Department of Computer and Software Engineering

SOEN 6611-E (Software Measurement)

Winter 2019

Milestone 1: Project Proposal

"We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality"

Submitted by (Team 10) :

- Dhaval Chandreshkumar Modi **(40083289)**
- Hari Narayanan Kannan **(40047827)**
- Harsh Divecha **(40084737)**
- Ramit Basra **(40045043)**
- Venkata Lakshmi Sai Krishna Chadalavada **(40087977)**

Submitted to:

Professor:

Dr. Rodrigo Morales Alvarado

1. Team Formation:

No.	Name	Roles/Responsibilities
1.	Hari Narayanan Kannan	Documentation, Research, Implementation of tools
2.	Venkata Lakshmi Sai Krishna Chadalavada	Documentation, Research
3.	Dhaval Chandreshkumar Modi	Research, Implementation of tools
4.	Ramit Basra	Documentation, Research
5.	Harsh Divecha	Coding, Research, Implementation of tools

2. Type of Study:

it is not clear to me the type of study. CK metrics does not measure fault-proneness. Additionally, there is overlapping between CK metrics and characterization of code smells.

In this study we will be using CK metrics to analyze bad smell ution of the software mentioned in this proposal. Source for mining repository has been selected as GitHub adhering to all the criteria mentioned in requirements, as it is a very well known and an extensively used revision control system.

3. Related Studies:

- CK metrics will be studied in correlation to faulty classes by detecting bad smells in the code retrieved as binary output (0 for no bad smell and 1 if detected) and finally using Bayesian inference for different releases of the selected open source software. [1]
- The fault-proneness of object-oriented systems has been studied by implementing a hybrid model of 12 metrics. [5]
- A metrics-based approach for detecting design flaws along with two concrete techniques for the detection of two well-known design flaws has been done. Based on the proposed approach, further detection techniques for other common design-flaws can be defined. [7]

I think is easy to analyse fault proneness directly into the bug repository, than using metrics as a proxy, isn't it?

4. Projects:

Projects selected are all based on Java, as most of the open source software's are available in this language and codes written in java require Object Oriented concepts to be implemented implicitly and it is one of the most extensively used and documented languages known. All the projects selected have more than 300k lines of code which increases the probability of the code being fault prone or consisting of defects in general.

Project Selected:	**Tools and plug-ins [6]:**
• Glassfish • Ant • Mockito • Wildfly	• Understand 5.0.9 by Scitools • JDeodorant (Eclipse) • Metrics 2 (Eclipse) • Analyst4J

what about the other projects, mockito and fly?

Revisions Selected for studying the most desirable projects:

Mockito	**Ant**
• v2.24.0 (Current Version) • v2.22.0 • v2.19.0 • v2.18.0 • v2.15.0 • v2.13.0 • v2.10.0	• 1.10.5 (Current Version) • 1.10.3 • 1.10.2 • 1.10.1 • 1.9.7 • 1.9.4 • 19.3

5. Metrics:

The C&K Metric Suite has been implemented to analyse fault proneness in the projects under consideration [3] [4]. Chidamber and Kemerer (CK) et al. [2] gives the formal definition of the metrics as follows:

(i)Weighted Methods per Class (WMC): This measures the sum of complexity of the methods in a class. The complexity of the class may be calculated by the cyclomatic complexity of the methods. The high value of WMC indicates that the class is more complex as compare to the low values.

(ii)Depth of Inheritance Tree (DIT): DIT metric is used to find the length of the maximum path from the root node to the end node of the tree. DIT represents the complexity and the behavior of a class, and the complexity of design of a class and potential reuse.

(iii)Number of children (NOC): According to Chidamber and Kemerer, the Number of Children (NOC) metric may be defined for the immediate sub class coordinated by the class in the form of class hierarchy. These points are come out as NOC is used to measure that "How many subclasses are going to inherit the methods of the parent class". The greater the number of children, the greater the potential for reuse. The greater the number of children, the greater the likelihood of improper abstraction of the parent class.

(iv)Coupling between Objects (CBO): CBO is used to count the number of the class to which the specific class is coupled. The rich coupling decreases the modularity of the class making it less attractive for reusing the class and more high coupled class is more sensitive to change in other part of the design through which the maintenance is so much difficult in the coupling of classes.

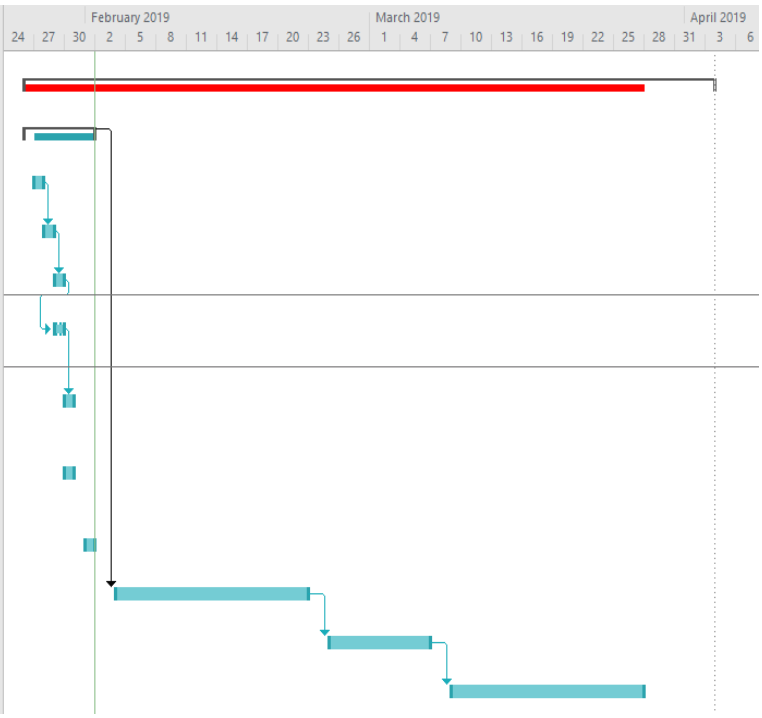
(v)Response for class (RFC): The response set of a class (RFC) is defined as set of methods that can be executed in response and messages received a message by the object of that class. Larger value also complicated the testing and debugging of the object through which, it requires the tester to have more knowledge of the functionality.

(vi)Lack of Cohesion in Methods (LCOM): This metric is used to count the number of disjoints methods pairs minus the number of similar method pairs used. Since cohesiveness within a class increases encapsulation it is desirable and due to lack of cohesion may imply that the class is split in to more than two or more sub classes. Low cohesion in methods increase the complexity, when it increases the error proneness during the development is so increasing.

you mentioned code smells, so you should define which code smells you want to study as well!

6. Resource Planning:

Task Name	Duration	Start	Finish	Team Member Worked
Resource Allocation	**50 days**	**Sat 1/26/19**	**Wed 4/3/19**	
Milestone 1 (Project Proposal)	**7 days**	**Sat 1/26/19**	**Fri 2/1/19**	
Identifying Metric Suite	1 day	Sun 1/27/19	Sun 1/27/19	Hari Narayanan Kannan, Harsh Divecha
Decision between Type 1 and Type 2	1 day	Mon 1/28/19	Mon 1/28/19	Hari Narayanan Kannan, Harsh Divecha
Research on Related Work	1 day	Tue 1/29/19	Tue 1/29/19	Hari Narayanan Kannan, Harsh Divecha
Research on Tools	1 day	Tue 1/29/19	Tue 1/29/19	Hari Narayanan Kannan, Harsh Divecha, Dhaval Chandreshkumar Modi
Experimenting on Tools	1 day	Wed 1/30/19	Wed 1/30/19	Hari Narayanan Kannan, Harsh Divecha, Dhaval Chandreshkumar Modi
Research on Open Source Repository	1 day	Wed 1/30/19	Wed 1/30/19	Dhaval Modi, Sai Krishna Chadalavada, Ramit Basra
Documentation	1 day	Fri 2/1/19	Fri 2/1/19	Hari Narayanan Kannan, Harsh Divecha, Ramit Basra
Milestone 2 (Survey of Related Work)	**15 days**	**Mon 2/4/19**	**Fri 2/22/19**	
Milestone 3 (Data Collection)	**8 days**	**Mon 2/25/19**	**Wed 3/6/19**	
Milestone 4 (Evolution Analysis)	**14 days**	**Sat 3/9/19**	**Wed 3/27/19**	



7. References:

[1] Amit Sharma, Sanjay Kumar Dubey, "Comparison of Software Quality Metrics for Object- Oriented System", IJCSMS International Journal of Computer Science & Management Studies, Special Issue of Vol. 12, June 2012 ISSN (Online): 2231 –5268.

[2] C. Shyam, Kemerer, F. Chris, "A Metrics Suite for Object- Oriented Design" M.I.T. Sloan School of Management, pp. 53-315, 1993.

[3] Istehad Chowdhury, Mohammad Zulkernine, "Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities", Journal of Systems Architecture 57 (2011) 294–313.

[4] Ramanath Subramanyam and M.S. Krishnan, "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects", IEEE Transactions on Software Engineering, Vol. 29, No. 4, April 2003.

[5] E Fioravanti, P. Nesi, "A Study on Fault-Proneness Detection of Object-Oriented Systems", Proceedings Fifth European Conference on Software Maintenance and Reengineering, 07 August 2002, 6905819.

[6] Rüdiger Lincke, Jonas Lundberg and Welf Löwe, "Comparing Software Metrics Tools", Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2008, Seattle, WA, USA, July 20-24, 2008.2.

[7] Radu Marinescu, “Measurement and Quality in Object-Oriented Design,” PhD thesis, Politechnica University of Timisoara, Romania, 2002.