Available online at http://acfa.apeejay.edu

APEEJAY JOURNAL OF COMPUTER SCIENCE AND APPLICATIONS

ISSN:0974-5742(P)
Vol. 1, January 2013

Evaluating the Effect of Code Cloning on Software Maintenance Cost

Gagandeep Singh[1], Neha Kohli[2]
Assistant Professor, Department of Computer Science & Engineering, Sai Institute of Engineering & Technology, Amritsar.
M.Tech, Department of Computer Science & Engineering & Technology, SIET, Amritsar.

ABSTRACT

The duplication of code is common practice to make software development faster, to enable "experimental" development without impacting the original code, or to enable independent evolution. Since these practices involve both duplication and modification, they are collectively called code cloning and the duplicated code is called a code clone. The effects of code cloning on the quality of source code are not well understood. In much of the literature on the topic, code cloning is considered detrimental to the quality of the source code, as it is generally believed that code clones can cause additional maintenance costs. In this dissertation comparative study is done between the cloned and cloned free code using the various software metrics. The cloned and cloned free code is taken in java. Then evaluation of both the codes is done using metric1.3.6 and eclipse as the simulator. Results are then tabulated in the table which shows that software maintenance cost is increased to a large extent when cloned code is used for the software development.

1. INTRODUCTION

Effective management of any process requires quantification, measurement, and modeling. Software metrics provide a quantitative basis for the development and validation of models of the software development process. Metrics can be used to improve software productivity and quality. This module introduces the most commonly used software metrics used for software development process.
It is important to further define the term *software metrics* as used in this module. Essentially, *software metrics* deals with the measurement of the software product and the process by which it is developed. For now, this discussion, the software product should be viewed as an abstract object that evolves from an from an initial statement of need to a finished software system, including source and object code and the various forms of documentation produced during development.

2. TYPES OF SOFTWARE METRICS:

Process Metrics:
Process metrics are known as management metrics and used to measure the properties of the process which is used to obtain the software. Process metrics include the cost metrics, efforts metrics, advancement metrics and reuse metrics. Process metrics help in predicting the size of final system & determining whether a project on running according to the schedule.

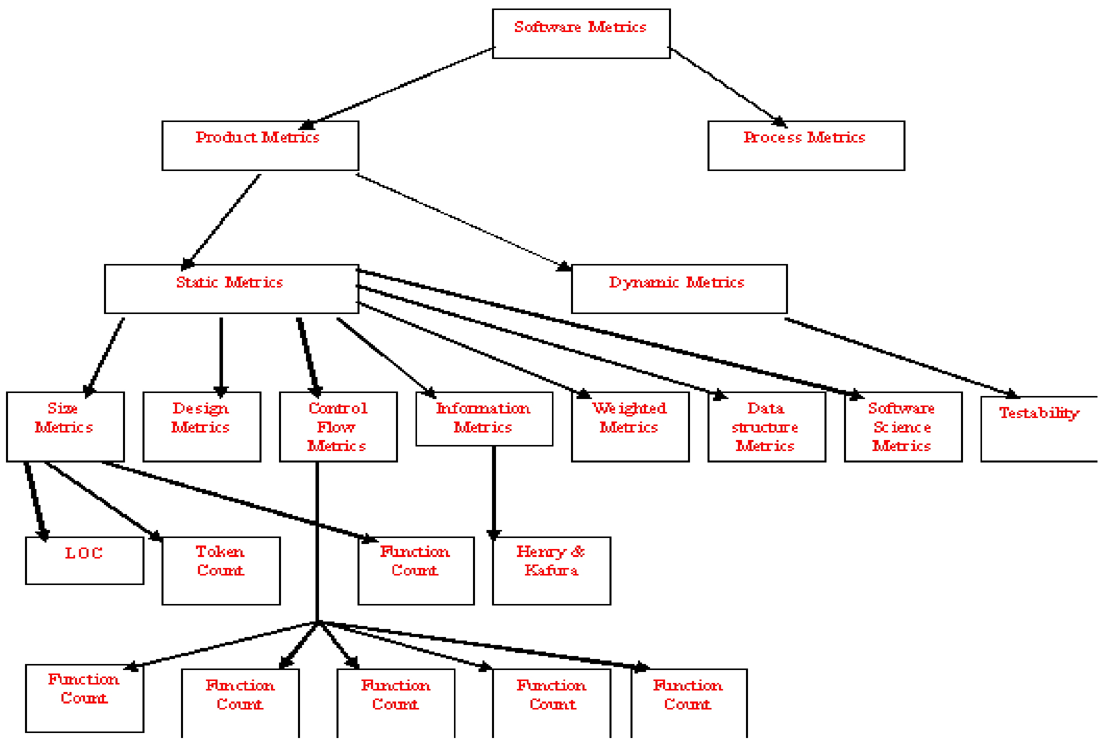
Products Metrics:
Product metrics are also known as quality metrics and is used to measure the properties of the software. Product metrics includes product non reliability metrics, functionality metrics, performance metrics, usability metrics, cost metrics, size metrics, complexity metrics and style metrics. Products metrics help in improving the quality of different system component & comparisons between existing systems.

3. SIZE METRICS:

Line of Code:
It is one of the earliest and simpler metrics for calculating the size of computer program. It is generally used in calculating and comparing the productivity of programmers.

• Productivity is measured as LOC/man-month.
• Any line of program text excluding comment or blank line, regardless of the number of statements or parts of statements on the line, is considered a Line of Code.

Classification of Software Metrics



Function Count:
• The size of a large software product can be estimated in better way through a larger unit called module. A module can be defined as segment of code which may be compiled independently.
• For example, let a software product require n modules. It is generally agreed that the size of module should be about 50-60 line of code. Therefore size estimate of this Software product
is about n x 60 line of code .

4. CONTROL FLOW METRICS:

McCabe's Cyclomatic Metric:McCabe interprets a computer program as a set of strongly connected directed graph. Nodes represent parts of the source code having no branches and arcs represent possible control flow transfers during program execution.
McCabe proposed the cyclomatic number, V(G) of graph theory as an indicator of software complexity. The cyclomatic number is equal to the number of linearly independent paths through a program in its graphs representation. For a program control graph G, cyclomatic number, V(G), is given as:
$V(G) = E - N + P$
E = The number of edges in graphs G
N = The number of nodes in graphs G
P = The number of connected components in graph G.

5. OBJECT ORIENTED METRICS:

Weight Method per Class (WMC)
Lack of Cohesion of Methods (LCOM)
Coupling between Object Classes (CBO)
Depth of Inheritance Tree (DIT)
Number of Children (NOC)

Weight Method per Class (WMC):

This metric is used to measure the understandability, reusability and maintainability.
The WMC is a count of the methods implemented within a class or the sum of the complexities of the methods. But the second measurement is more difficult to implement because not all methods are accessible within the class hierarchy because of inheritance.

Lack of Cohesion of Methods (LCOM):
Cohesion is the degree to which methods within a class are related to one another and work together to provide well bounded behavior.
• LCOM uses variable or attributes to measure the degree of similarity between methods.
• We can measure the cohesion for each data field in a class; calculate the percentage of methods that use the data field.

Coupling between Object Classes (CBO):
• Coupling is a measure of strength of association established by a connection from one entity to another.
• CBO is a count of the number of other classes to which a class is coupled. It is measured by counting the number of distinct non inheritance related class hierarchy on which a class depends.

Depth of Inheritance Tree (DIT):
• Inheritance is a type of relationship among classes that enables programmers to reuse previously defined object objects, including variables & operators.
• Inheritance decrease the complexity by reducing the number of operations and operators, but this abstraction of objects can make maintenance and design more difficult.
• The deeper a class within the hierarchy, the greater the number of methods and is likely to inherit, making it more complex to predict its behavior.

6. RESEARCH DESIGN

In this section different version of an open source software has been taken whose program code is written in java. The code is written in both ways, 1. Cloned free code 2. Cloned code. Evaluation of these two codes is done using metric Analyst 4J. It is a tool used in eclipse. It consists of large number of source code metrics which are used to compare the results obtained from both the codes. The results achieved are summarized under the various tables as shown later.

ANALYST 4j
Analyst4j automates software measurement and empowers the users with an unique search and analysis environment based on metrics. Software metrics deals with the measurement of software product and the process by which it is developed. Metrics provide a quantitative basis for the development and validation of models of the software development process. Software metrics expose various quality attributes of code, which are of great use for software development and maintenance team. Analyst4j ensures effective use of metrics by providing a search facility based on automated software metrics.

ECLIPSE 3.11
The Eclipse Platform is designed for building integrated development environments (IDEs). It can be used to create diverse end-to-end computing solutions for multiple execution environments. The Eclipse Platform is the foundation for constructing and running integrated end-to-end software development tools. The platform consists of open source software components that tool vendors use to construct solutions that plug in to integrated software workbenches. The Eclipse Platform incorporates technology expressed through a well-defined design and implementation framework.
In table 1. All the listed metrics of the metric analyst 4j compares the values for the cloned and cloned free sytem A. The value of all the metrics varies from the original code and hence it defines how the maintenance cost of a software increased when a code is cloned.
Eclipse act as the simulator for the all the metrics present in the tool and calculate the all the results for each of them. We can notice in the above table that all the values are different from each other but hold a great existence as well. All the metrics provide the code with distinct values and features.

CATEGORY	METRICS	SYSTEM A	CLONED SYSTEM A
Size	No of java files	73	73
	Java lines of code	3770	4202
Maintainability index (mi)	Loc, # of comments, cyclomatic Complexity, halstead's volume	117.92	109.10
Structural measures (sm)	Coupling	8.59	7.83
	Cohesion	0.5	0.4
	Size of classes	8.98	9.96
	Depth of inheritance tree	1.4	1.3

Table 1. **Maintenance metrics to System A**

In table 2. All the listed metrics of the metric analyst 4j compares the values for the cloned and cloned free system B. The value of all the metrics varies from the original code and hence it defines how the maintenance cost of software decreases when a code is cloned free in nature.
All the values are the original values as the code is cloned free. All the metrics have different values. These values are calculated by eclipse which acts as the simulator for this.

CATEGORY	METRICS	SYSTEM B	CLONED SYSTEM B
Size	No of java files	86	86
	Java lines of code	4323	5889
Maintainability index (mi)	Loc, # of comments, cyclomatic Complexity, halstead's volume	112.3	103.14
Structural measures (sm)	Coupling	8.0	9.3
	Cohesion	0.6	0.5
	Size of classes	9.36	10.51
	Depth of inheritance tree	1.6	1.7

Table 2. **Maintenance Metrics applied to the System B**

METRICS THAT EFFECT SOFTWARE MAINTAINENCE COST THE MOST

In this section we will discuss the metrics used that determine the raised maintenance cost of the software product. The values of metrics are different tor both codes and hence these values are used to calculate that the cloned code results in higher maintenance cost of the software product. The below table 3 denotes comparison of both cloned codes of system A and system B The basic metrics that effect maintenance of the cloned software product are-:

- ✓ Size of project
- ✓ Maintainability Index
- ✓ Structural Measures

1. SIZE OF PROJECT

Maintenance of a software product increase to a large extent when size of the product increase. The size in turn relates to the lines of code, number of files analyzed within the product. When size increases, maintainability also increases consequently. In the case of cloning, size is increased as lines of code are increase. Hence maintenance cost also increases.

2. MAINTAINBILTY INDEX

The Maintainability Index (MI) is used for assessing the maintainability of complete systems. The three-metric MI uses a polynomial to combine the average per module of three traditional code measures (lines of code, cyclomatic complexity and Halstead Volume) into a single-value indicator of maintainability.
To our knowledge, there are no heuristics for MI classification values for object-oriented systems. However, because classes are smaller in such systems than modules in conventional systems,
Researchers have argued that the MI for object-oriented systems should be higher . In cloned systems the value of MI decreases due to increase in size of the code which affect the maintenance of the software. Thus maintaince cost increases due to this.

3. STRUCTURAL MEASURES

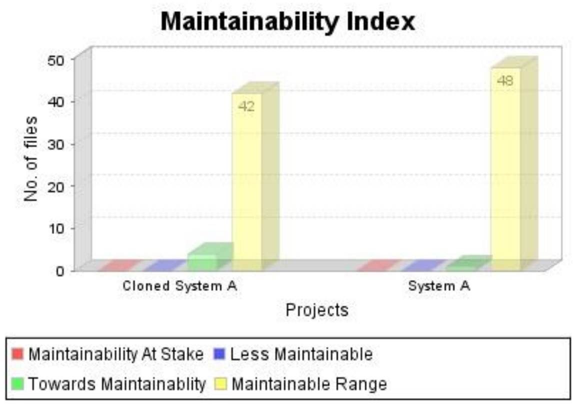
The most common set of metrics for assessing code maintainability is structural measures. The subset includes the coupling measure , the
cohesion measure and the measure of size of classes , number of methods per class and depth of inheritance tree (DIT). By combining these metrics we can conclude that s whether the system is maintainable or not. If the system is less maintainable, it requires high maintainence cost.

CATEGORY	METRICS	CLONED SYSTEM A	CLONED SYSTEM B
Size	No of java files	73	86
	Java lines of code	4202	5889
Maintainability index (mi)	Loc, # of comments, cyclomatic Complexity, halstead's volume	109.10	103.14
Structural measures (sm)	Coupling	7.83	9.3
	Cohesion	0.4	0.5
	Size of classes	9.96	10.51
	Depth of inheritance tree	1.3	1.7

Table 3 . Maintenance metrics of the Two cloned systems A and B

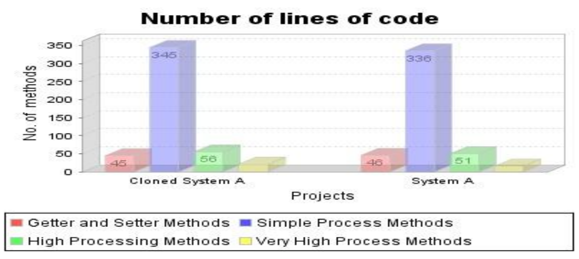
7. GRAPHICAL VIEW

GRAPH 1.



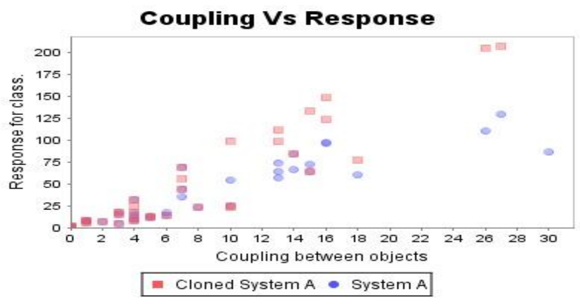
In the above graph 1 the maintainability index metric is plotted between cloned free and cloned system A. No of files are plotted on Y- axis and both the system are plotted on X-axis. It can be clearly viewed that cloned free system has high value of maintainability index, thus system is less maintainable and hence maintenance cost will also be less. But the value of MI has been decreased for cloned code which shows that system needs to be maintained and a high cost will be incurred to maintain the system according to the needs of the software engineers.

GRAPH 2.



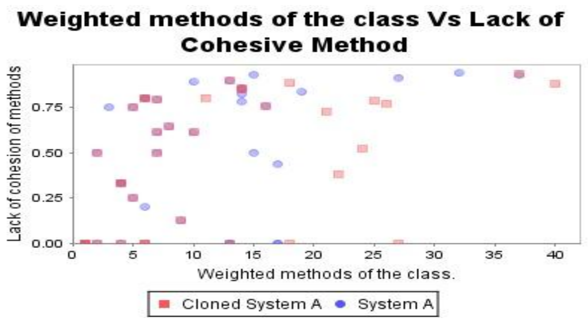
In graph 2 LOC metric is considerd and compared of both the codes. Number of methods are incresed in cloned code due to which length of the code is increaesd and in turn complexity of the code is increased too. This is a simple factor that will show that when complexity and length of the code is increased maintaince cost is increased automatiacllyy. Blue colour in the graph defines the increased number of methods.

GRAPH 3.



In graph 3 we have compared the coupling and response of class metrics together for Cloned system A and system A. A good software code should have low coupling and high cohesion for low maintainability of software product. Coupling is placed on X axis and response for class is placed on Y-axis. Red colour indicates high coupling which means that cloned code needs to be maintained spending high maintainance cost.

GRAPH 4



Cohesiveness of methods within a class is desirable, since it promotes encapsulation. In above graph weighted methods are placed on X axis and cohesion methods are placed on Y axis. Hence the deviation in values clearly shows the variation between cloned and cloned free codes of system A. cohesion should be high for a less maintainable software code.

8. CONCLUSION

In this paper various software metrics are used and their values have been evaluated and compared with two different codes and hence analysis have been performed to determine the metrics that are responsible for the increased maintenance cost of a software product.

Dependency graphs and applet viewers of both the codes defined the major differences and comparison values.

Dependency graphs and applet viewers of both the codes defined the major differences and comparison values.

Our observation is that at the entire system level, the simplest metric of size was the best predictor of maintainability. The choice of metrics, rather than actual

Maintainability, may determine the outcome of a study. Consequently, this indicates that overall system size and maintainability index as a measure of maintainability has been underrated in the software engineering community. However, the other maintenance metrics are overrated. Researchers in software engineering should be cautious when using such metrics as for actual maintainability unless the metrics have been properly evaluated in the same context for which they serve.

In this dissertation comparative study is done between the cloned and cloned free code using the various software metrics. The cloned and cloned free code is taken in java. Then evaluation of both the codes is done using metric1.3.6 and eclipse as the simulator. Results are then tabulated in the table which shows that software maintenance cost is increased to a large extent when cloned code is used for the software development.

9. FUTURE WORK

We plan to investigate the following regarding clones.

1. Relationship of clone instability with unintentional inconsistent changes and bug propagation:

We have already observed that clones exhibit higher instability and higher maintenance cost as compared to non-cloned code. We would like to investigate how clone instability is related with unintentional inconsistent changes and hidden bug propagation. We also plan to investigate which type(s) of clones is (are) mainly responsible for unintentional inconsistent changes and bug propagation.

2. **Detection of refactorable fragments**:

As we have observed, for some clone fragments refactoring is even impossible. So, identification of refactorable clones can be a promising research topic. We would like to explore this topic.

3. Investigation on the co-changeability of program artifacts:

We investigated the impact of clones on method co-changeability and found that clones can possibly increase method co-changeability. We would like to further investigate the effects of clones on the co-changeability of other program artifacts such as classes, and packages.

4. **Development of an effort calculation model:**

There are many existing effort calculation models but these cannot be used to calculate the efforts required for cloned and non-cloned code separately. For this reason, we plan to develop an effort calculation model which will facilitate the calculation of efforts for cloned and non-cloned code separately. We have already started working on it and had a considerable amount of progress.

References

[1]. Paulo Meirelles, Carlos Santos Jr., Joao Miranda, Fabio Kon, Antonio Terceiro and Christina Chavez (2010), “A Study of the Relationships between Source Code

[2]. A Study of Software Metrics, IJCEM International Journal of Computational Engineering & Management, Vol. 11, January 2011

[3]. “The Role of Object Oriented metrics” from archive.eiffel.com/ doc/ manuals/ technology.

[4]. Lines of Code, http://en.wikipedia.org/wiki/Source_lines_of_code

[5]. Eclipse Metrics, http://metrics.sourceforge.net/

[6]. Questioning Software Maintenance Metrics:

[7]. A Comparative Case Study ESEM’12, September 19–20, 2012, Lund, Sweden.

[8]. Chanchal K. Roy, James R. Cordy and Rainer Koschke (2009), “Comparison and Evaluation of Code Clone Detection Techniques and Tools: A Qualitative Approach”, Science of Computer Programming February 24, 2009

[9]. Research on Maintainability Evaluation of Service-Oriented Software. Ben Korge

[10]. Anda. Assessing Software System Maintainability using Structural Measures and Expert Assessments, Proc. 23rd Int’l Conf. on Software Maintenance, pp. 204-213, 2007.

[11]. Assessing The Influence Of Software Cloning On Software Maintenance, International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 4, April - 2013 ISSN: 2278-0181