SOEN 6611: Software Measurement

Project proposal

Team - H

S.No	Name	Roll No.	Email ID
1	Haifeng Wu	40102956	hanfordwu@gmail.com
2	Sourabh Rajeev Badagandi	40098471	sourabh.rajeev@gmail.com
3	Sandeep Kaur Ranote	40089795	ranote9466@gmail.com
4	Meet Patel	40071820	91meetpatel@gmail.com
5	Basant Gera	40082433	basantgera29@gmail.com

Submitted to Dr. Jinqiu Yang

Selected Metrics and Correlation analysis

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

Branch coverage is 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 is 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.

Tools: JaCoCo

Metric 3: ***Mutation score***

Mutation score is used to evaluate the effectiveness of test suites in terms of its ability to detect faults.It is widely used to design new software tests.[Wiki] The idea of it is to identify the mutant and kill it, so we have to compare the original code with the mutated code, if the output of tests are similar, the mutant remains alive, otherwise the mutant is killed.

Tools: PITClipse

Metric 4: ***Cyclomatic complexity (McCabe metric)***

McCabe measures the number of linearly independent paths that can be executed through a piece of source code. It is used as an indicator of program modularization, revising specifications, and test coverage. It also can be used in software quality prediction. For calculation, we draw the CFG of the source code:

E: The number of edges in CFG;

N: The number of nodes in CFG;

P: The number of connected components in CFG(For a single method, P = 1);

D: The number of control predicate/decision statements;

Cyclomatic Complexity= E - N + 2P

OR**Cyclomatic Complexity**= D + 1

Tools: JaCoCo

Metric 5 :*Maintainability, Extensibility, Reusability*

Instability:

The instability (I) metric, is defined as the ratio of efferent coupling (Ce) to total coupling[9]:

I = Ce/(Ca + Ce)

It has the range [0, 1]. I=1 indicates a maximally unstable package. I=0 indicates a maximally stable package.

The efferent coupling (Ce) of a package is the number of dependencies of that package on other packages. This is defined as the number of classes inside a given package that depend upon classes in other packages. Efferent coupling (Ce) of a package shows the degree to which other packages may necessitate changes in the given package, and thus, it is an indicator of the lack of independence of the package. In contrast, afferent coupling (Ca) can be defined as the number of classes outside a package which depend on the classes inside this package. Afferent coupling (Ca) of a package is an indicator of the packages responsibility.

Tools: Jdepend

Metric 6: *Post Release Defect Density*

A defect density is generally the number of defects confirmed / module during a specific period of development divided by the size of the software.

Defect Density = Defect count/size of the release

Tools: CLOC, JIRA website

Related Work

Metric 1 & 2

J. M. Bieman, D. Dreilinger and Lijun Lin, "Using fault injection to increase software test coverage," Proceedings of ISSRE '96: 7th International Symposium on Software Reliability Engineering, White Plains, NY, USA, 1996, pp. 166-174.The above paper describes the best approach for test coverage for branch & statement coverage.

Metric 3

S. Nica, "On the Improvement of the Mutation Score Using Distinguishing Test Cases," 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation, Berlin, 2011, pp.

423-426.By reading this paper we can think about how mutation score affects the test suite if the code is complex.

Metric 4

T. J. McCabe, "A Complexity Measure," in IEEE Transactions on Software Engineering, vol. SE-2, no. 4,

pp. 308-320, Dec. 1976.We came to know about complexity and how linearly independent path affects a project in terms of complexity which is measurable.

Metric 5

Almugrin introduced R. C. Martin's seminal work *Agile Software Development Principles, Patterns, and Practice.* Martin invented the instability and abstractness metrics to be used in the analysis of dependencies between packages. Martin defined eight metrics calculate different features of a package. These metrics are: efferent coupling (Ce), afferent coupling (Ca), Instability (I), Number of Abstract Classes (Na), Number of Classes (Nc), Abstractness (A), the distance from the main sequence (D), and the normalized distance from the main sequence (Dn). Among these metrics Martin uses efferent coupling (Ce), afferent coupling (Ca) to calculate **Instability.** Many research studies in the literature have acknowledged and applied Martin's metrics.

Metric 6

N. Nagappan and T. Ball, "Static analysis tools as early indicators of pre-release defect density," *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, Saint Louis, MO, USA, 2005, pp. 580-586

Open-Source Systems Selected

Project No : 1 Apache Commons Collections

Project Description: Apache common collections package generally consists of Java Collection framework classes. This included features like Bag interface for collection, BidiMap Interface, Mapitertor & many Comparator information.

Current version: Commons-Collection-collections-4.4*(Stable Version)*

Programming Language : Java

Requirements to Run: Java Runtime Environment (JRE) version 1.8 or higher and JDK 1.8 or higher.

SLOC (Source Line of Code): 63,365 (SLOC for java files) & 67,517 (SLOC for total files in a project). ***Version 4.4***

Project No : 2 Alibaba Fastjson

Project Description: Fastjson is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object. Fastjson can work with arbitrary Java objects including pre-existing objects that you do not have source-code of.

Current version: Alibaba FastJson (1.2.57) (*Stable Version)*

Programming Language: Java

Requirements to Run: Java Runtime Environment (JRE) version 1.8 or higher and JDK 1.8 or higher.

SLOC (Source Line of Code): 163,561(SLOC for java files) & 184,831 (SLOC for total files in a project). ***[Requirement satisfied above 100k SLOC]***

Project No : 3 Apache Common Lang

Project Description 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. It provides various utilities for numerical methods, concurrency & serialization.

Current version: Commons-Lang-3.9 *(Stable Version)*

Programming Language: Java

Requirements to Run: Java Runtime Environment (JRE) version 1.8 or higher and JDK 1.8 or higher and maven 3.0.5 or later.

SLOC (Source Line of Code): 75,643 (SLOC for java files) & 80,793 (SLOC for total files in a project).

Project No : 4 Apache Common Configuration

Project Description: It is a tool to assist in the reading of configuration / preferences files in various formats.

Current version: Apache Common Configuration 1.x and 2.x

Programming Language: Java

Requirements to Run: Java Runtime Environment (JRE) version 1.8 or higher and JDK 1.8 or higher.

And can be used to include in eclipse too.

SLOC (Source Line of Code): 108,531. ***[Requirement satisfied above 100k SLOC]***

Metrics Applied on above 4 Open source projects and their respective needs:

S. No	Metric Name	Need for Calculate the following Metrics
Metric 1	Statement coverage	To check whether no. of **statements** in the source code are getting executed if the SLOC is above 100K or less than that. And to check whether any unused statements are written in the code if any.
Metric 2	Branch coverage	**Branch coverage**to be done to know whether the code written in each branch is executed at least once if the SLOC is above 100K or less than that. And to check whether if any condition any branch is not covered.
Metric 3	Mutation Score	To check the measure of quality of the **test suite**based on faults which were detected in the software. And to check whether tests written in the test suite are effective as per the code written in the software. An adequate score can be up to 100%.
Metric 4	McCabe Complexity	We have chosen **McCabe complexity**to check the complexity of the program written by developers. **Cyclomatic complexity score**:4 [Good score]. **Cyclomatic complexity score**:5-7 [Medium score]. **Cyclomatic complexity score**:8-10 [High Complexity]. And above that is extreme complexity.

Metric 5	Maintainability, Extensibility, Reusability	1. We have chosen **maintainability** because it is an important measure since 75 % of the project cost is related to that. 2. We choose **Resusality** to check what things were reused in a program which will help us to know the difference between reuse Similarity between copied code.
Metric 6	Post Release Defect Density	We have chosen this metrics so to determine how many defects are there in the version post release and how can we relate it with Package instability. We wanted to study how come faults affect the versions.

Distribution of work among team members

Name/Metric	Metric 1 [Statement Coverage]	Metric 2 [Branch Coverage]	Metric 3 [Mutation Score]	Metric 4 [McCabe Complexity]	Metric 5 [Relative Code Churn]	Metric 6 [Defect Density]
Haifeng Wu	Project 1	**Discussion**	**Discussion**	Project 3	Project 4	Project 1
Sourabh Rajeev Badagandi	Project 2	Project 1	Project 4	**Discussion**	Project 2	Project 3
Sandeep Kaur Ranote	Project 3	Project 2	Project 1	Project 4	Project 3	Project 2
Meet Patel	Project 4	Project 3	Project 2	Project 1	**Discussion**	Project 4
Basant Gera	**Discussion**	Project 4	Project 3	Project 2	Project 1	**Discussion**

Resource Planning

Week (Number)	Week Planned	Work Done by team or Scheduled
Week 1	Feb 16	Study and do R & D on tools which will help us to calculate all the metrics. If done manually will show how it is done.
Week 2	Feb 23	Start performing Statement & branch Coverage on project 1,2,3,4
Week 3	March 1	Start performing mutation testing & McCabe Complexity on project 1,2,3,4.
Week 4	March 8	Start performing metric 5 (relative code churn) after studying on the project 1,2,3,4.
Week 5	March 15	Start performing metric 6 (Defect Density) after studying on the project 1,2,3,4
Week 6	March 22	Compare & Evaluate the result for all 4 milestones in Report
Week 7	April 1	Compare & Evaluate the result for all 4 milestones in Report
Week 8	April 10	Delivery Date

References

1. https://github.com/apache/commons-configuration
2. https://lucene.apache.org/solr/
3. https://lucene.apache.org/
4. https://camel.apache.org/
5. https://camel.apache.org/manual/latest/getting-started.html
6. https://maven.apache.org/
7. https://en.wikipedia.org/wiki/Apache_Maven
8. S. Almugrin, W. Albattah, O. Alaql, M. Alzahrani and A. Melton, "Instability and Abstractness Metrics Based on Responsibility," *2014 IEEE 38th Annual Computer Software and Applications Conference*, Vasteras, 2014, pp. 364-373.
9. Bansiya, Jagdish, Letha H. Etzkorn, Carl G. Davis and Wei Li. “A Class Cohesion Metric For Object-Oriented Designs.” *JOOP*11 (1999): 47-52.

10.