Role of Code Clone Detection, Analysis and Removal in Enhancing Software Quality

Sarveshwar Bharti[1], Hardeep Singh[2]
[1, 2] Guru Nanak Dev University, Amritsar, Punjab, India
(E-mail: sarveshwar.dcsrsh@gndu.ac.in, hardeep.dcse@gndu.ac.in)

Abstract—Improving Software Quality with minimized development as well as maintenance cost is the main focus of Software Engineering. But there are number of definite causes that decrease the software quality and increases the maintenance cost. Duplicate code fragments, popularly known as Code Clones, have emerged as one of the main causes of Software Quality depreciation, due to increased code size as well as complexity. This paper answers the basic questions that need to be studied to discuss the role of Code Clone Detection, Analysis and Removal in enhancing Software Quality and thus attempts to make a single canvas of whole research on code clones found in literature.

Keywords—Software Quality; Structural Quality; Code Clones; Code Bloating; Refactoring

I. Introduction

The main aim of Software Engineering is to improve the Software Quality and minimize the software development cost. In spite of using various engineering principles in the design, development and maintenance of software, literature lists different definite causes of software quality depreciation and increased maintenance cost, and Code Clones are one of them. Even though effect of code clones on software quality is not very much clear, but most of the literature points to harmfulness of code clones. These code clones are nothing but a copy of the original code snippet, with or without modifications.

This paper attempts to provide a brief overview of Code Cloning literature and its relation to the Software Quality. In this paper authors discusses how Software Quality is enhanced using the concept of code clones. These duplicated code fragments present in the source code needs to be detected first, so that they can be analysed and removed. Presently there are number of clone detection tools available that can be used for this purpose.

To study the role of Code Clone Detection, Analysis and Removal in enhancing Software Quality, this paper answers some basic questions listed below in the form of Research Questions (RQ):

RQ1: How are Code Clones related with Software Quality?

RQ2: What is the effect of Code Clones on Software Quality?

RQ3: How are Code Clones Detected, Analysed and Removed?

RQ4: How Software Quality is enhanced using Code Clone Detection, Analysis and Removal?

To answer these questions, this paper first discusses the notion of Software Quality and Code Clones, in Section II (A) and Section II (B) respectively. Then, Section III (A) discusses the relation of Code Clones with Software Quality and Section III (B) discusses various consequences of Code Cones. Section III (C) presents an overview of whole literature on Code Clones, with discussion on Clone Detection in Section III (C(a)), Analysis in Section III (C(b)) and Removal in Section III (C(c)). And then, Section III (D) discusses how clone detection and removal is helpful in improving the Software Quality and thus provides a way to enhance the Software Quality. Finally this paper is concluded in Section IV, along with acknowledgements, and then references are given in support of what is mentioned in this paper.

II. Background

To answer the Research Questions mentioned in the Section-I, there is a need to understand the concept of Code Clones and Software Quality. Next two sub sections presents the concept of Software Quality and Code Clones, as found in literature.

A. Software Quality

Assessing quality means measuring a value of its various parameters. The term “Software Quality” seems to be simple to define, but is very complicated. The best way to understand this concept is to divide it into three aspects: functional quality, structural quality and process quality [1] as shown in figure 1. The figure 1 also shows the three groups of people who cares about the value of Quality.

Functional quality is related to the software performance, process quality is related to the software development process and structural quality is related to code structure. Various Structural Quality attributes are [1]:

- Code Testability
- Code Maintainability
- Code Understandability
- Code Efficiency
- Code Security

Code testability means, how easy is it to test a code. Code maintainability means, how easy is it to update a code or add a new code, without introduction of bugs. Code understandability relates to the readability of the code. Code efficiency indicates the way of utilizing the resources and finally code security depicts that, is code prone to common attacks such as buffer overruns.

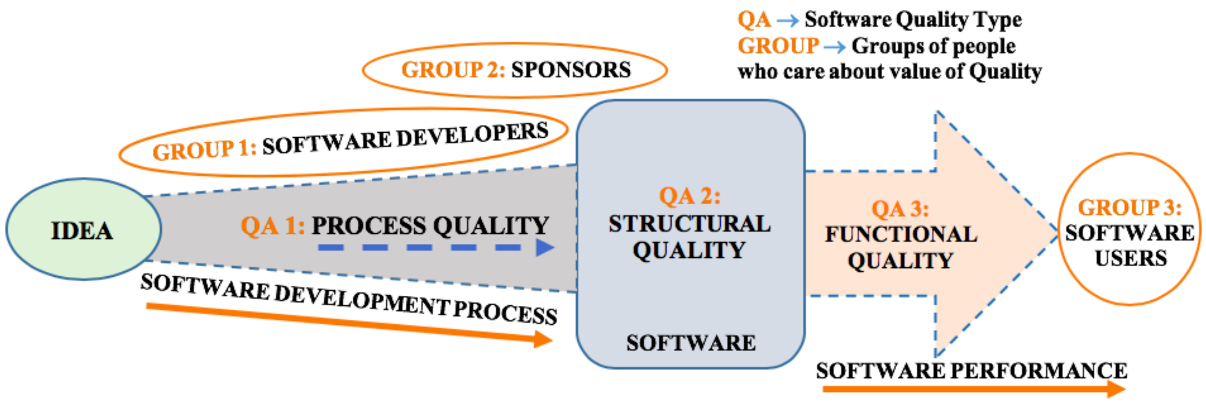

Figure 1. The Three Aspects of Software Quality: Functional, Structural, and Process

For evaluation of software there is an international standard ISO 9126. ISO 9126-1 quality model specifies 6 quality characteristics as [2]:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

Functionality relates to the satisfaction of needs. Reliability means the capability to maintain the specific level of performance. Understandability, learn ability and operability are sub attributes that are taken into consideration under usability. Efficiency describes the capability to provide the appropriate performance relative to the efficient use of resources. Maintainability relates to the capability to modify the system and portability means the adaptability to different environments.

B. Code Clones

The term *'Clone'* despite being so popularly used in the field of computer science, has its origin from Plant Physiology, coined in 1903 by Plant Physiologist Herbert J. Webber [3]. The term *'Clone'* is derived from the Greek word '*klon'*. *Code Clones* are basically copies of the original code fragment. So, clones represent duplicated fragments in the code base. Till now, in literature, we found that, there is not any unanimously agreed definition of the term *clone* in computer science and the use of term clone for syntactically or semantically similar code fragments is still an open question of debate. But, in most of the literature, authors found a mostly used definition of the term *clone*, defined by Ira D. Baxter [4] as "a clone is a program fragment that [is] identical to another fragment". The process of replicating a code fragment is referred to as 'Code Cloning'. Code fragments have been seen replicated with or without modifications and so these can be of different types. In literature, the popular classification of Clones among researchers is, given by S Bellon et al. [5], that classify clones into four types based on syntactic and semantic similarity. Based on syntactic similarity they classify clones into three types as:

Type 1: These are exact copy without modification, except white spaces and comments

Type 2: These are Type 1 syntactically identical copy with allowed modification in identifiers

Type 3: These are Type 2 clones with statements changed, added are removed

Then based on the semantic similarity they defined a new type of clones called semantic clones, and in literature this type of clones are referred as **Type 4** Clones.

III. Answers to Research Questions

Based on literature study, this section attempts to answer the Research Questions pointed in Section I.

A. Answer to RQ1: How are Code Clones related with Software Quality?

As discussed in Section II (A), the Structural Quality is related to the Code Structure i.e. it specifies whether the code is well structured or not. The concept of Code Clones as discussed in Section II (B) is related with this Structural Quality aspect of the Software Quality and thus this quality aspect in mainly addressed by code clone detection and removal, along with the functional quality aspect, because code structure also affects the software performance.

Figure 1 clearly depicts the relation of Code Clones with Software Quality, because duplicate code fragments are part of the overall code structure of the software, and this code structure is specified with the Structural Quality of the software.

With the presence of duplicate code fragments, updating and deletion of these fragments becomes very difficult, thus severely affects the maintainability of the software.

Section II (A) also discussed the six quality attributes as defined in ISO 9126 Quality Model. Most of the literature mentions that Code Clones (discussed in Section II (B)) have a direct impact on the maintainability and reliability, which are part of the six quality characteristics mentioned in ISO 9126-1 Quality Model.

Thus, in this way we can relate Code Clones with Software Quality.

B. Answer to RQ2: What is the effect of Code Clones on Software Quality?

In spite of having few advantages, most of literature mentions the harmfulness of code clones present in the software systems. These duplicated code fragments are the main cause of increasing the size of the code base, so clones cause 'Code bloating'.

With the presence of multiple copies of same code fragment, causes updating and deletion of these fragment very difficult, because cloning is not documented and thus severely affects the maintainability of the software. This causes maintenance cost to increase and Software Quality to decrease.

Code Clones have a direct impact on the Structural Quality of software and its various attributes and also maintainability and reliability, which are part of the six quality characteristics mentioned in ISO 9126-1 Quality Model.

For this reason these Code Clones have become one of the mature areas of research under the domain of software engineering. Thus these Code Clones needs to be detected and if needed removed from the software system.

C. Answer to RQ3: How are Code Clones Detected, Analysed and Removed?

Due to the harmfulness of Code Clones, these duplicated code fragments needs to be detected. After detection, these should be analysed and if needed removed from the code base, so that Software Quality can be enhanced and the maintenance cost can be set to as minimum as it can be. Based on literature study, the next three sections present various Code Clone Detection, Analysis and Removal techniques, as found in literature.

1) Code Clone Detection: As the process of clone creation is not documented and as different developers are involved in coding and maintenance of software system, so there is no way to identify where the copies of a particular code are present in the code base. To identify these code fragments different researchers gave different methods or more specifically we can say different algorithms. These algorithms implement different code similarity detection techniques, popularly known as Code Clone detection techniques. In literature, authors found a classification of these algorithms given by C. K. Roy and J. R. Cordy [6] as discussed below:

a) String Based: In this type of source code similarity detection algorithms, exact textual matches of source code segments are taken into consideration. This technique is seen to be fast but cannot find similarity when identifiers are renamed.

b) Token Based: This technique is basically same as of string based techniques but here in the case of token based techniques, the source code is first converted into tokens using lexer. This technique works a step further than string based algorithms, by discarding white spaces, comments and identifier names, thus making algorithm more robust to text replacements.

c) Parse Tree Based: With this technique higher level similarity can be detected by creating parse tree from the source code and then comparing its subtrees.

d) Program Dependency Graph (PDG) Based: To locate much higher-level equivalence among source code fragments, the actual flow of control in a program is captured and represented in the form of Program Dependency Graph.

e) Metrics Based: In this type of similarity detection, firstly various software metrics are calculated, then these metrics are compared to capture the similarity among code fragments. Various software metrics used are, for example, LOC, SLOC, No. of Loops etc.

f) Hybrid Techniques: This type of code similarity detection combines the capability of more than one code similarity detection algorithm. Present literature mostly points towards the use of this type of clone detection algorithm.

In literature there are number of different tools mentioned that implements the above mentioned code similarity detection algorithms. For example CCFinder, Duploc, CloneDr, Dup etc. These tools can be used to detect clones in a software system.

2) Code Clone Analysis: After the Code Clones are detected using any technique implemented by any tool, next step is to analyse them, so that decision can be made to deal with the results efficiently. To have a better decision, analysis must be efficient. In literature there are various visualization techniques that can be used to analyse the detected clones in software. Various visualization techniques are discussed below:

a) Scatter Plot / Dot Plot: In this type of visualization technique, the clones are represented in the form of two-dimensional charts. The two axes of the chart represent all the units of the software system under study.

b) Hasse Diagrams: This type of visualization consists of nodes and edges, where a node represents the code clone and the edge represents the relationship among them.

c) Metrics Graphs and File Similarity Tables: Visualization through metrics graphs and file similarity table allows user to browse the clones, either using clone classes or by clone pair.

d) Polymetric Views: This visualization technique provides more information about the cloning in the software system. It uses different levels of abstraction for investigation of clones.

e) Hyper-Linked Web: This technique uses the hyperlink functionality of the HTML. This technique provides very efficient navigation between source files having clone relations.

f) Coupling and Cohesion: This technique visualizes the clone relations in the architectural level, thus extending the concept of Coupling and Cohesion to code clones.

In addition to the above mentioned visualization techniques for representing the information about code clones in a software system there are other techniques also that are used for this purpose like tree maps, dependency graphs etc.

3) Code Clone Removal: After clone detection and analysis we have to make decision on clone removal, which is the main objective of the clone detection process. The removal of clones from the system is done through automatic 'refactoring' so that systems quality can be improved. Refactoring means, changing the internal structure of the software system without affecting its overall functionality. With the help of refactoring complexity can be decreased while increasing the understandability of the software system.

There are 72 refactoring patterns for refactoring source code mentioned in a book by Fowler [7]. Presently there is a refactoring catalog8] that lists 93 refactoring patterns. Out of these refactoring patterns, few patterns that are suitable for clone refactoring are discussed below:

a) Extract Method: In this method, block of code is extracted as a new method and the block is replaced with the call to that new method.

b) Pull Up Method: In this method of refactoring, all the similar methods present in subclasses are removed and pulled up to the common superclass.

c) Move Method: This type of refactoring is used to merge identical methods, with relocation of method from one class to another.

d) Extract Superclass: This type of refactoring includes extraction of two or more classes having common methods into a new common superclass.

e) Extract utility-class: In this refactoring extraction of common methods from different classes is carried out to a new class.

f) Rename Refactor: This type of refactoring involves simply renaming the names of variables, methods, classes etc.

In addition to the above mentioned refactoring patterns, there are other refactoring patterns like method parameter reordering, identifier renaming, changes in type declarations, splitting of loops, substitution of conditionals, algorithms, loops, and relocation of method or field, that are also necessary to be taken into consideration while dealing with near-miss clones i.e. similar, but not exact clones.

D. Answer to RQ4: How Software Quality is enhanced using Code Clone Detection, Analysis and Removal?

To confer to the main objective of the software engineering, i.e. to enhance the software quality with minimized maintenance as well as overall cost of the software system, the above discussed Code Clone Detection, Analysis and Removal techniques plays a vital role in achieving it. The duplicated code fragments are first detected, analysed and then removed from the software system using above mentioned code clone detection, analysis and removal techniques.

With the induction of code clones in the software system, the code base of the software increases, thus making it memory inefficient. The increased LOC (Lines of Code) has a severe effect on the efficiency of the software by increasing the memory usage. The removal of replicated fragments can enhance the efficiency of the system.

In case of updating any code fragment, and if it has replicas in the system, then updating is also very difficult, as each code clone fragment needs to be updated. But by using clone removal technique i.e. refactoring, all replicas are extracted into a single fragment, thus there is a need of updating only one fragment, thus decreases the maintenance cost, with increased quality.

As there is no documentation of the replicated fragments present in the software system, to update all the clone fragments seems to be impossible, so clone detection and removal techniques can be useful in this case.

If there is a defect in a code fragment i.e. if it contains a bug and if it is duplicated by the developer, may be by copy and pasting, then the defect or bug will be propagated into the system. To remove the bug or correct the defect, it needs to be done in each copy of the duplicated code fragment, that is very inefficient to the maintenance point of view. In this case code clone detection techniques are used to detect the duplicated code fragments and then the refactoring is performed, so that maintenance cost can be minimized.

Code Clones also cause the complexity of the software to get increased. With increased technical depth of the software system, its maintenance becomes difficult. Code Clone Removal with the refactoring helps in minimizing the technical depth of the software.

Code clones if not removed from the system affects the evolvability of the software. Thus with clone detection and removal, the effect on evolvability of the software system can also be minimized.

Thus, with code clone detection and removal, performance of the software system in terms of time and space complexity can be enhanced, and with the minimization of the maintenance effort, the maintenance cost of the system is also decreased. With the refactoring techniques, system understandability is enhanced along with the decreased effort in modification or improvement of the software system.

Thus, in this way, the Software Quality is enhanced with the help of Code Clone Detection, Analysis and Removal techniques.

IV. Conclusion

In this paper, authors attempted to discuss the Software Quality enhancement using the Code Clone Detection, Analysis and Removal techniques. This paper started with the creation of four basic Research Questions and then authors attempted to answer these question, based on literature study. This paper discussed the three aspects of the Software Quality viz. structural, functional and process quality. Then authors discuss the relation of the Structural Quality aspect with the Code Clones and provide an overview of the Code Clone literature, including Code Clone Detection, Analysis and Removal. Finally this paper discusses the various consequences of code clones and how these are addressed with code clone detection, analysis and removal, so, discussing how a Quality of the Software System is enhanced with minimized maintenance cost.

This paper discusses how code clones are related with Software Engineering and thus attempts to present the concept of Code Clones as one of the research areas under the domain of software engineering, so, this paper may serve as the potential roadmap for young researchers who want to choose Code Clones as their research area.

Acknowledgment

Authors are very thankful to the University Grants Commission, Govt. of India, for providing financial assistance for the current research in the form of fellowship, and also wish to thank the Department of Computer Science, Guru Nanak Dev University, Amritsar for providing the needed infrastructure for the ongoing research.

References

[1] David Chappell. David Chappell & Associates Website. [Online]. http://www.davidchappell.com/writing/white_papers/The_Three_Aspects_of_Software_Quality_v1.0-Chappell.pdf

[2] Wikipedia, the free Encyclopedia. [Online]. https://en.wikipedia.org/wiki/ISO/IEC_9126

[3] NPR. [Online]. http://www.npr.org/2011/03/11/134459358/Science-Diction-The-Origin-Of-The-Word-Clone

[4] Ira D. Baxter, Andrew Yahin, Leonardo Moura, Marcelo Sant' Anna, and Lorraine Bier, "Clone Detection Using Abstract Syntax Tree," in Proceedings of 14th International Conference on Software Maintenance(ICSM'98), Bethesda, Mayland, 1998, pp. 368 - 377.

[5] Stefan Bellon, Rainer Koschke, Giuliano Antoniol, Jens Krinke, and Ettore Merlo, "Comparision and Evaluation of Clone Detection Tools," IEEE Transaction on Software Engineering, vol. 33, no. 9, pp. 577 - 591, 2007.

[6] Chanchal K. Roy and James R. Cordy, "A Survey on Software Clone Detection Research," Queen's University, Kingston, Technical Report 2007-541, 2007.

[7] Martin Fowler, Refactoring: Improving the design of Existing Code.: Addison Wesley, 1999.

[8] Martin Fowler. Catalog of Refactoring. [Online]. http://refactoring.com/catalog/

Mr. Sarveshwar Bharti is presently working at the Department of Computer Science, Guru Nanak Dev University, Amritsar, India, as a Phd Research Fellow. He has received his Master of Computer Applications (MCA) degree from University of Jammu, Jammu, India. He is a Software Engineering Researcher with research interests including Software Clones, Integrated Clone Management, and Clone Management Plug-in.

Dr. Hardeep Singh is a Professor and Head at the Department of Computer Science, Guru Nanak Dev University, Amritsar, India. His research interests lie within Software Engineering and Information Systems. He has been awarded with various prestigious awards including Dewang Mehta Award for best Professor in Computer Engineering, ISTE Award for Best Teacher in Computer Science and Rotract International Award for best Teacher.