1. Team Information

Ali Zafar Iqbal 40076898 - alizafar9361@gmail.com

Micheal Hanna 40075977 **-** michael_baligh@yahoo.com

Han 40020549 - rebeccahangao@gmail.com

Arjun singh Thakur 40076630 **-** arjun.thakur565@gmail.com

Mahshad Saghaleini 40058409 - Mahshad.saghaleini@gmail.com

2. Selected Metrics and Correlation analysis

Metric 1 & 2: Statement coverage and branch coverage

Bench coverage : Shows how many branches from each decision point is executed at least once thereby the more coverage percent it shows, the more opportunity to find the existing bug.

Statement coverage : Shows how many statements are executed at least once during the test and thereby the more coverage percent it shows, the more opportunity to find the existing bug.

Metric 3: Mutation Score: The goal of Mutation Testing is to assess the quality of the test cases which should be robust enough to fail mutant code

Mutation Score = (Killed Mutants / Total number of Mutants) * 100

Test cases are mutation adequate if the score is 100%.

Metric 4: McCabe metric (Cyclomatic complexity)

McCabe measures the number of linearly independent paths that can be executed through a piece of source code.

E = the number of edges in CFG

N = the number of nodes in CFG

P = the number of connected components in CFG

D = is the number of control predicate (or decision) statements

For a single method or function, P is equal to 1

Cyclomatic Complexity = E – N + 2P

Or Cyclomatic Complexity = D + 1

Metric 5: Maintainability index

Maintainability Index is a software metric which measures how maintainable (easy to support and change) the source code is.

The maintainability index is calculated as a factored formula consisting of **Lines of Code**, **Cyclomatic Complexity** and **Halstead volume**.

The calculation method as follows:

First, we need to measure the following metrics from the source code:

V = Halstead Volume

G = Cyclomatic Complexity

LOC = count of source Lines of Code (SLOC)

CM = percent of lines of Comment (optional)

From these measurements the MI can be calculated:

The original formula:

MI = 171 - 5.2 * ln(V) - 0.23 * (G) - 16.2 * ln (LOC)

The derivative used by SEI is calculated as follows:

MI = 171 - 5.2 * log2(V) - 0.23 * G - 16.2 * log2 (LOC) + 50 * sin (sqrt (2.4 * CM))

In all derivatives of the formula, the most major factor in MI is Lines of Code, which effectiveness has been subjected to debate.

Metric 6: Defect Density

Defect Density is the number of confirmed defects detected in software/component during a defined period of development/operation divided by the size of the software/component.

Defect Density = number of defects / size

3. Correlation Analysis

1 Correlation between **Metric 1** and **Metric 2**: The rationale is that test case with higher statement covers mostly lead to higher. .

3 Correlation between **Metric 1 & 2** and **Metric 4**: The classes with higher complexity are less likely to have high coverage test suites.

4 Correlation between **Metric 1 & 2** and **Metric 6**: illustrates that the classes with low test coverage often contain more bugs.

5 Correlation between **Metric 5** and **Metric 6**: According to understand the meaning of Maintainability index (MI) and Backlog Management Index (BMI). It can be found that if the value of MI is high while the value of BMI should be high. The high value of BMI means the bugs of the software are easier to handle, and it also means that the software is easier to maintain. Thus, followed by high-valued MI.

4. Tools Utilized

Metrics 1 & 2: It can be collected by using tools like JaCoCo, JCov and CodeCover, and most projects that we choose contains the developers' test case, so it should be easy to collect and analyze.

Metric 3: The mutation testing is a time consuming if it is done manually. Stryker or PIT Testing are automation tools to speed up the process.

Metrics 4: McCabe is based on control flow graph, so we need to collect data like SLOC, the number of control predicate from the projects that we chose. To collect that data, we can use tools such as MCCABE IQ.

Metrics 5: To calculate the maintainability index, we need to collect data from other metrics such as metrics 1&2&4, because while calculating this index, line of codes, Halstead Volume, Cyclomatic Complexity and percent of lines of Comment are needed.

Metrics 6: Different systems of issue tracking systems can be used for example: JIRA, GitHub,..etc

5. Selected Open-Source Systems

1:Apache bookkeeper

Bookkeeper is a large project (more than 100k SLOC) offers a reliable replicated log service. We plan to use from 4.0 to 4.9, 10 versions for evolution-related metrics. It is a maven project and contains **JaCoCo plugin** which will be easier for us to test the coverage (metrics 1&2) of the existing developer's test suit. What's more, there is a bug tracking system on the website of this project, so it will be used for metrics 5. Above all, It is overall a good project for us to study and practice analysis of an open source object and get an in-depth look in software measurement.

Source code: https://github.com/apache/bookkeeper
Issue Tracker: https://github.com/apache/bookkeeper/issues

2-Project Name: RxJava

Source code: https://github.com/ReactiveX/RxJava
Issue Tracker: https://github.com/ReactiveX/RxJava/issues
RxJava is a Java VM implementation of Reactive Extensions: a library for composing asynchronous and event-based programs by using observable sequences.

3-Project Name: guava

Guava is an open-source set of common libraries for Java, mainly developed by Google engineers.Guava can be roughly divided into three components: basic utilities to reduce menial laborers to implement common methods and behaviors, an extension to the Java collections framework (JCF) formerly called the Google Collections Library, and other utilities which provide convenient and productive features such as functional programming, graphs, caching, range objects, and hashing.

Source code: https://github.com/google/guava
Issue Tracker: https://github.com/google/guava/issues
LOC: 656883

4-Project Name: Apache Commons

Apache Commons Lang, a package of Java utility classes for the classes that are in java.lang's hierarchy, or are considered to be so standard as to justify existence in java.lang.
Source Code:https://github.com/apache/commons-lang
Issue Tracker:https://issues.apache.org/jira/projects/LANG/issues/LANG-1462?filter=allopenissues

LOC:125547

6. Resource Planning

Name	Tasks
Ali Zafar Iqbal	Studying about coverage testing tools; Statistical data analysis. Documentation; Combine the result of metrics 1 & 2 & 4 to do the metrics 5;
Micheal Hanna	Build up the source code; Coding the report script; Gitmaster
Mahshad Saghaleini	Studying about coverage testing tools; Collecting data and report by using Jacoco for matrics 1&2 Documentation;
Arjun singh Thakur	Collect data for metrics 3 ; Studying statistics and exploring statistical tools;
Han	Studying about coverage testing tools; Collect data for metrics 2 ; Documentation;

7. References

[1] G. K. Gill and C. F. Kemerer, "Cyclomatic complexity density and software maintenance productivity," IEEE Trans. Software Eng., vol. 17, (12), pp. 1284-1288, 1991. Available: http://dx.doi.org/10.1109/32.106988.DOI:10.1109/32.106988.

[2] M. M. Suleman Sarwar, S. Shahzad and I. Ahmad, "Cyclomatic complexity: The nesting problem," in 8th International Conference on Digital Information Management, ICDIM 2013, September 10, 2013 - September 12, 2013, Available: http://dx.doi.org/10.1109/ICDIM.2013.6693981. DOI: 10.1109/ICDIM.2013.6693981.

[3] Y. H. Yang, "Software quality management and ISO 9000 implementation," Industrial Management + Data Systems, vol. 101, (7), pp. 329-38, 2001. Available: http://dx.doi.org/10.1108/EUM0000000005821. DOI: 10.1108/EUM0000000005821.

[4] Luca Cardelli, Serge Abiteboul.2015. Software Quality Management, p75-77. https://www.tutorialspoint.com/software_quality_management/index.htm

[5] M. H. Moghadam and S. M. Babamir, "Mutation score evaluation in terms of object oriented metrics," in 4th International Conference on Computer and Knowledge Engineering, ICCKE 2014, October 29, 2014 - October 30, 2014, Available: http://dx.doi.org/10.1109/ICCKE.2014.6993419. DOI: 10.1109/ICCKE.2014.6993419.

[6] Mutation Testing in Software Testing

https://www.guru99.com/mutation-testing.html#1

[7] Defect Density

http://softwaretestingfundamentals.com/defect-density/