****

**SOEN 6611**

**SOFTWARE MEASUREMENT**

**GROUP R**

|  |  |  |
| --- | --- | --- |
| Name | Student ID | Email |
| Jaswanth Banavathu | 40080737 | jaswanthbanavathu@gmail.com |
| Manushree Mallaraju | 40082236 | manushreemallaraju@gmail.com |
| Arvind Korchibettu Adiga | 40105178 | adiga1993@gmail.com |
| Anusha Keralapura Thandavamurthy | 40102962 | kt.anusha21@gmail.com |
| Kranthi Kumar Sankuru | 40102793 | kranthikumar9922@gmail.com |

**Index**

1. INDEX

2. Selected metrics and correlation analysis.

2.1. [Selected Metrics](#Metrics)

2.2. [Correlation between the metrics](#Correlation)

[3. Related work](#Relatedwork)

[4. Selected projects](#Projects)

[5. Resource planning](#RP)

[6. References](#Ref)

**2.1 Selected Metrics and Correlation Analysis**

**Metric 1 & 2: Statement and Branch coverage**

**Statement Coverage:**

Statement coverage is a white box testing technique. It is a metric, which is used to calculate and measure the number of statements in the source code which have been executed. Using this technique, we can check what the source code is expected to do, its quality and the flow of different paths in the program. It can test only true conditions, but not false conditions.

**Statement coverage** = No of statements Executed/Total no of statements in the source code \* 100

**Branch Coverage:**

A branch is the outcome of a decision, so branch coverage simply measures which decision outcomes have been tested. Branch coverage is also known as **Decision coverage** or **all-edges coverage**. Unlike statement coverage, this metric can calculate both true and false conditions.

**Decision Coverage**= (Number of decision outcomes executed/Total number of decision outcomes) \*100%

**Metric 3: Test Suite Effectiveness**

Software testing is widely used in the industry for quality assurance as tests can tackle software bugs early in the development process and serve for regression purposes. Test suite effectiveness is measured by the number of faulty versions of a System Under Test (SUT) that are detected by a test suite. Since mutant detection correlates with real fault detection, we can use Mutation testing tools to generate faulty versions of the program and then run the tests to determine if the fault was detected.

**Metric 4: Code Complexity**

Software complexity refers to the complexity of the code developed and is based on various factors such as the number of looping or conditional statements, code complexity and more.

Examples: Halstead and McCabe Complexity.

Control Flow /Cyclomatic Complexity (McCabe Metric):

McCabe’s metrics are based on a control flow representation of the program. A program graph is used to depict control flow. Nodes represent processing tasks (one or more code statements). Edges represent control flow between nodes.

According to the McCabe, the cyclomatic complexity M is the measure of linearly independent paths for a program is. Which is:

M = E - N + 2P. where,

E-Number of edges in the graph

N-Number of Nodes in the graph

P-Number of connected components.

**Metric 5: Software maintenance Metrics**

**Fix Backlog and Backlog Management Index**

Fix backlog is a workload statement for software maintenance. It is related to both the rate of defect arrivals and the rate at which fixes for reported problems become available. It is a simple count of reported problems that remain at the end of each month or each week. Using it in the format of a trend chart, this metric can provide meaningful information for managing the maintenance process. Another metric to manage the backlog of open, unresolved, problems is the backlog management index (BMI).

http://www.informit.com/content/images/chap4_0201729156/elementLinks/04eq04.gif

As a ratio of number of closed, or solved, problems to number of problem arrivals during the month, if BMI is larger than 100, it means the backlog is reduced. If BMI is less than 100, then the backlog increased.

**Metric 6: Software Quality Metrics**

**Post-release defect density**

Defect Density is the number of defects confirmed in software/module during a specific period of operation or development divided by the size of the software/module. It enables one to decide if a piece of software is ready to be released.

Defect Density = Defect count/size of the release OR defect count/lines of code

**2.2 Correlation between metrics:**

* Correlation between each coverage metric (Metric 1&2) and test suite

effectiveness (Metric 3). The rationale is that test suites with higher coverage

might show better test suite effectiveness.

* Correlation between one complexity metric (Metric 4) and each coverage metric

(Metric 1&2). The rationale is that classes with higher complexity are less likely to

have high coverage test suites.

* Correlation between each coverage metric (Metric 1&2) and Metric 6.

Classes with low each coverage contain more bugs.

* Correlation between Metric 5 and Metric 6: Backlog is directly proportional to the defect density.

**3. Related Work**

Several factors need to be considered for the calculation of metrics, such as releases, bug count, test case availability, complexity etc. Metrics 5 and 6 require information such as number of bugs resolved every month, the pending backlog etc.

**4. Selected Projects**

**#1 Name: Apache Sling**

**Version:** 2.10.0

**SLOC:** 3.5 million LOC

**Purpose:** Apache Sling is a framework for RESTful web applications based on extensible content tree. It maps the HTTP request URLs to content resources based on the request’s path, extension and selectors.

**Programming Language:** Java

**Requirements to run:** Java Runtime Environment (JRE) version 1.8 or higher.

**#2 Name: JFreeChart**

**Version:** 1.5.0

**SLOC:** 317,178 Lines of Code

**Purpose:** JFreeChart is a free java chart library that makes it easy for developers to display professional quality charts in their applications. It supports for many output types, including Swing and JavaFX components, image files and vector graphics file formats. JFreeChart also works with [GNU Class path](https://en.wikipedia.org/wiki/GNU_Classpath), a [free software](https://en.wikipedia.org/wiki/Free_software) implementation of the standard [class library](https://en.wikipedia.org/wiki/Java_Class_Library) for the [Java programming language](https://en.wikipedia.org/wiki/Java_(programming_language)).  
**Programming Language:** Java

**Requirements to run:** JFreeChart requires JDK 1.6 or later.

**#3 Name: Apache Commons Lang**

**Version:** 3.9

**SLOC:** 80.7k LOC

**Purpose:** This project provides the methods required for manipulation of its core classes which the standard Java libraries fail to provide. The methods include basic numerical methods, object reflection and String manipulation methods.

**Programming Language:** Java

**Requirements to run:** JDK 1.8.

**#4 Name:** **Apache Commons Collections**

**Version: 4.**4.2

**SLOC:** 132k LOC

**Purpose**: It became a standard for collection handling in Java because it has many powerful data structures that enhance the development of many significant Java applications.

**Programming Language:** Java

**Requirements to run:** Java Development kit

**#5** **Name: Apache Commons Math**  
**SLOC:** 186k LOC   
**Purpose:** This project is a library of lightweight, self-contained mathematics and statistics components addressing the most common practical problems not immediately available in the Java programming language.  
**Programming language:** Java  
**Requirements**: JDK 1.6 or later.

**Tools:**

* For issue tracking, we have decided to select the JIRA issue tracker, which is helpful in obtaining graphical representation of reports.

<https://issues.apache.org/jira/projects>

* For obtaining the correlation, we will be using the Pearson and Spearman correlation methods.

<https://support.minitab.com/en-us/minitab-express/1/help-and-how-to/modeling-statistics/regression/supporting-topics/basics/a-comparison-of-the-pearson-and-spearman-correlation-methods/>

* To calculate complexity and volume metrics, we will be using scitools.

<https://scitools.com/feature/metrics/>

**5. Resource Planning**

All members are assigned one project, each having at least 100k lines of code. Each member is responsible to calculate metrics 1-6 for their respective projects. This way each member can understand and learn how to calculate metrics for any given application effectively.

**6. References:**

<https://www.guru99.com/code-coverage.html>

<http://tryqa.com/what-is-decision-coverage-its-advantages-and-disadvantages/>

<http://ceur-ws.org/Vol-2070/paper-03.pdf>

<https://www.academia.edu/5563523/Software_Complexity_Metrics_A_Survey>

<http://www.informit.com/articles/article.aspx?p=30306&seqNum=3>

<https://www.guru99.com/defect-density-software-testing-terminology.html>