**Software Requirements Specification (SRS)**

*Document Number: SRS03*

*Date: 5/1/2017*

***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**

**2. REFERENCED DOCUMENTS**

**3. REQUIREMENTS**

**3.1. Required States and Modes**

**3.2. Software Item Capability Requirements**

3.2.x. [Software Item Capability]

**3.3. Software Item External Interface Requirements**

3.3.1. Interface Identification and Diagrams

3.3.x. [Project-Unique Identifier of Interface]

**3.4. Software Item Internal Interface Requirements**

**3.5. Software Item Internal Data Requirements**

**3.6. Adaptation Requirements**

**3.7. Safety Requirements**

**3.8. Security and Privacy Requirements**

**3.9. Software Item Environment Requirements**

**3.10. Computer Resource Requirements**

3.10.1. Computer Hardware Requirements

3.10.2. Computer Hardware Resource Utilization Requirements

3.10.3. Computer Software Requirements

3.10.4. Computer Communications Requirements

**3.11. Software Quality Factors**

**3.12. Design and Implementation Constraints**

**3.13. Personnel-Related Requirements**

**3.14. Training-Related Requirements**

**3.15. Logistics-Related Requirements**

**3.16. Other Requirements**

**3.17. Packaging Requirements**

**3.18. Precedence and Criticality of Requirements**

**4. QUALIFICATION PROVISIONS**

**5. REQUIREMENTS TRACEABILITY**

**6. NOTES**

# 1 Scope

## 1.1 Identification

This document applies to version 1.0 of the MU GitHub Analyzer.

## 1.2 System Overview

The MU GitHub Analyzer (MUGHA) will list all open source projects using GitHub API, compile project attributes for each of those projects and build analytical functions to derive metrics like number of issues per contributor, number of issues per 1000 lines of code, number of lines of code per contributor. 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 describes the functional and nonfunctional requirements we intend for the MU GitHub Analyzer to fulfill. Numbered points may be referred to by placing the number of their header beforehand, E.G. in section 3.1 the first requirement is requirement 3.1.1. and the seventh sub requirement of the second requirement in section 3.2.1 is requirement 3.2.1.2.7.

# 2 Referenced Documents

N/A

# 3 Requirements

## 3.1 Required States and Modes

1. The MU GitHub Analyzer shall have a single interactive user interface.

## 3.2 Software Item Capability Requirements

MU GitHub Analyzer shall have the following capabilities

* GitHub interface to download open source project attributes.
* GitHub interface to Relational Database Management System (RDBMS)
* UI to query attributes
* Generate derived analytics

### 3.2.1 Searching

1. The MU GitHub Analyzer shall be able to search GitHub Repositories by entering the username of the git projects in the text box.
2. The MU GitHub Analyzer shall allow the User to specify the GitHub Repositories to search for according to these parameters:
   1. Programming language(s).
   2. Number (range) of contributors.
   3. Number (range) of pull requests.
   4. Frequency (range) of issues.
   5. Number (range) of commits.
   6. Number (range) of releases.
   7. Date (range) of latest commit.
   8. Whether it has been unit tested.
3. Users shall be able to clear their search results.
4. Users shall be able to select individual search results.
5. Users shall be able to select all search results.
6. User shall be able to add their selected search results to a candidate list.
7. Users shall be able to remove search results from the list of search results.
8. The UI shall have a search menu.
   1. The UI shall have the option to enter the username and password to login to github account.
   2. The menu shall have areas to input search parameters.
   3. The menu shall have a Search button to initiate searching based on user name entered.
   4. The menu shall have an Add Selection to List button.
   5. The menu shall have a Remove Selection button.
9. The UI shall display search results with a unique identifier (likely the URL extension of its page on GitHub.com) as well as the information pertaining to the search parameters.

### 3.2.2 Managing Candidate Repositories

1. The User Interface shall display a list of all stored repositories and their attributes.
2. Users shall be able to export their candidate pools to their machines.
3. Users shall be able to create new candidate pools.
4. Users shall be able to open saved pools.
5. The UI shall display the candidate pool as a list of the repositories with the information carried over from the search list (3.2.1.12) as well as the metrics that can be found in static analysis.
6. Users shall be able to rearrange the list of candidate repositories by the values in their project metrics.
7. The UI shall have a Candidate pool menu.
   1. The menu shall have a Check For Updates button.
   2. The menu shall have a Remove Selection button.
   3. The menu shall have a Static Analysis button.
   4. The menu shall have an Import Local Repository button.
8. MUGHA shall be able to check if candidate repositories are up to date.
9. MUGHA shall display an annotation if candidate repositories are not up to date.
10. MUGHA shall be able to update candidate repositories during static analysis.
11. Users shall be able to select individual candidate repositories.
12. Users shall be able to select all candidate repositories.
13. Users shall be able to remove candidate repositories.

### 3.2.3 Downloading Open Source Project Attributes

1. MUGHA shall import repository information from selected search results to a database.

### 3.2.4 GitHub Interface to RDBMS

1. MUGHA shall create tables in a relational database to capture project attributes and derived metrics.
2. MUGHA shall establish a connection to the database.
3. MUGHA shall insert all fetched attributes of GitHub projects in a table in the relational database.
4. MUGHA shall update all fetched attributes of GitHub projects in a table in the relational database.

### 3.2.5 Querying Attributes

1. MUGHA shall perform static analysis on stored repositories to determine the following attributes for the project at each iteration of the main branch of the repository:
   1. Total Source Lines of Code
   2. Number of Methods per Class
   3. Cyclomatic Complexity
2. Stored repositories shall be given the following attributes:
   1. Project URL
   2. Programming language(s)
   3. Number of contributors
   4. Number of pull requests
   5. Frequency of issues
   6. Number of commits
   7. Number of releases
   8. Date of latest commit
   9. Whether it has been unit tested.
   10. Total Source Lines of Code
   11. Number of Methods per Class
   12. Cyclomatic Complexity
3. Users shall be able to select repositories to perform static analysis on.
4. MUGHA shall take files from the main branch of the repository and save them on the User’s machine while performing static analysis.
5. MUGHA shall be able to delete repositories from the User's machine after static analysis.
6. Users shall have the option to delete repositories from their machine after static analysis.
7. Users shall be able to include local Git repositories from outside GitHub in their candidate pool.
8. MUGHA shall be able to perform static analysis on local Git repositories from outside GitHub.
9. Users shall be able to enter the following metrics for local repositories in the candidate list: Project ID, Number of contributors, Number of pull requests, Frequency of issues, Number of commits, Number of releases, Date of latest commit, Whether it has been unit tested.
10. MUGHA shall be able to detect if another branch from main was the central source of a repository's activity.
11. MUGHA shall provide an annotation if the main branch was not the central source of a repository's activity.
12. Users shall be able to make MUGHA analyze other branches if provided.

### 3.2.6 Generating Derived Analytics

1. Users shall be able to generate scatter plots with data from the collected repositories
   1. Users shall be able to select from the different numerical attributes to set them as axis on the scatter plots

## 3.3 Software Item External Interface Requirements

### 3.3.1 Interface Identification and Diagrams

The MU GitHub Analyzer must interface with the device running it and GitHub so as to search, retrieve and store repository data. This amounts to two interfaces, the Repository Crawl Interface between MUGHA and GitHub, and the Database Storage Interface between MUGHA and the database.



### 3.3.2 Repository Crawl Interface

The Repository Crawl Interface is the interface of the MU GitHub Analyzer and GitHub that occurs when searching for repositories that match the user’s input.

1. MUGHA shall receive the following information through this interface:
   1. The project files in every iteration of the main branch, organized as such.
   2. A unique identifier (likely the URL extension of its page on GitHub.com)
   3. The number of contributors.
   4. The number of pull requests.
   5. The number of commits.
   6. Date of latest commit.
2. The Repository Crawl Interface shall retrieve the information in one instant.
3. The Repository Crawl Interface shall store the information in the user’s database.
4. The Repository Crawl Interface shall be facilitated by an internet connection.

### 3.3.3 Database Storage Interface

The Database storage Interface is the interface of the MU GitHub Analyzer and the user’s database that occurs during storage of repositories.

1. The Database Storage Interface shall be a storage and retrieval of data type interface.
2. The Database shall receive the following information through this interface:
   1. The organization of files in every iteration of the main branch.
   2. A unique identifier (likely the URL extension of its page on GitHub.com)
   3. The number of contributors.
   4. The number of pull requests.
   5. The number of commits.
   6. Date of latest commit.
3. MUGHA shall retrieve the same information from the Database Storage Interface as well after transferring it there.

## 3.4 Software Item Internal Interface Requirements

TBD. All internal interfaces are left to the design.

## 3.5 Software Item Internal Data Requirements

TBD. Internal data requirements are determined by RDBMS table design.

## 3.6 Adaptation Requirements

N/A

## 3.7 Safety Requirements

N/A

## 3.8 Security and Privacy Requirements

N/A

## 3.9 Software Item Environment Requirements

1. The software should be able to run with minimal computing resources – 500MB of memory.

## 3.10 Computer Resource Requirements

### 3.10.1 Computer Hardware Requirements

1. MU GitHub Analyzer shall be able to run in any commodity hardware.

### 3.10.2 Computer Hardware Resource Utilization Requirements

1. MU GitHub Analyzer shall use at the most 2GB of memory.

### 3.10.3 Computer Software Requirements

1. MU GitHub Analyzer shall support Windows 10 OS’s with an open source RDBMS like MySQL.
2. MUGHA shall be supported by Java 8.

### 3.10.4 Computer Communications Requirements

1. MU GitHub Analyzer shall require internet access to connect to GitHub.

## 3.11 Software Quality Factors

1. The customer should be able to understand the MUGHA’s features in under 4 hours of initial use.

## 3.12 Design and Implementation Constraints

1. MUGHA shall be built only with open source tools and libraries.
2. MUGHA shall be designed to only support analysis of Java code.

## 3.13 Personnel-Related Requirements

N/A

## 3.14 Training-Related Requirements

N/A

## 3.15 Logistics-Related Requirements

N/A

## 3.16 Other Requirements

N/A

## 3.17 Packaging Requirements

1. This software shall be available as a digital package of jar files.

## 3.18 Precedence and Criticality of Requirements

Each requirement is of varying importance. The table below provides a list of the requirements by the number of their section and place on a list inside that section. Beside the requirement is a judgement as to where they fall along the MSCW method. Requirements labeled Must have, Should have, Could have, and Won’t have are of descending precedence in that order.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3.1.1 | Must have |  | 3.2.2.13 | Must have |  | 3.3.2.1 | Must have |
| 3.2.1.1 | Must have |  | 3.2.3.1 | Must have |  | 3.3.2.1.1 | Must have |
| 3.2.1.2.1 | Must have |  | 3.2.4.1 | Must have |  | 3.3.2.1.2 | Must have |
| 3.2.1.2.2 | Must have |  | 3.2.4.2 | Must have |  | 3.3.2.1.3 | Must have |
| 3.2.1.2.3 | Must have |  | 3.2.4.3 | Must have |  | 3.3.2.1.4 | Must have |
| 3.2.1.2.4 | Could have |  | 3.2.4.4 | Could have |  | 3.3.2.1.5 | Must have |
| 3.2.1.2.5 | Must have |  | 3.2.5.1 | Should have |  | 3.3.2.1.6 | Must have |
| 3.2.1.2.6 | Could have |  | 3.2.5.1.1 | Should have |  | 3.3.2.2 | Must have |
| 3.2.1.2.7 | Must have |  | 3.2.5.1.2 | Should have |  | 3.3.2.3 | Must have |
| 3.2.1.2.8 | Could have |  | 3.2.5.1.3 | Should have |  | 3.3.2.4 | Must have |
| 3.2.1.3 | Could have |  | 3.2.5.2 | Must have |  | 3.3.3.1 | Must have |
| 3.2.1.4 | Could have |  | 3.2.5.2.1 | Must have |  | 3.3.3.2 | Must have |
| 3.2.1.5 | Could have |  | 3.2.5.2.2 | Must have |  | 3.3.3.2.1 | Must have |
| 3.2.1.6 | Could have |  | 3.2.5.2.3 | Must have |  | 3.3.3.2.2 | Must have |
| 3.2.1.7 | Should have |  | 3.2.5.2.4 | Must have |  | 3.3.3.2.3 | Must have |
| 3.2.1.8 | Should have |  | 3.2.5.2.5 | Should have |  | 3.3.3.2.4 | Must have |
| 3.2.1.9 | Should have |  | 3.2.5.2.6 | Must have |  | 3.3.3.2.5 | Must have |
| 3.2.1.10 | Should have |  | 3.2.5.2.7 | Must have |  | 3.3.3.2.6 | Must have |
| 3.2.1.11 | Must have |  | 3.2.5.2.8 | Must have |  | 3.3.3.3 | Must have |
| 3.2.1.11.1 | Must have |  | 3.2.5.2.9 | Should have |  |  |  |
| 3.2.1.11.2 | Should have |  | 3.2.5.2.10 | Should have |  | 3.9.1 | Could have |
| 3.2.1.11.3 | Could have |  | 3.2.5.2.11 | Should have |  |  |  |
| 3.2.1.11.4 | Could have |  | 3.2.5.2.12 | Should have |  | 3.10.1.1 | Could have |
| 3.2.1.11.5 | Should have |  | 3.2.5.3 | Should have |  | 3.10.2.1 | Could have |
| 3.2.1.11.6 | Could have |  | 3.2.5.4 | Could have |  | 3.10.3.1 | Should have |
| 3.2.1.11.7 | Could have |  | 3.2.5.5 | Should have |  | 3.10.3.2 | Should have |
| 3.2.1.12 | Must have |  | 3.2.5.6 | Should have |  | 3.10.4.1 | Must have |
| 3.2.2.1 | Must have |  | 3.2.5.7 | Could have |  |  |  |
| 3.2.2.2 | Should have |  | 3.2.5.8 | Could have |  | 3.11.1 | Could have |
| 3.2.2.3 | Must have |  | 3.2.5.9 | Could have |  |  |  |
| 3.2.2.4 | Should have |  | 3.2.5.10 | Could have |  | 3.12.1 | Must have |
| 3.2.2.5 | Must have |  | 3.2.5.11 | Could have |  | 3.12.2 | Could have |
| 3.2.2.6 | Could have |  | 3.2.5.1 | Could have |  |  |  |
| 3.2.2.7 | Must have |  |  |  |  | 3.17.1 | Must have |
| 3.2.2.7.1 | Could have |  | 3.2.6.1 | Won’t have |  |  |  |
| 3.2.2.7.2 | Should have |  | 3.2.6.1.1 | Won’t have |  |  |  |
| 3.2.2.7.3 | Must have |  |  |  |  |  |  |
| 3.2.2.7.4 | Could have |  |  |  |  |  |  |
| 3.2.2.8 | Could have |  |  |  |  |  |  |
| 3.2.2.9 | Could have |  |  |  |  |  |  |
| 3.2.2.10 | Must have |  |  |  |  |  |  |
| 3.2.2.11 | Should have |  |  |  |  |  |  |
| 3.2.2.12 | Should have |  |  |  |  |  |  |

# 4 Qualification Provisions

User Interface requirements will be confirmed with exploratory testing.

We can verify the data stored in the database like project identifier, programming language of a project, number of contributors, number of commits, number of pull requests, number of issues, number of releases and date of latest commit by comparing it to GitHub.com.

# 5 Requirements Traceability

The MU GitHub Analyzer consists of three major components, the user interface, the database, and the interface with GitHub. This table plots which requirements are relevant to each component.

|  |  |  |  |
| --- | --- | --- | --- |
|  | User interface | Database | GitHub |
| 3.1.1 | x |  |  |
| 3.2.1.1 |  |  | x |
| 3.2.1.2.1 | x |  | x |
| 3.2.1.2.2 | x |  | x |
| 3.2.1.2.3 | x |  | x |
| 3.2.1.2.4 | x |  | x |
| 3.2.1.2.5 | x |  | x |
| 3.2.1.2.6 | x |  | x |
| 3.2.1.2.7 | x |  | x |
| 3.2.1.2.8 | x |  | x |
| 3.2.1.3 | x |  |  |
| 3.2.1.4 | x |  | x |
| 3.2.1.5 | x |  |  |
| 3.2.1.6 | x |  |  |
| 3.2.1.7 | x |  |  |
| 3.2.1.8 | x |  |  |
| 3.2.1.9 | x |  |  |
| 3.2.1.10 | x |  |  |
| 3.2.1.11 | x |  |  |
| 3.2.1.11.1 | x |  |  |
| 3.2.1.11.2 | x |  |  |
| 3.2.1.11.3 | x |  |  |
| 3.2.1.11.4 | x |  |  |
| 3.2.1.11.5 | x |  |  |
| 3.2.1.11.6 | x |  |  |
| 3.2.1.11.7 | x |  |  |
| 3.2.1.12 | x | x |  |
| 3.2.2.1 | x | x |  |
| 3.2.2.2 | x | x |  |
| 3.2.2.3 | x | x |  |
| 3.2.2.4 | x | x |  |
| 3.2.2.5 | x | x |  |
| 3.2.2.6 | x | x |  |
| 3.2.2.7 | x |  |  |
| 3.2.2.7.1 | x |  |  |
| 3.2.2.7.2 | x |  |  |
| 3.2.2.7.3 | x |  |  |
| 3.2.2.7.4 | x |  |  |
| 3.2.2.8 |  | x | x |
| 3.2.2.9 | x | x |  |
| 3.2.2.10 |  | x | x |
| 3.2.2.11 | x |  |  |
| 3.2.2.12 | x |  |  |
| 3.2.2.13 | x | x |  |
| 3.2.3.1 |  | x |  |
| 3.2.4.1 |  | x |  |
| 3.2.4.2 |  | x |  |
| 3.2.4.3 |  | x | x |
| 3.2.4.4 |  | x | x |
| 3.2.5.1 |  |  | x |
| 3.2.5.1.1 |  |  | x |
| 3.2.5.1.2 |  |  | x |
| 3.2.5.1.3 |  |  | x |
| 3.2.5.2 |  | x |  |
| 3.2.5.2.1 |  | x |  |
| 3.2.5.2.2 |  | x |  |
| 3.2.5.2.3 |  | x |  |
| 3.2.5.2.4 |  | x |  |
| 3.2.5.2.5 |  | x |  |
| 3.2.5.2.6 |  | x |  |
| 3.2.5.2.7 |  | x |  |
| 3.2.5.2.8 |  | x |  |
| 3.2.5.2.9 |  | x |  |
| 3.2.5.2.10 |  | x |  |
| 3.2.5.2.11 |  | x |  |
| 3.2.5.2.12 |  | x |  |
| 3.2.5.3 | x |  |  |
| 3.2.5.4 |  | x | x |
| 3.2.5.5 |  | x |  |
| 3.2.5.6 |  |  | x |
| 3.2.5.7 |  |  | x |
| 3.2.5.8 |  |  | x |
| 3.2.5.9 |  |  | x |
| 3.2.5.10 | x | x |  |
| 3.2.5.11 |  |  | x |
| 3.2.6.1 | x | x |  |
| 3.2.6.1.1 | x | x |  |
| 3.3.2.1.1 |  |  | x |
| 3.3.2.1.2 |  |  | x |
| 3.3.2.1.3 |  |  | x |
| 3.3.2.1.4 |  |  | x |
| 3.3.2.1.5 |  |  | x |
| 3.3.2.1.6 |  |  | x |
| 3.3.2.1.7 |  |  | x |
| 3.3.2.2 |  |  | x |
| 3.3.2.3 |  | x |  |
| 3.3.2.4 |  |  | x |
| 3.3.3.1 |  | x |  |
| 3.3.3.2 |  | x |  |
| 3.3.3.2.1 |  | x |  |
| 3.3.3.2.2 |  | x |  |
| 3.3.3.2.3 |  | x |  |
| 3.3.3.2.4 |  | x |  |
| 3.3.3.2.5 |  | x |  |
| 3.3.3.2.6 |  | x |  |
| 3.3.3.3 |  | x |  |
| 3.9.1 |  |  |  |
| 3.10.1.1 |  |  |  |
| 3.10.2.1 |  |  |  |
| 3.10.3.1 |  | x |  |
| 3.10.3.2 |  |  |  |
| 3.10.4.1 |  |  | x |
| 3.11.1 | x |  |  |
| 3.12.1 |  |  |  |
| 3.12.2 |  |  |  |
| 3.17.1 |  |  | x |

# 

# 6 Notes

**6.1 Glossary**

|  |  |
| --- | --- |
| API | Application programming interface |
| GUI / UI | Graphical user interface |
| JDBC | Java database connectivity |
| LOC | Lines of code |
| MU | Monmouth University |
| MUGHA | MU GitHub Analyzer |
| N/A | Not applicable. |
| Pool | The list of repositories chosen for analysis. |
| RDBMS | Relational database management system. |
| TBD | To be decided. |