**Software Design Description (SDD)**

*Document Number: SDD03*

*Date: Thursday, May 01, 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. CSCI-WIDE DESIGN DECISIONS**

**4. CSCI ARCHITECTURAL DESIGN**

**4.1. CSCI Components**

**4.2. Concept of Execution**

**4.3. Interface Design**

4.3.1. Interface Identification and Diagrams

4.3.2. Repository Crawl Interface

4.3.3. Database Storage Interface

4.3.4. Local Repository Interface

4.3.5 Static Analysis Extension

**5. CSCI DETAILED DESIGN**

**5.1. MainController**

**5.2 StaticAnalysis**

**5.3 Repository**

**6. REQUIREMENTS TRACEABILITY**

**7. 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.

## 1.3 Document Overview

This document describes the design details of the MU GitHub Analyzer.

# 2 Referenced Documents

L. Innes and S. Muralidharan “Software Requirements Specification,” 2016

# 3 Software/Hardware Item-Wide Design Decisions

**3.1.** The UI shall have a search menu.

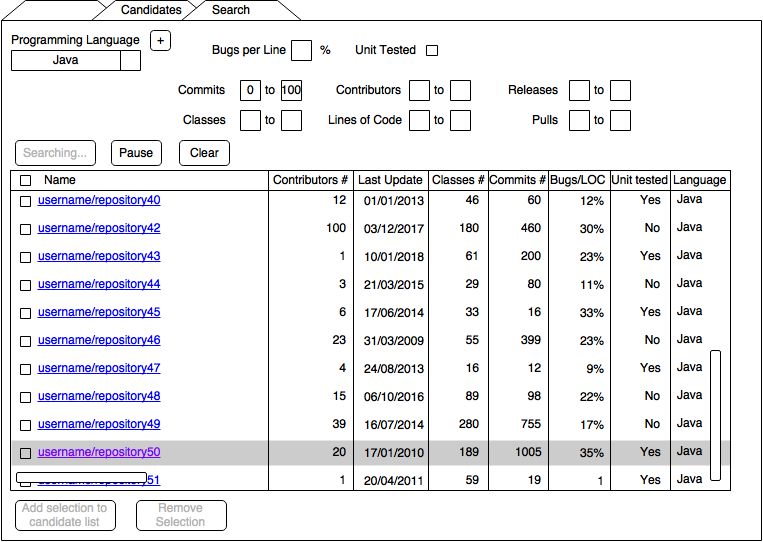
1. The menu shall have areas to input search parameters.

2. The menu shall have a Search button to initiate searching.

3. The menu shall have an Add Selection to List button.

4. The menu shall have a Remove Selection button.

The user shall be able to search from the menu and the user will be able to view the search results in the format below



**3.2**. The UI shall have a Candidate pool menu.

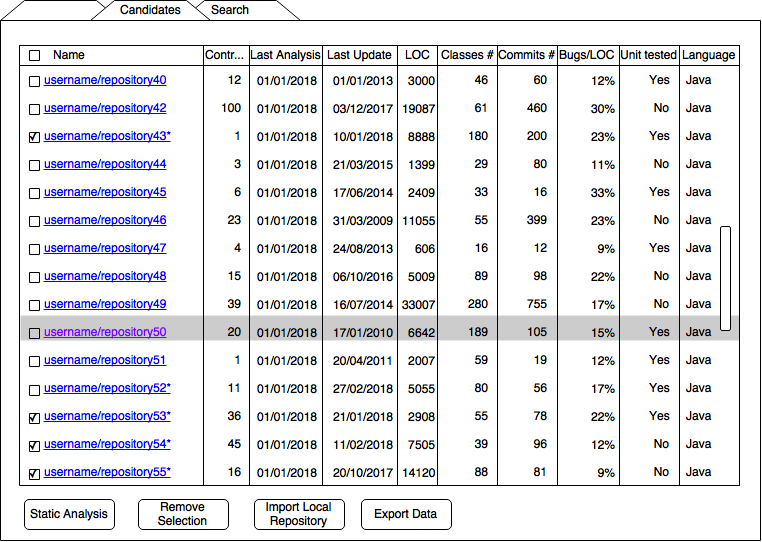
1. The menu shall have a Check For Updates button.

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.

The user shall have the option of choosing from the Candidate pool menu and when the user selects it the user shall be able to view the results in the format below

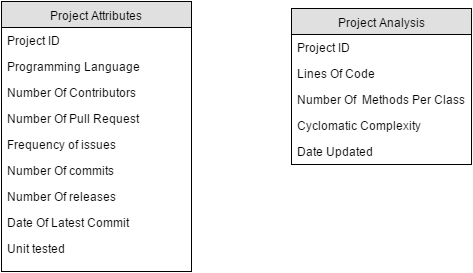


**3.3.** Users shall be able to select repositories to perform static analysis on.

**3.4.** 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 which will be stored in database.

**3.5.** User view provides project level breakdown of attributes and derived metrics.

GitHub projects and their generated metrics are captured in the “Project Analysis” table. Non-GitHub projects will have generated metrics in the “Project Analysis” table. Non-GitHub projects attributes are available in “Project Attributes” table. The relational database table schema for Project Attributes and Project Analysis would be as follows:

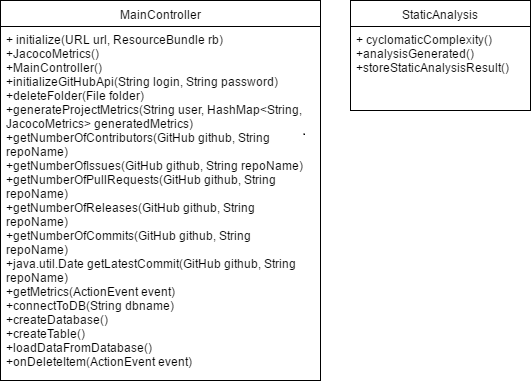


**3.6.** This application will run as a stand alone application.

**3.7.** The software design is modular and each module can be maintained independently. Coding style followed will be consistent across modules to facilitate easy maintenance. Updates to the software

# 4 Software Item Architectural Design

## 4.1 Software Item Components



There are two main components to MUGHA’s software.

MainController is going to be class for interacting with the actual GitHub API, containing methods pertaining to accessing GitHub’s information while searching for repositories.

* initializeGitHubApi() will be the method that takes user input to login to github.
* getNumberOfContributors(),getNumberOfIssues(),getNumberOfPullRequests(),

getNumberOfReleases(),getNumberOfCommits(),getLatestCommit() are all accessed thorugh github API .

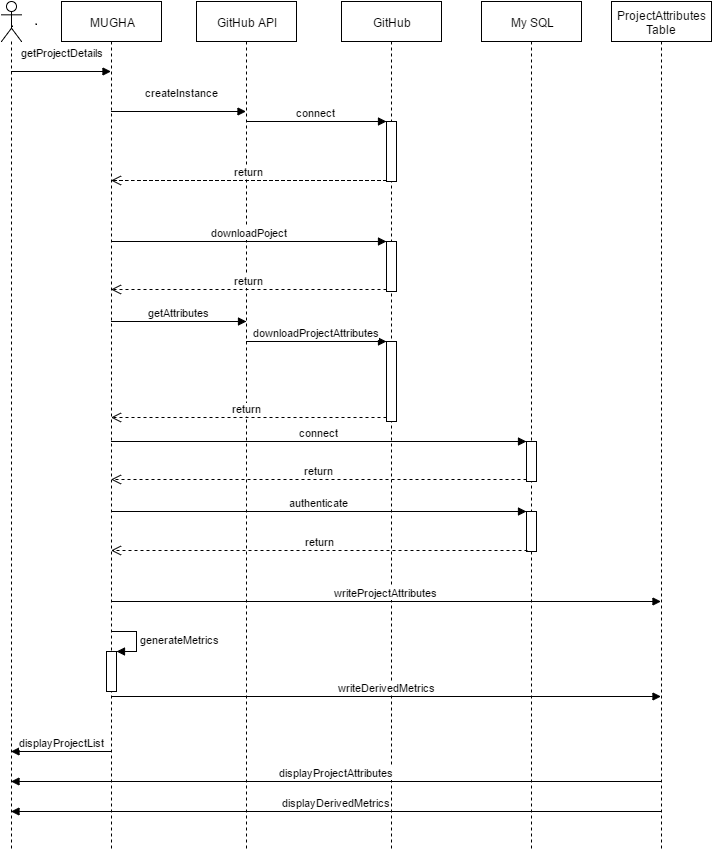
Static Analysis is going to be a class containing methods pertaining to the Static Analysis Process.

* runStaticAnalysis() initiates the analysis process on a given repository, runs each other method, and returns a report in the form of a String.
* cyclomaticComplexity() calculates the cyclomatic complexity of the project in the repository.
* analysisGenerated() captures the date the analysis of a metric was made.
* storeStaticAnalysisResult() stores the metrics found by the other static analysis methods in the repository’s database entry.

Repository is going to be an object for representing and handling project repositories and holding their attributes.

## 4.2 Concept of Execution

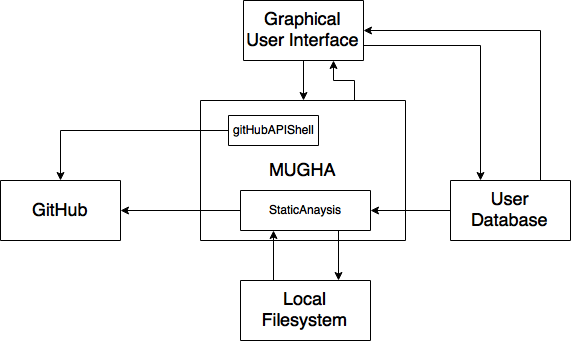
The following sequence diagram describes the sequence of operations to be performed when the user adds public project repositories from GitHub to their candidate pool.



## 4.3 Interface Design

### 4.3.1 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, the Database Storage Interface between MUGHA and the database. There is also a Graphical User Interface and an interface between MUGHA and the user’s local filesystem. All interfaces are in development but the Repository Crawl interface is built on the GitHub API and the Database Storage interface is built on the JDBC API.



### 4.3.2 Repository Crawl Interface

Interfacing Entities:

* MU GitHub Analyzer
  + MainController class
  + staticAnalysis class
* GitHub API
* GitHub

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. It is facilitated by the GitHub API and an internet connection. Through the API, MUGHA can retrieve project attributes and the files from the projects. All API access is over HTTPS, and accessed from the https://api.github.com. All data is sent and received as JSON.

The staticAnalysis class shall receive the following information regarding project attributes through this interface:

* The organization of files in every iteration of the main branch, processed as a data tree of objects

The MainContrller class shall receive the following information regarding project attributes through this interface:

* A unique identifier (likely the URL extension of its page on GitHub.com), a String
* The number of contributors, an integer.
* The number of pull requests, an integer.
* The number of commits, an integer.
* Date of latest commit, a string.

### 4.3.3 Database Storage Interface

Interfacing Entities:

* MU GitHub Analyzer
  + User Interface
* JDBC API
* Database

The Database storage Interface is the interface of the MU GitHub Analyzer and the user’s database that occurs during storage of repositories. It is facilitated by the JDBC API. It is meant for storage and retrieval of the following information:

* The organization of files in every iteration of the main branch, processed as a data tree of objects.
* A unique identifier (likely the URL extension of its page on GitHub.com), a String
* The number of contributors, an integer.
* The number of pull requests, an integer.
* The number of commits, an integer.
* Date of latest commit, a string.

### 4.3.4 Local Repository Interface

Interfacing Entities:

* staticAnalysis class
* User’s file system

MUGHA’s static code analysis tool only gets used on repositories if they have been imported to the user’s machine. When given a folder path to analyze, it will generate stats, of data types that can be stored in the database, based on the commit history of the repository.

### 4.3.5 Static Analysis Extension Interface

Interfacing Entities:

* staticAnalysis class
* FindBugs

FindBugs is an example of one possible open-source analysis tool that will be utilized by the broad Static Analysis function. The static analysis tool will relay the source code (possibly String) to FindBugs and FindBugs will produce a report (integer).

# 5 Software Item Detailed Design

## 5.1 MainController

## Methods:

* initializeGitHubApi()takes input from the UI to carry through the GitHub Search to find matching repositories.
* getNumberOfContributors(GitHub github, String repoName) Gets the total number of contributors for a particular project.
* getNumberOfIssues(GitHub github, String repoName) Gets the total number of issues which includes open and closed issue for a particular project.
* getNumberOfPullRequests(GitHub github, String repoName) Gets number of pull requests from github API.
* getNumberOfReleases(GitHub github, String repoName) Gets number of release using GitHub API.
* getNumberOfCommits(GitHub github, String repoName) Gets number of commits for a particular project.
* getLatestCommit(GitHub github, String repoName) Gets the latest commit made for that particular project.
* compareUnitTestedStatus(boolean unitTested) : boolean
  + Takes whether the User only wants repositories with unit testing and determines whether the found repository matches. Accessed by searchGitHub()

## 5.2 StaticAnalysis

Methods:

* findLinesOfCode() : int
  + Counts functional lines of code for the project in the repository.
* cyclomaticComplexity() : int
  + Calculates the cyclomatic complexity of the project in the repository.
* runStaticAnalysis(Repository) : String
  + Initiates the analysis process on a given repository, runs each other method, and returns a report in the form of a String.
* storeStaticAnalysisResult() : boolean
  + Stores the metrics found by the other static analysis methods in the repository’s database entry.
* numberOfmethodsPerClass() : int
  + Returns the dividend of the number of methods and the number of classes in the latest version of the project.
* analysisGenerated() : Date
  + Gets the date the analysis of a metric was made.

## 5.3 Repository

Attributes:

* ProjectURL:String
  + The key that is used to access the project again.
* ProgrammingLanguage:String
  + The programming language of the project.
  + Should be an array for projects that use multiple languages but this version of MUGHA only supports analyzing Java.
* Contributors:int
  + The number of contributors to a repository.
* PullRequests:int
  + The number of pull requests that have been made to the repository.
* Issues:int
  + The percentage of issues in the current project version.
* Commits:int
  + The number of commits made to the repository.
* Releases:int
  + The number of releases of the repository’s project.
* LastUpdate:Date
  + The date of the last update to the repository.
* UnitTested:boolean
  + Whether the repository contains Unit test classes.
* LinesofCode:int
  + The number of lines of code of the latest version of the project.
* Methods:int
  + The amount of methods in the latest version of the project.
* CyclomaticComplexity:int
  + The cyclomatic complexity number of the project.
* FromGithub:boolean
  + Whether the repository is from GitHub or added locally.

# 6 Requirements Traceability

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | User interface | StaticAnalysis | GitHubAPIShell | Repository |
| 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 |  |  | x |
| 3.2.1.6 | x |  |  |  |
| 3.2.1.7 | x |  |  |  |
| 3.2.1.8 | x |  |  |  |
| 3.2.1.9 | x |  |  | x |
| 3.2.1.10 | x |  |  | 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 |  | x | x |
| 3.2.2.7.2 | x |  |  | x |
| 3.2.2.7.3 | x | x |  | x |
| 3.2.2.7.4 | x |  |  |  |
| 3.2.2.8 |  |  | x | x |
| 3.2.2.9 | x |  | x | x |
| 3.2.2.10 |  | x |  | x |
| 3.2.2.11 | x |  |  |  |
| 3.2.2.12 | x |  |  |  |
| 3.2.2.13 | x | x |  | x |
| 3.2.3.1 |  |  | x | 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 |  |  |
| 3.2.5.5 |  | x |  |  |
| 3.2.5.6 |  | x |  | x |
| 3.2.5.7 |  |  |  | x |
| 3.2.5.8 |  | x |  |  |
| 3.2.5.9 | x |  |  | x |
| 3.2.5.10 |  | x |  |  |
| 3.2.5.11 |  | x |  | x |
| 3.2.5.12 | x | x |  |  |
| 3.2.6.1 |  |  |  |  |
| 3.2.6.1.1 |  |  |  |  |
| 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 | x |  |
| 3.11.1 | x |  |  |  |
| 3.12.1 |  |  |  |  |
| 3.12.2 |  |  |  |  |
| 3.17.1 |  |  |  |  |

# 

# 7 Notes

**7.1 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. |