ID-10-T 

SOFTWARE REQUIREMENTS SPECIFICATION

for

Software Metrics Calculation System

Version 1.1

Prepared by Christopher Silva
Anthony Enem
Nathan Durst
Da Dong
Shujing Zhang

Contents

Revision History

Version	Date	Description
1.0	February 4, 2017	Christopher Silva, Anthony Enem, Nathan Durst, Da Dong, and Shujing Zhang met at the Midwestern State University Library to work on the initial draft.
1.1	February 15, 2017	Christopher Silva made changes based on customer feedback.

1 Introduction

Dr. Stringfellow (hereafter referred to as the client), is interested in software that will help ensure that her computer science students are writing programs that fit her specifications. This software should calculate and display metrics about the users source code such as line of code, lines of documentation, the ratio of the two, etc.

1.1 Purpose

This document details the software requirements for the Software Metrics Calculation System (hereafter referred to as SMCS), which the Software Engineering group ID-10-T (hereafter also referred to as the team) has devised to assist in the software development process. The plan outlines the different areas of the project that must be addressed for successful development of the software. It establishes guidelines for resources that will be used in the project, and also points out additional resources that are needed. This plan addresses some of the risks involved in the project and the steps to correct those risks, if they occur. Also, quality assurance will be mentioned, and a glossary of terms used in this document is included.

1.2 Scope

The client wants SMCS to quickly calculate code metrics on student source code. The client currently spends an excessive amount of time looking for issues that could be solved if students had software to point them out. The client would like SMCS to support C++ and/or Java source code. The client would like SMCS to be easily extensible in the future to allow for more types of metrics or languages.

1.3 Main Objective

The main objective of SMCS is to help first and second year computer science students become better programmers by giving them a tool that will point out some frequent simple mistakes that they make.

1.4 Overview of Document

The remainder of the document is intended to inform the client of the intended system. Hardware and software requirements, major users, both major and minor functions, constraints, and intended user interface are described.

2 Users

This section will define who the users of SMCS are and give details on the use case of the system.

2.1 Who are the Users?

The principal users of the SMCS are students taken freshmen or sophomore level computer science courses. Due to the SMCS being able to analyze only C++ and/or Java source code files, the users would also need to be familiar with the language(s) accepted by the SMCS. Other users might include upper level CS students who need some analysis on their source code written in the specified language, or professors who might want to integrate the SMCS as part of their grading system to analyze student's source code written in programming languages accepted by SMCS.

2.2 Use Cases

SMCS has one use case in which the system analyzes a users source code file. This use case is shown in figure 2.1.

1. User double clicks the SMCS' executable file and that launches the SMCS web interface on the computer's default browser.
2. User loads file(s) by clicking on an open file button and selecting file(s) from an Open file dialog OR by dragging and dropping source code file(s) onto a drag-and-drop panel on the SMCS interface.
3. User chooses programming language of uploaded source code. file(s) in one of two ways:
   a) User accepts SMCS' automatically detected programming language from the file(s) extension.
   b) User selects file(s) extension of source code file(s) from a drop-down menu of accepted languages if automatically detected language is incorrect.
4. User selects the metrics they wish to use.
5. User clicks a button for SMCS to begin source code analysis.
6. SMCS analysis

a) Main success scenario:
   i. SMCS successfully completes analysis of uploaded source code file(s).
   ii. SMCS displays results from analysis of uploaded file(s) to the user

b) Incorrect source code file scenario :
   i. File(s) uploaded by user is not of the selected programming language. SMCS displays an error of this scenario and prompts the user to select the correct programming language.

7. User closes the SMCS application.

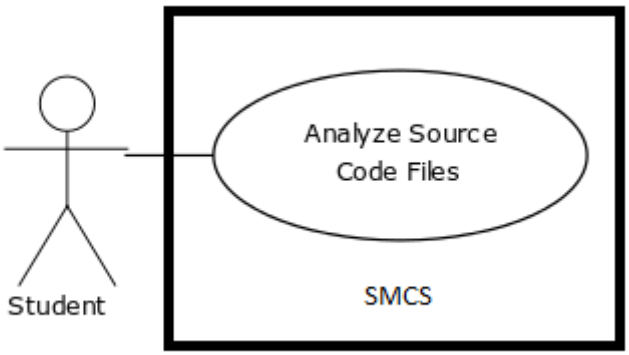

Figure 2.1: Use Case Diagram

3 System

This section will define the development and target environments, the functional and non-functional requirements, and the user interface.

3.1 Development and Target Environments

The team will program SMCS in Go using Gogland as the IDE. The team will use Git for version control, Slack for communications, and LaTeX for documentation. The target environments of SMCS are Windows XP or later, Linux, and Mac OS X 10.7 or later.

3.2 Functional Requirements

Allow the user to load a source code file and view metrics about their code.
Support the following metrics:

- Lines of Code (LOC) - The number of lines of code.
- Lines of Documentation (LOD) - The number of lines of documentation.
- Ration of LOC to LOD - This will be used to tell the user if there is too little documentation.
- Blank Lines - The number of blank lines. This will be used to tell a user if there is not enough whitespace.
- Total Lines - The total number of lines.
- Number of Functions - The number of functions in a file.
- Number of Function Parameters - The number of parameters in each function. This will be used to warn a user if there is an excessive amount of parameters in a function.
- Number of Non-void Functions - The number of functions that do not have a void return type.
- Methods Per Class - The number of methods in each class.
- Lines Per Function - The number of lines in each function.
- Cyclomatic complexity - The number of linearly independent paths within a program.

3.3 User Interface

The SMCS user interface will consist of two screens shown in figure 3.1 and 3.2. The first will allow the user to submit source code file(s) to be analyzed and the second will display the results of the analysis. The submission screen will have a file selection button, a drop-down list of supported languages, an area to select which metrics the user wishes to use, and a submission button that will transfer the user to the results screen. The results screen will display a list of metrics that the user selected followed by a section for each metric. The list of metrics will be links that will scroll to that section.

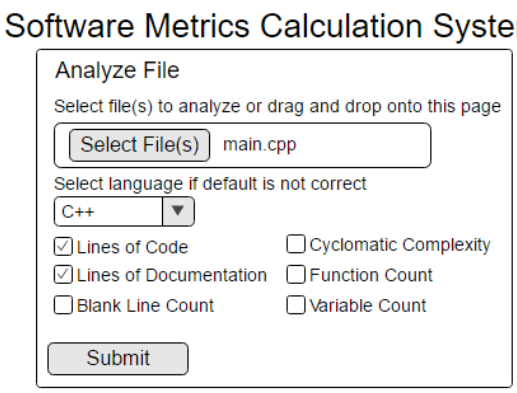

Figure 3.1: Source code selection interface

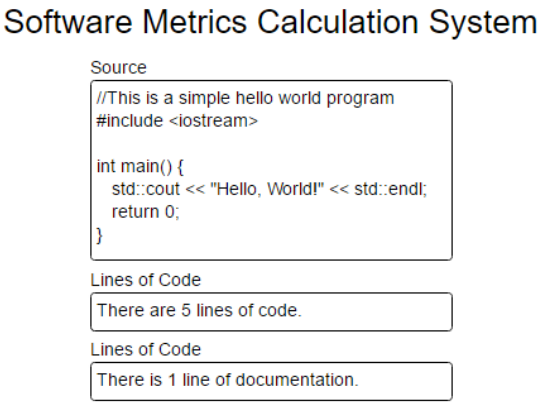

Figure 3.2: Source code metrics interface

3.4 Non-functional Requirements

- Programmed in an OOP language to allow for future extension of functionality.
- Initially support C++ and/or Java.
- Run on Windows XP or later.

4 Risks

The development process of the SMCS involves several risks. All risks should be identified and action should be taken to reduce these risks to an acceptable level. Development risks include project knowledge, team member turnover, requirement adjustments, and changes in the specifications.

4.1 Possible Risks

The risks involved in the development of SMCS are that the developers may not have a complete understanding of the entire metric calculation system in general. Lack of experience with parsing files or using the designated programming language plays a factor in the development process. Also, the risk of team member turnover can involve a member of our team withdrawing from the course, inevitably withdrawing from the project. Additionally, the requirements set by the client can change unpredictably if the client sees issues with the current requirements of the system during the time of development. Another risk is that the client, or team, may change specifications during the development process.

4.2 Risk Managment

Risk	Solution
Project Knowledge	Group meetings weekly as well as constant communication between the team using applications such as Slack and GitHub so the team is all on the same page.
Team Member Turnover	Team members must discuss potential drop with the rest of the team so plans can be made.
Requirements Change	Constant communication between the client and a team member to make sure that the information related to the requirements is collected accurately.
Specifications Change	The client and all team members will be constantly updated on status of the project through each step of the development process.

5 Schedule

The schedule will identify the steps needed to complete the project.

5.1 Project Breakdown

1. Obtain information from client pertaining to project details.
2. Compose an outline of the requirements.
3. Develop user interface diagram.
4. Create prototype of SMCS.
5. Test prototype.
6. Make corrections to the prototype.
7. Develop user manual.
8. Submit final product to the client.

Repeat steps 5 and 6 when necessary.

6 Glossary

- Client - Dr. Catherine Stringfellow
- Git - Version control software
- Go - OOP language developed by Google
- Gogland - Go IDE developed by JetBrains
- IDE - Integrated Development Environment
- LaTeX - Markdown language used for creating high quality documents
- OOP - Object Oriented Programming
- Slack - Group communication software used by developers
- SMCS - Software Metrics Calculation System
- Team - Software Engineering team ID-10-T Comprised of Christopher Silva, Anthony Enem, Nathan Durst, Da Dong, and Shujing Zhang

7 References

- Go Documentation golang.org
- LaTeX Documentation latex-project.org