C++ Automated Test Suite

Software Engineering Documentation

Intolerable Optimists

Charles Parsons Christopher Smith Jarod Hogan

February 19, 2014

Contents

List of Figures

List of Tables

List of Algorithms

Mission

Create an automated testing suite for C++ programs.

Document Preparation and Updates

Current Version [X.X.X]

Prepared By:
Team Member #1
Team Member #2
Team Member #3

Revision History

Date	***Author***	***Version***	***Comments***
2/2/12	*Team Member #1*	*1.0.0*	*Initial version*
3/4/12	*Team Member #3*	*1.1.0*	*Edited version*

1

Overview and concept of operations

The overview should take the form of an executive summary. Give the reader a feel for the purpose of the document, what is contained in the document, and an idea of the purpose for the system or product.

1.1 Scope

The purpose of this project is to create a program that will run another program against test documents and output the results into another file forthe user. The program is specifically geared toward computer science professors for their use in grading student programs. This program will compile and run the submitted program imputting the test data. The results, which are saved to a text document, are evaluated and and percentage of pass/fail is also computed and saved in the document.

1.2 Purpose

This system is designed in order to make it easier for professors to grade their students programs. The test files need only be in the same directory (or within a subdirectory) as the program to be compiled and run. Also, the output is to be detailed enough so that the user knows exactly which test cases passed and which failed.

1.2.1 Program Compiler

Though rather simple to code this is a mojor component because it depends on which system the program is running as to how the program is compiled. This program is made to run using the gcc command.

1.2.2 Test File Location

Also a simple but necessary component to our system. The program is going to run every file ending in ".tst" within the same directory as the progam itself. Therefore, searching the current working directory as well as every subdirectory is very important to ensure every test case is run.

1.2.3 Test Case Evaluation

Here is the most critical part of the program. This is where the tests are read in by our program and the results evaluated. Onceevaluated the results are stored in a text file for the user to review.

1.3 Systems Goals

The goals of our system is to successfully compile and run a program with the test files in the directory. Then to output the result of the tests in a text file.

1.4 System Overview and Diagram

1.5 Technologies Overview

2

Project Overview

This section provides some housekeeping type of information with regard to the team, project, etc.

2.1 Team Members and Roles

Charles Parsons - Product Owner Christopher Smith - Scrum Master Jarod Hogan - Technical Lead

2.2 Project Management Approach

The management approach we are taking with this project is Agile Scrum. We are using a Trello board to keep track of the product backlog, sprint backlog, sprint tasks, and user stories.

2.3 Phase Overview

The first phase of the testing suite will accept a C++ source file and search the current directory, as well as all sub-directories, for test case files that end in ".tst".

2.4 Terminology and Acronyms

3

User Stories, Backlog and Requirements

3.1 Overview

This system is for professors to test their students' programs against test data contained in text files. The results of these tests will then be written to a text file for th user to evaluate the tested programs performance.

Title: Dr. Logar's User Story As a professor I want to have a program compile, run, test, and evaluate the tests run on a program submitted to me by my students so that I no longer have to run each test case individually and evaluate them by hand.

Below: list, describe, and define the requirements in this chapter. There could be any number of subsections to help provide the necessary level of detail.

The progam should compile and run a program against all test cases contained in files ending in ".tst" within the current directory or within a subdirectory. In addition, it will evaluate the passes and fails of the program and log this in a text document for user review.

3.1.1 Scope

The purpose of this project is to create a program that will run another program against test documents and output the results into another file forthe user. The program is specifically geared toward computer science professors for their use in grading student programs. This program will compile and run the submitted program imputting the test data. The results, which are saved to a text document, are evaluated and and percentage of pass/fail is also computed and saved in the document.

What scope does this document cover? This document would contain stakeholder information, initial user stories, requirements, proof of concept results, and various research task results.

3.1.2 Purpose of the System

This system is designed in order to make it easier for professors to grade their students programs. The test files need only be in the same directory (or within a subdirectory) as the program to be compiled and run. Also, the output is to be detailed enough so that the user knows exactly which test cases passed and which failed.

3.2 Stakeholder Information

There are only two main stake holders in this project. The first is Dr, Logar, our customer, who naturally expects a working progam. The second stake holder is this team who greatly desires an A in this course.

3.2.1 Customer or End User (Product Owner)

Dr. Logar is our customer and end user in this case.

3.2.2 Management or Instructor (Scrum Master)

Cris Smith is our Scrum Master. He plans the time line we are to follow as well as the scrum meetings

3.2.3 Investors

There are no Investors.

3.2.4 Developers –Testers

There is not set developer(s) or tester(s). We divide up the work as evenly as possible. After running our own tests on our code we are sure to test the finished product to ensure integrating the code was successful.

3.3 Business Need

This program is aimed at reducing the amount of time a professor must personally spend on each program by automating the test phase.

3.4 Requirements and Design Constraints

The main constraint of this program is that it must run on a Linux based operating system and be used to compile C++ programs. Other constraints are that this program does require a basic knowledge of command line compiling.

3.4.1 System Requirements

The system requirements are that only C++ programs be used with this project on a Linuz machine.

3.4.2 Network Requirements

There are no network requirements as everything can be done offline.

3.4.3 Development Environment Requirements

Our developement enviornment is Linux using simple text editors with g++ compile command.

3.4.4 Project Management Methodology

The stakeholders might restrict how the project implementation will be managed. There may be constraints on when design meetings will take place. There might be restrictions on how often progress reports need to be provided and to whom.

- What system will be used to keep track of the backlogs and sprint status?
- Will all parties have access to the Sprint and Product Backlogs?
- How many Sprints will encompass this particular project?
- How long are the Sprint Cycles?
- Are there restrictions on source control?

3.5 User Stories

Title: Dr. Logar's User Story As a professor I want to have a program compile, run, test, and evaluate the tests run on a program submitted to me by my students so that I no longer have to run each test case individually and evaluate them by hand.

This section should contain sub-sections to define and potentially provide a breakdown of larger user stories into smaller user stories.

3.5.1 User Story #1

User story #1 discussed.

3.5.1.a User Story #1 Breakdown

Does the first user story need some division into smaller, consumable parts by the reader? This does not need to go to the level of actual task definition and may not be required.

3.5.2 User Story #2

3.5.2.a User Story #2 Breakdown

User story #2 ....

3.5.3 User Story #3

3.5.3.a User Story #3 Breakdown

User story #3 ....

3.6 Research or Proof of Concept Results

This section is reserved for the discussion centered on any research that needed to take place before full system design. The research efforts may have led to the need to actually provide a proof of concept for approval by the stakeholders. The proof of concept might even go to the extent of a user interface design or mockups.

3.7 Supporting Material

This document might contain references or supporting material which should be documented and discussed either here if approprite or more often in the appendices at the end. This material may have been provided by the stakeholders or it may be material garnered from research tasks.

4

Design and Implementation

4.1 Major Component #1

4.1.1 Technologies Used

Text editor used was Vim. Development was done on the Linux OS.

4.1.2 Component Overview

Testing Suite for C++ programs.

4.1.3 Phase Overview

Initial phase for testing suite. Suite runs a single executable compiled from a *.cpp file against *.tst files.

4.1.4 Architecture Diagram

Compile a *.cpp file passed as a command line argument. Run that executable against all *.tst files within the current directory and all of its sub-directories.

4.1.5 Data Flow Diagram

Input is redirected from the *.tst file and output is redirected to a *.out file. The *.out file is compared to the corresponding *.ans file. The results are printed in a *.log file.

4.1.6 Design Details

4.2 Major Component #2

4.2.1 Technologies Used

4.2.2 Component Overview

4.2.3 Phase Overview

4.2.4 Architecture Diagram

4.2.5 Data Flow Diagram

4.2.6 Design Details

4.3 Major Component #3

4.3.1 Technologies Used

4.3.2 Component Overview

4.3.3 Phase Overview

4.3.4 Architecture Diagram

4.3.5 Data Flow Diagram

4.3.6 Design Details

5

System and Unit Testing

5.1 Overview

Some of the functionality was independently tested apart from the completed program. The completed program was run against several test files in their respective directories.

5.2 Dependencies

The program requires *.tst and *.ans files that are either in the same directory as the *.cpp file or in a sub-directory of that directory.

5.3 Test Setup and Execution

Dr. Logar provided several test files and directories for the program to process. The executable was copied into each of the directories containing a *.cpp file to test against.

6

Development Environment

The basic purpose for this section is to give a developer all of the necessary information to setup their development environment to run, test, and/or develop.

6.1 Development IDE and Tools

The Linux package installed on our tablets curtiosy of the ACM.

6.2 Source Control

Git along with bitbucket was used for source control.

6.3 Dependencies

The chief dependency was each other. In order for one of us to move on to the next phase we had to have completed code from our partners. We also needed the test cases for our project in order to make sure it was operating accepably.

6.4 Build Environment

The build enviornment uses the gcc command to complie the program.

6.5 Development Machine Setup

If warranted, provide a list of steps and details associated with setting up a machine for use by a developer.

7

Release – Setup – Deployment

This program will be submitted as an assignment. Therefore there is no release or deployment. It is a standard C++ program which will complie and run on Linux traversing its way through the directory looking for test files to use on the computer program it is testing.

7.1 Deployment Information and Dependencies

Main dependencies are the Linux operating system, C++ programs, and a basic knowledge of command line compiling.

7.2 Setup Information

The project should reside in the directory or in a parent directory of where the programs and test files are which it will be used to test.

7.3 System Versioning Information

There will only be one version of this program.

8

User Documentation

8.1 User Guide

8.2 Installation Guide

8.3 Programmer Manual

9

Class Index

9.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

10

Class Documentation

10.1 Poly Class Reference

Public Member Functions

- Poly ()
- ~Poly ()
- int myfunction (int)

10.1.1 Constructor & Destructor Documentation

10.1.1.a Poly::Poly ( )

My constructor

10.1.1.b Poly::~Poly ( )

My destructor

10.1.2 Member Function Documentation

10.1.2.a int Poly::myfunction (int *a*)

my own example function fancy new function
new variable
The documentation for this class was generated from the following file:

- hello.cpp

Acknowledgement

Thanks

Supporting Materials

This document will contain several appendices used as a way to separate out major component details, logic details, or tables of information. Use of this structure will help keep the document clean, readable, and organized.

Sprint Reports

10.1 Sprint Report #1

10.2 Sprint Report #2

10.3 Sprint Report #3

Industrial Experience

10.4 Resumes

10.5 Industrial Experience Reports

10.5.1 Name1

10.5.2 Name2

10.5.3 Name3

Appendix

Latex sample file:

10.1 Introduction

This is a sample input file. Comparing it with the output it generates can show you how to produce a simple document of your own.

10.2 Ordinary Text

The ends of words and sentences are marked by spaces. It doesn't matter how many spaces you type; one is as good as 100. The end of a line counts as a space.

One or more blank lines denote the end of a paragraph.

Since any number of consecutive spaces are treated like a single one, the formatting of the input file makes no difference to TEX, but it makes a difference to you. When you use LATEX, making your input file as easy to read as possible will be a great help as you write your document and when you change it. This sample file shows how you can add comments to your own input file.

Because printing is different from typewriting, there are a number of things that you have to do differently when preparing an input file than if you were just typing the document directly. Quotation marks like "this" have to be handled specially, as do quotes within quotes: "'this' is what I just wrote, not 'that'".

Dashes come in three sizes: an intra-word dash, a medium dash for number ranges like 1–2, and a punctuation dash—like this.

A sentence-ending space should be larger than the space between words within a sentence. You sometimes have to type special commands in conjunction with punctuation characters to get this right, as in the following sentence. Gnats, gnus, etc. all begin with G. You should check the spaces after periods when reading your output to make sure you haven't forgotten any special cases. Generating an ellipsis ... with the right spacing around the periods requires a special command.

TEX interprets some common characters as commands, so you must type special commands to generate them. These characters include the following: $ & % # { and }.

In printing, text is emphasized by using an *italic* type style.

A long segment of text can also be emphasized in this way. Text within such a segment given additional emphasis with Roman *type. Italic type loses its ability to emphasize and become simply distracting when used excessively.*

It is sometimes necessary to prevent TEX from breaking a line where it might otherwise do so. This may be at a space, as between the "Mr." and "Jones" in "Mr. Jones", or within a word—especially when the word is a symbol like *itemnum* that makes little sense when hyphenated across lines.

Footnotes[1] pose no problem.

TEX is good at typesetting mathematical formulas like $x - 3y = 7$ or $a_1 > x^{2n}/y^{2n} > x'$. Remember that a letter like x is a formula when it denotes a mathematical symbol, and should be treated as one.

[1] This is an example of a footnote.

10.3 Displayed Text

Text is displayed by indenting it from the left margin. Quotations are commonly displayed. There are short quotations

> This is a short a quotation. It consists of a single paragraph of text. There is no paragraph indentation.

and longer ones.

> This is a longer quotation. It consists of two paragraphs of text. The beginning of each paragraph is indicated by an extra indentation.
>
> This is the second paragarph of the quotation. It is just as dull as the first paragraph.

Another frequently-displayed structure is a list. The following is an example of an *itemized* list.

- This is the first item of an itemized list. Each item in the list is marked with a "tick". The document style determines what kind of tick mark is used.
- This is the second item of the list. It contains another list nested inside it. The inner list is an *enumerated* list.
  1. This is the first item of an enumerated list that is nested within the itemized list.
  2. This is the second item of the inner list. LaTeX allows you to nest lists deeper than you really should.

  This is the rest of the second item of the outer list. It is no more interesting than any other part of the item.
- This is the third item of the list.

You can even display poetry.

> There is an environment for verse
> Whose features some poets will curse.
>
> For instead of making
> Them do *all* line breaking,
> It allows them to put too many words on a line when they'd rather be forced to be terse.

Mathematical formulas may also be displayed. A displayed formula is one-line long; multiline formulas require special formatting instructions.

$$x' + y^2 = z_i^2$$

Don't start a paragraph with a displayed equation, nor make one a paragraph by itself.

10.4 Build process

To build LaTeX documents you need the latex program. It is free and available on all operating systems. Download and install. Many of us use the TexLive distribution and are very happy with it. You can use a editor and command line or use an IDE. To build this document via command line:

```
alta>  pdflatex SystemTemplate
```

If you change the bib entries, then you need to update the bib files:

```
alta>  pdflatex SystemTemplate
alta>  bibtex SystemTemplate
alta>  pdflatex SystemTemplate
alta>  pdflatex SystemTemplate
```

Acknowledgement

Thanks to Leslie Lamport

Bibliography