to read source code files.
* Must be able to produce test variables and test drivers.
* Must be able to compare expected results and program results
* Must be able to produce a template for a testing report document based on current session tests.
* Must be able to save testing report document.

**3.3.1. Issues**

* + - Parse a specified file for input.
    - Generate a test report for current session.
    - Create a user friendly GUI that is easy to work with.
    - Generate the different types of test cases.

**3.3.2. Major Subsystems**

**3.3.3. Major Functions**

* + - * Read Source code.
      * Generate test cases or drivers.
      * Create testing report.
      * Save testing report.

**3.3.4. Major Classes identified and listed**

* Test Generator.
* User Interface.

**3.3.5. Minor System Functions**

* + - Check data type.
    - Read class functions

**3.4. User Interface Specifications**

In accordance to the customer’s needs, the preliminary user interface should consist of at least the following:

* Main window with title text and window management buttons
* Main menu (file, about, etc)
* Check Boxes for the options given to the user
* Grid for displaying generated test cases, input, expected output, and the source file program output.
* Buttons (save, check expected results with the actual results, etc)

**3.5. Non-Functional Requirements**

Nonfunctional requirements:

* Must be extendable to reading more languages over time.
* Must be scalable to allow for other types of testing methods.
* Must be able to run on Windows Operating Systems.
* Source code must be in C++ (.cpp and .h).

**3.5.1. Management**

No financial cost during project development. It must be developed during a 4 month period within the spring 2017 semester, and finished by May 2017.

**3.5.2. Technical**

The system must run on Windows 7 or above. It will be written in C#.

**3.5.3. Performance**

The system will make at most one pass per test session over the source file.

**3.5.4. Security**

N/A

**3.6 System Evolution/ Maintenance**

The system will be developed with expansion in mind. The delivered system will be passed on, so that in the future it will be able to be analyze additional languages and use various other testing methods.

**4. Other Deliverables required**

At the end of the semester the system will satisfy the requirements stated in this document. The system will be delivered to the next batch of developers for further expansions.

**5. Risks**

The risks involved in the development of the system include lack of communication. Communication between the team and the customer. Also, another risk would be that the developers would not have a complete understanding of the program requirements or what the customer is expecting or wants.

**6. Additional Items**

**7. Glossary of terms used in this document**

GUI - Graphical User Interface

**8. References**

Stringfellow, Dr. Catherine