Project proposal

1. Team Information

Team O:

Tian Wang	40079289	wangtian5329@gmail.com
Minxue Sun	40084491	minxue.sun@gmail.com
Tiancheng Xu	40079681	tianchengxu1121@gmail.com
Qing Li	40082701	liqing51269@gmail.com

2. Selected Metrics and Correlation analysis

Metric 1&2: Statement Coverage and Branch Coverage

Statement Coverage technique can check what the source code is expected to do and what it should not. It can also check the quality of the code and the flow of different paths in the program. The main drawback of this technique is that we cannot test the false condition in it.

Statement Coverage Formula:

$$Statement\ Coverage = \frac{Number\ of\ executed\ statements}{Total\ number\ of\ statements} \times 100\%$$

Branch coverage aims to ensure that each one of the possible branches from each decision point is executed at least once and thereby ensuring that all reachable code is executed.

Branch Coverage Formula:

$$Branch\ Coverage = \frac{Number\ of\ executed\ branches}{Total\ number\ of\ branches} \times 100\%$$

Metric 3: Mutation testing

Mutation testing has traditionally been used to evaluate the effectiveness of test suites. Mutation testing involves the creation of many versions of a program each with a single syntactic fault. A test suite is evaluated against these program versions (i.e. mutants) in order to determine the percentage of mutants a test suite is able to identify (i.e. mutation score).

Mutation Score Formula:

$$Mutation\ Score = \frac{Killed\ Mutants}{Total\ number\ of\ Mutants} \times 100\%$$

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

Metric 4: McCabe(Cyclomatic Complexity)

McCabe is one of metrics that measure structure complexity of a program.

Formula 1:

$$Cyclomatic\ Complexity\ =\ E - N + 2P$$

where E = the number of edges in CFG, N = the number of nodes in CFG, P is the number of connected components in CFG

Formula 2:

$$Cyclomatic\ Complexity\ =\ D + 1$$

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

Metric 5: LOC difference (DLOC)

The number of lines of codes changed edited, added or deleted (DLOC) were the most effective for predicting adaptive maintenance effort.

DLOC Formula:

$$DLOC\ =\ The\ number\ of\ lines\ of\ codes\ changed\ edited\ or\ added\ or\ deleted$$

Metric 6: Backlog Management Index (BMI)

Backlog Management Index (BMI) is an important metric to manage the backlog of open, unresolved problems in a Software Project.

BMI Formula:

$$BMI = \frac{Number\ of\ problems\ closed\ during\ the\ month}{Number\ of\ problems\ arrivals\ during\ the\ month} \times 100\%$$

3. Related Work

Metric 1&2:

Statement coverage is a white box test design technique involving the execution of all the executable statements in the source of code at least once. It is used to calculate and measure the number of statements in the source code which can be executed given requirements. [1]

Branch coverage, every outcome from a code module is tested. For example, if the outcomes are binary, we need to test both True and False outcomes. By using the Branch

coverage method, we can measure the fraction of independent code segments. It also helps us to find out which sections of code don't have any branches. [1]

Metric 3:

Effectiveness of testing is often measured by the quality of a test suite, i.e., the ability of a test suite to uncover faults in a program. A test suite that reveals more bugs is considered of higher quality than the one which reveals fewer bugs. The best test suite can detect every bug; whenever a bug is introduced, at least one test case will fail.

The mutation score is the result of mutation testing. Mutation testing is a fault-based testing technique where modifications of the program under test lead to faulty versions of the program under test. These faulty versions are called mutants. A mutant is said to be a killed mutant if a test suite can distinguish the mutant from the original program. [3]

Metric 4:

We measure the number of linearly independent paths that can be executed through a piece of source code. The number of test cases should be equal to the number of independent paths.

McCabe introduced a complexity metric by the name of cyclomatic complexity (CC), which measured the maximum number of linearly independent paths through a control flow graph. Apart from being used as a complexity metric, it was also an indicator of the testability and maintainability of a program. [4]

Metric 5:

The maintenance effort for a software application depends on measurable metrics that can be derived from the software development process. Previous research [5] displays the results of correlation analysis: the higher the value of the coefficient, the stronger the relationship between effort and the metric. The results suggest that percentage of operators changed and the number of lines of codes changed edited, added or deleted (DLOC) were the most effective for predicting adaptive maintenance effort.

Table 1. Correlation between metrics and effort.

	Metrics	Coefficient of Determination (R^2)	Significance level from ANOVA Regression
1	%Operators Changed	0.978	0.0002
2	LOC Delta -DLOC	0.779	0.00003
3	% Mod change/add	0.192	0.117
4	Noprtr	0.152	0.444
5	CF	0.108	0.525
6	CR	0.071	0.358
7	Hdiff	0.046	0.683
8	LCOM	0.034	0.725
9	AC	0.033	0.518
10	CC	0.032	0.581
11	TCR	0.011	0.741
12	PM	0.008	0.77
13	MP	0.006	0.79
14	Classes Changed	0.006	0.8
15	MI	0.003	0.874
16	HPVol	0.0008	0.929
17	Classes Added	0.0006	0.946
18	Heff	0.0004	0.969
19	LOC	0.00001	0.991

Metric 6:

Backlog Management Index (BMI) is an important metric to manage the backlog of open, unresolved problems in a Software Project. It is typically used extensively in maintenance and Support projects (or phases) to ensure the project remains under control. It is related to both the rate of defect arrivals and the rate with which the fixes to the defects become available. A Backlog of workload is defined as the count of problems that remain unresolved after a particular time span (week/month).

If the number of Problems/Service Desk Incidents closed in the month is more than the number of arrivals, the BMI will be more than 100 percent. This will indicate that the Software system and the project for maintaining and supporting it are under control and moving towards more stability.

If on the other hand the number of Problems/Service Desk Incidents closed in the month is less than the number of arrivals, the BMI will be less than 100 percent. This will indicate that the Software System and (or) project maintaining and supporting it is not stable. In the long term if this condition continues, an intervention from Project Manager, Sponsors, Technical Resources and other Stakeholders may be necessary.[2]

4. Selected Open-Source Systems

(1) Apache log4j (SLOC 191K)
https://github.com/apache/log4j
We use the version in 2.13.0. The project satisfies the requirement of SLOC over 100K. There are existing test cases for calculating metrics 1&2&3, The source code of this project can be used for calculating metric 4. The project has previously released versions to measure the DLOC(metric 5) as well as the BMI(metric 6).

(2) Spotify docker maven plugin (SLOC 6.39K)
https://github.com/spotify/docker-maven-plugin
We use the version in 1.2.2 to measure all the metrics. There are existing test cases for calculating metrics 1&2&3, The source code of this project can be used for calculating metric 4. The project has previously released versions to measure the DLOC(metric 5) as well as the BMI(metric 6).

(3) Openfire（SLOC 172K）
https://github.com/igniterealtime/Openfire
We use the version in 4.5.1 to measure all the metrics. The project satisfies the requirement of SLOC over 100K. There are existing test cases for calculating metrics 1&2&3, The source code of this project can be used for calculating metric 4. The project has previously released versions to measure the DLOC(metric 5) as well as the BMI(metric 6).

5. Resource Planning

Task	Assignment
1. Data Collection:	
Metric 1&2	Two members collect data for metric1&2 (three projects)
Metric 3&4	Two members collect data for metric 3&4 (three projects)
Metric 5&6	Four members collect data for metric5&6 (three projects)
2. Statistical analysis:	
Correlation between Metric 1&2 and Metric 3	One member
Correlation between Metric 1&2 and Metric 4	One member

Correlation between Metric 1&2 and Metric 6	One member
Correlation between Metric 5 and Metric 6	One member
3. Report writing	Four members
4. Presentation preparation	Four members

References

[1] Code Coverage Tutorial: Branch, Statement, Decision, FSM [Online]. Available: https://www.guru99.com/code-coverage.html#6

[2] Kan, Stephen H. "Software quality metrics overview." *Metrics and Models in Software Quality Engineering* (2002): 85-120.

[3]H. Felbinger, F. Wotawa and M. Nica, "Mutation Score, Coverage, Model Inference: Quality Assessment for T-Way Combinatorial Test-Suites," *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Tokyo, 2017, pp. 171-180.

[4]D. I. De Silva, N. Kodagoda and H. Perera, "Applicability of three complexity metrics," *International Conference on Advances in ICT for Emerging Regions (ICTer2012)*, Colombo, 2012, pp. 82-88.

[5] J. H. Hayes, S. C. Patel and L. Zhao, "A metrics-based software maintenance effort model," *Eighth European Conference on Software Maintenance and Reengineering*, 2004. CSMR 2004. Proceedings., Tampere, Finland, 2004, pp. 254-258.