**Software Development Plan (SDP)**

*Document Number: SDP02*

*Date: 10/27/16*

***MU GitHub Analyzer***

*Liam Innes*

*Swarna Muralidharan*



Professor Edwin Torres

Software Engineering Department

Monmouth University

West Long Branch, NJ 07764-1898

**Table of Contents**

**1. SCOPE**

**1.1. Identification**

**1.2. System Overview**

**1.3. Document Overview**

**1.4. Relationship to Other Plans**

**2. REFERENCED DOCUMENTS**

**3. OVERVIEW OF REQUIRED WORK**

**4. PLANS FOR PERFORMING GENERAL SOFTWARE DEVELOPMENT ACTIVITIES**

**4.1. Software Development Process**

**4.2. General Plans for Software Development**

4.2.1. Software Development Methods

4.2.2. Standards for Software Products

4.2.3. Reusable Software Products

4.2.4. Handing of Critical Requirements

4.2.5. Computer Hardware Resource Utilization

4.2.6. Recording Rationale

4.2.7. Access for Acquirer Review

**5. PLANS FOR PERFORMING DETAILED SOFTWARE DEVELOPMENT ACTIVITIES**

**5.1. Project Planning and Oversight**

5.1.1. Software Development Planning (covering updates to this plan)

5.1.2. Software Item Test Planning

5.1.3. System Test Planning

5.1.4. Software Installation Planning

5.1.5. Software Transition Planning

5.1.6. Following and Updating Plans (including the intervals for management review)

**5.2. Establishing a Software Development Environment**

5.2.1. Software Engineering Environment

5.2.2. Software Test Environment

5.2.3. Software Development Library

5.2.4. Software Development Files

5.2.5. Non-Deliverable Software

**5.3. System Requirements Analysis**

5.3.1. Analysis of User Input

5.3.2. Operational Concept

5.3.3. System Requirements

**5.4. System Design**

5.4.1. System-Wide Design Decisions

5.4.2. System Architectural Design

**5.5. Software Requirements Analysis**

**5.6. Software Design**

5.6.1. Software Item-Wide Design Decisions

5.6.2. Software Item Architectural Design

5.6.3. Software Item Detailed Design

**5.7. Software Implementation and Unit Testing**

5.7.1. Software Implementation

5.7.2. Preparing for Unit Testing

5.7.3. Performing Unit Testing

5.7.4. Revision and Re-testing

5.7.5. Analyzing and Recording Unit Test Results

**5.8. Unit Integration and Testing**

5.8.1. Preparing for Unit Integration and Testing

5.8.2. Performing Unit Integration and Testing

5.8.3. Revision and Re-testing

5.8.4. Analyzing and Recording Unit Integration and Test Results

**5.9. Software Item Qualification Testing**

5.9.1. Independence in Software Item Qualification Testing

5.9.2. Testing on the Target Computer System

5.9.3. Preparing for Software Item Qualification Testing

5.9.4. Dry Run of Software Item Qualification Testing

5.9.5. Performing Software Item Qualification Testing

5.9.6. Revision and Re-testing

5.9.7. Analyzing and Recording Software Item Qualification Test Results

**5.10. Software/Hardware Item Integration and Testing**

5.10.1. Preparing for Software/Hardware Item Integration and Testing

5.10.2. Performing Software/Hardware Item Integration and Testing

5.10.3. Revision and Re-testing

5.10.4. Analyzing and Recording Software/Hardware Item Integration and Test Results

**5.11. System Qualification Testing**

5.11.1. Independence in System Qualification Testing

5.11.2. Testing on the Target Computer System

5.11.3. Preparing for System Qualification Testing

5.11.4. Dry Run of System Qualification Testing

5.11.5. Performing System Qualification Testing

5.11.6. Revision and Re-testing

5.11.7. Analyzing and Recording System Qualification Test Results

**5.12. Preparing for Software Use**

5.12.1. Preparing the Executable Software

5.12.2. Preparing Version Descriptions for User Sites

5.12.3. Preparing User Manuals

5.12.4. Installation at User Sites

**5.13. Preparing for Software Transition**

5.13.1. Preparing the Executable Software

5.13.2. Preparing Source Files

5.13.3. Preparing Version Descriptions for the Support Site

5.13.4. Preparing the “As Built” Software Item Design and Other Software Support Information

5.13.5. Updating the System Design Description

5.13.6. Preparing Support Manuals

5.13.7. Transition to the Designated Support Site

**5.14. Software Configuration Management**

5.14.1. Configuration Identification

5.14.2. Configuration Control

5.14.3. Configuration Status Accounting

5.14.4. Configuration Audits

5.14.5. Packaging, Storage, Handling, and Delivery

**5.15. Software Product Evaluation**

5.15.1. In-Process and Final Software Product Evaluations

5.15.2. Software Product Evaluation Records (including items to be recorded)

5.15.3. Independence in Software Product Evaluation

**5.16. Software Quality Assurance**

5.16.1. Software Quality Assurance Evaluations

5.16.2. Software Quality Assurance Records (including items to be recorded)

5.16.3. Independence in Software Quality Assurance

**5.17. Corrective Action**

5.17.1. Problem/Change Reports

5.17.2. Corrective Action System

**5.18. Joint Technical and Management Reviews**

5.18.1. Joint Technical Reviews (including a proposed set of reviews)

5.18.2. Joint Management Reviews (including a proposed set of reviews)

**5.19. Other Software Development Activities**

5.19.1. Risk Management (including known risks and corresponding strategies)

5.19.2. Software Management Indicators (including indicators to be used)

5.19.3. Security and Privacy

5.19.4. Subcontractor Management

5.19.5. Interface with Software Independent Verification and Validation (IV&V) Agents

5.19.6. Coordination with Associate Developers

5.19.7. Improvement of Project Processes

5.19.8. Other Activities Not Covered Elsewhere in this Plan

**6. SCHEDULES AND ACTIVITY NETWORK**

**7. PROJECT ORGANIZATION AND RESOURCES**

**7.1. Project Organization**

**7.2. Project Resources**

**8. NOTES**

# **1 Scope**

This section describes the development plan for MU GitHub Analyzer - a Java based application to analyse projects attributes related to software development.

## **1.1 Identification**

This document applies to version 1.0 of MU GitHub Analyzer.

## **1.2 System Overview**

The MU GitHub Analyzer will list all open source projects using GitHub API, compile project attributes for each of these project and build analytical function to derive metrics useful for the user. Common software quality measurement tools will be run against the GitHub public project codebase to generate metrics. The concept is driven by Mike Bush and it is being developed by Liam Innes and Swarna Muralidharan in a Software Engineering Practicum course at Monmouth University. Mike will provide support to decide on the requirements.

## **1.3 Document Overview**

This document describe the software system to be implemented using waterfall SDLC model.

## **1.4 Relationship to Other Plans**

The proposal document has multiple sections plotting out the development of the Software Requirements Specification, the Software Development Plan, the Software Design Description, and the Master Test Plan.

# **2 Referenced Documents**

N/A

# **3 Overview of Required Work**

GitHub is a web based Git repository hosting service which offers all the distributed revision control and source code management functionality of Git. GitHub provides several collaboration features such as access control, wiki, task management, feature requests, bug tracking for every project. GitHub supports both public and private repositories. Many open source projects are hosted as public repositories in GitHub. MU GitHub analyzer will list all open source projects, compile project attributes for each of these projects and build analytical functions to derive metrics useful for Software Engineering researchers. Common software quality measurement tools will be run against the GitHub public project codebase to generate metrics.

The project attributes to be captured from Git projects are

* Number of contributors to a project.
* Number of pull requests for a project.
* Number of fork requests for a project.
* Number of watchers for a project.
* Number of releases for a project.
* Number of commits for a project.
* LOC
* Number of issues reported.

The attributes will be persisted using a open source RDBMS and various derived metrics will be computed based on end user needs. The project will be a new development effort that uses the GitHub API to interface with GitHub.

MU GitHub Analyzer will be implemented as a standalone java application.

# **4 Plans for Performing General Software Development Activities**

## The following activities has been identified and reviewed to

* Build connection to GitHub with authentication using GitHub API
* Retrieve GitHub public project attributes using GitHub API
* Compute derived metrics
* Design schema for RDBMS to persist project attributes
* Build connection to RDBMS
* Persist attributes of projects in RDBMS tables
* Build UI to query project attributes

## **4.1 Software Development Process**

All software will be implemented using Java and GitHub API. Each will have an associated unit test. Functional and Nonfunctional tests will be documented and exercised upon successful integration builds.

## **4.2 General Plans for Software Development**

During the process of software development, the team members will follow the rules related to Waterfall SDLC model.

### **4.2.1 Software Development Methods**

A Waterfall SDLC model will be followed while building the software. We will begin with functional and nonfunctional requirements identification and documentation. The customer will approve of the Software Requirement Specification document. This will be followed by Architecture design and Software Design discussion and documentation. Implementation will follow with testing to validate the implementation. The software will be available for use after successful testing.

### **4.2.2 Standards for Software Products**

All implementation will follow Google Java coding style guidelines for coding standards related to formatting, Javadoc compatible comments, naming convention, coding practices, etc.(https://google.github.io/styleguide/javaguide.html)

All Programming Language constructs or features must be open source.

### **4.2.3 Reusable Software Products**

MU GitHub Analyzer software can be potentially reused for research related to open source software development practices.

#### **4.2.3.1 Incorporating Reusable Software Products**

GitHub API will be used to interface with GitHub and retrieve project attributes. JDBC will be used to interface with RDBMS. An open Source RDBMS like MySql will be used.

#### **4.2.3.2 Developing Reusable Software Products**

N/A

### **4.2.4 Handling of Critical Requirements**

The critical requirements for this project are to

* Build an interface using GitHub API to download public project metrics. GitHub API is well documented and is used extensively by developers to build applications
* Build an interface using JDBC library to persist and query GitHub metrics. JDBC API is well documented and is being used in a wide range of applications across enterprises.
* Build an User Interface to allow the user to interact with the system.

#### **4.2.4.1 Safety Assurance**

N/A

#### **4.2.4.2 Security Assurance**

N/A

#### **4.2.4.3 Privacy Assurance**

N/A

#### **4.2.4.4 Assurance of Other Critical Requirements**

N/A

### **4.2.5 Computer Hardware Resource Utilization**

Our workstations will only comprise of our individual devices. No outsider servers will be used.

The MU GitHub Analyzer is the simple java application that requires internet connectivity to download GitHub metrics and connectivity to RDBMS to persist and query the metrics.

### **4.2.6 Recording Rationale**

N/A

### **4.2.7 Access for Acquirer Review**

N/A

# **5 Plans for Performing Detailed Software Development Activities**

The following paragraphs describe methods, procedures, and/or tools for the software development activities that will occur over the course of the project. The small amount of personnel and lack of financial stakes contribute to why some sections are not applicable.

## **5.1 Project Planning and Oversight**

### **5.1.1 Software Development Planning (covering updates to this plan)**

The development team has a schedule to adhere to, outlined in section 6, that entails the documentation of planning for the software, including this document and a later draft of it.

### **5.1.2 Software Item Test Planning**

N/A

### **5.1.3 System Test Planning**

System test cases will be written to validate the Software Requirements Specification.

### **5.1.4 Software Installation Planning**

The development team will arrange the terms of installing the software with the customer.

### **5.1.5 Software Transition Planning**

N/A

### **5.1.6 Following and Updating Plans (including the intervals for management review)**

N/A

## **5.2 Establishing a Software Development Environment**

### **5.2.1 Software Engineering Environment**

Code will be shared with GitHub using GitHub desktop. Eclipse will be installed on our machines so that the code will be written in Java. The GUI will be built with JavaFX.

### **5.2.2 Software Test Environment**

Unit testing and Integration testing will be performed in Eclipse. System testing will be performed on a machine running Windows 10.

### **5.2.3 Software Development Library**

N/A

### **5.2.4 Software Development Files**

The system will comprise of Java 8 class files according to section 5.4.2, System Architectural Design.

### **5.2.5 Non-Deliverable Software**

N/A

## **5.3 System Requirements Analysis**

### **5.3.1 Analysis of User Input**

User input will be analyzed by constructing use cases. The goal is to require very little effort on behalf of the user. Results will be recorded in the Software Requirements Specification.

### **5.3.2 Operational Concept**

The operational concept will be provided by the customer and reinterpreted by the team. The current concept is that the MU GitHub Analyzer receives parameters to crawl GitHub and find matching repositories. When it has collected a candidate pool of repositories, it can compare the data sets collected from them to produce analytical data. This will be more fleshed out in the Software Requirements Specification.

### **5.3.3 System Requirements**

The application is expected to be platform independent because it will be written in Java, however system requirements will be written primarily so that they can be validated with the development team’s equipment that is not out of reach for the customer to replicate. System Requirements will be recorded in Section 3 of the Software Requirements Specification.

## **5.4 System Design**

### **5.4.1 System-Wide Design Decisions**

* The software will be written in Java.
* The software will be executable on any personal computer that runs Java.
* MySQL 5.7.16 will be used to store the collected data.
* The GUI will be built with JavaFX 8.

### **5.4.2 System Architectural Design**

TBD, however we currently have plans to include a class to retrieve data from GitHub and a class for the UI.

## **5.5 Software Requirements Analysis**

Requirements will be construed from discussions with the customer and edited to fit our team’s ability, and then they will be presented to the customer for approval. If there are aspects of the requirements that are not approved, they will be negotiated or fixed until approval is met. A Software Requirements Specification document will be produced at the end of this process. The SRS will also include a requirements traceability matrix that groups each requirement in with either the user interface, database, or GitHub interface.

## **5.6 Software Design**

### **5.6.1 Software Item-Wide Design Decisions**

Item-Wide Design Decisions will be formed in collaboration with the team and the customer and finalized in the Software Design Description.

### **5.6.2 Software Item Architectural Design**

The Software Item Architectural Design will be created by the development team and finalized in the Software Design Description.

### **5.6.3 Software Item Detailed Design**

The Software Item Detailed Design will be written by the development team and finalized in the Software Design Description.

## **5.7 Software Implementation and Unit Testing**

### **5.7.1 Software Implementation**

The software will be implemented on computers running Java 8.

### **5.7.2 Preparing for Unit Testing**

Prior to Unit Testing, test cases will be written.

### **5.7.3 Performing Unit Testing**

Unit testing will be based on test cases and performed in JUnit. The test cases and their results will be added to reports.

### **5.7.4 Revision and Re-testing**

When a critical bug is found and requires changing, the code will be modified so as to resolve that bug and not introduce other critical bugs. When that modification is complete, the unit test case will be repeated.

### **5.7.5 Analyzing and Recording Unit Test Results**

Unit test results will be compiled into a spreadsheet.

## **5.8 Unit Integration and Testing**

### **5.8.1 Preparing for Unit Integration and Testing**

The tester will have the latest updates to the relevant units. Test cases will be written.

### **5.8.2 Performing Unit Integration and Testing**

Integration testing will be based on test cases and performed in JUnit. The test cases and their results will be added to reports.

### **5.8.3 Revision and Re-testing**

When a critical bug is found and requires changing, the code will be modified so as to resolve that bug and not introduce other critical bugs. When that modification is complete, the integration test case will be repeated.

### **5.8.4 Analyzing and Recording Unit Integration and Test Results**

Integration test results will be compiled into a spreadsheet.

## **5.9 Software Item Qualification Testing**

### **5.9.1 Independence in Software Item Qualification Testing**

N/A

### **5.9.2 Testing on the Target Computer System**

N/A

### **5.9.3 Preparing for Software Item Qualification Testing**

N/A

### **5.9.4 Dry Run of Software Item Qualification Testing**

N/A

### **5.9.5 Performing Software Item Qualification Testing**

N/A

### **5.9.6 Revision and Re-testing**

N/A

### **5.9.7 Analyzing and Recording Software Item Qualification Test Results**

N/A

## **5.10 Software/Hardware Item Integration and Testing**

### **5.10.1 Preparing for Software/Hardware Item Integration and Testing**

N/A

### **5.10.2 Performing Software/Hardware Item Integration and Testing**

N/A

### **5.10.3 Revision and Re-testing**

N/A

### **5.10.4 Analyzing and Recording Software/Hardware Item Integration and Test Results**

N/A

## **5.11 System Qualification Testing**

### **5.11.1 Independence in System Qualification Testing**

N/A

### **5.11.2 Testing on the Target Computer System**

N/A

### **5.11.3 Preparing for System Qualification Testing**

N/A

### **5.11.4 Dry Run of System Qualification Testing**

N/A

### **5.11.5 Performing System Qualification Testing**

N/A

### **5.11.6 Revision and Re-testing**

N/A

### **5.11.7 Analyzing and Recording System Qualification Test Results**

N/A

## **5.12 Preparing for Software Use**

### **5.12.1 Preparing the Executable Software**

The Software will be an executable JAR file that can run on any machine with Java 8. Prepared by the development team.

### **5.12.2 Preparing Version Descriptions for User Sites**

N/A because the customer will receive one version created at the end of one waterfall cycle.

### **5.12.3 Preparing User Manuals**

When an executable for the software is produced, the development team will write a user manual to help unfamiliar users understand and learn how to use the product.

### **5.12.4 Installation at User Sites**

The customer will be sent the executable JAR file and User Manual.

## **5.13 Preparing for Software Transition**

### **5.13.1 Preparing the Executable Software**

The development team will make the software so that it can be executed by opening the executable JAR file.

### **5.13.2 Preparing Source Files**

The development team will package the source files in the executable JAR file.

### **5.13.3 Preparing Version Descriptions for the Support Site**

N/A

### **5.13.4 Preparing the “As Built” Software Item Design and Other Software Support Information**

The development team will rewrite the architecture design to reflect the final product once it is completed, if there are differences.

### **5.13.5 Updating the System Design Description**

If changes need to be made to the System Design Description, the development team will make a new version of the document with those changes.

### **5.13.6 Preparing Support Manuals**

TBD

### **5.13.7 Transition to the Designated Support Site**

N/A

## **5.14 Software Configuration Management**

### **5.14.1 Configuration Identification**

N/A

### **5.14.2 Configuration Control**

N/A

### **5.14.3 Configuration Status Accounting**

N/A

### **5.14.4 Configuration Audits**

N/A

### **5.14.5 Packaging, Storage, Handling, and Delivery**

N/A

## **5.15 Software Product Evaluation**

### **5.15.1 In-Process and Final Software Product Evaluations**

Software product evaluations will be done by the instructor of the Software Engineering Practicum course.

### **5.15.2 Software Product Evaluation Records (including items to be recorded)**

N/A

### **5.15.3 Independence in Software Product Evaluation**

N/A

## **5.16 Software Quality Assurance**

### **5.16.1 Software Quality Assurance Evaluations**

N/A

### **5.16.2 Software Quality Assurance Records (including items to be recorded)**

N/A

### **5.16.3 Independence in Software Quality Assurance**

N/A

## **5.17 Corrective Action**

### **5.17.1 Problem/Change Reports**

N/A

### **5.17.2 Corrective Action System**

N/A

## **5.18 Joint Technical and Management Reviews**

### **5.18.1 Joint Technical Reviews (including a proposed set of reviews)**

N/A

### **5.18.2 Joint Management Reviews (including a proposed set of reviews)**

N/A

## **5.19 Other Software Development Activities**

### **5.19.1 Risk Management (including known risks and corresponding strategies)**

N/A

### **5.19.2 Software Management Indicators (including indicators to be used)**

N/A

### **5.19.3 Security and Privacy**

N/A

### **5.19.4 Subcontractor Management**

N/A

### **5.19.5 Interface with Software Independent Verification and Validation (IV&V) Agents**

N/A

### **5.19.6 Coordination with Associate Developers**

N/A

### **5.19.7 Improvement of Project Processes**

N/A

### **5.19.8 Other Activities Not Covered Elsewhere in this Plan**

TBD

# **6 Schedules and Activity Network**

The table below shows when the tasks will be initiated and delivered throughout the SDLC process.

|  |  |  |  |
| --- | --- | --- | --- |
| Task # | Task name | Estimated completion date | Task dependencies |
| 1 | Discuss requirements with customer |  |  |
| 2 | Draft Software Requirement Specification | 10/13/2016 | 1 |
| 3 | Customer sign off on Software Requirement Specification | 11/03/2016 | 2 |
| 4 | Draft System Architectural Design | 11/10/2016 | 3 |
| 5 | Finalize System Architectural Design | 12/01/2016 | 3, 4 |
| 6 | Draft Software Design Description | 12/01/2016 | 3, 5 |
| 7 | Finalize Software Design Description | 12/15/2016 | 3, 5, 6 |
| 8 | Master test Plan | TBD | 3, 7 |
| 9 | Software User Manual | TBD | 1, 3 |
| 10 | Installation Manual | TBD | 7 |
| 11 | Software Development | TBD | 3, 5, 7 |
| 12 | Initiate Testing | TBD | 8, 11 |
| 13 | Install software at customer site | TBD | 10, 11 |
| 14 | Customer demonstration | TBD | 11, 13 |

# **7 Project Organization and Resources**

This section describes the resources allotted to MU GitHub Analyzer project and their roles and responsibilities towards this project.

## **7.1 Project Organization**

This project consists of two team members and all the responsibility related to this project will be shared between them equally.

## **7.2 Project Resources**

This project will be developed by a team of two developers who will also be sharing the responsibility of project manager and architect.

# **8 Notes**

**8.1 Glossary**

|  |  |
| --- | --- |
| API | Application programming interface |
| GUI / UI | Graphical user interface |
| JDBC | Java database connectivity |
| LOC | Lines of code |
| MU | Monmouth University |
| N/A | Not applicable. |
| RDBMS | Relational database management system |
| TBD | To be decided. |