Informatics Bulletin, Faculty of Computers and Artificial Intelligence, Helwan University
Published Online ***Vol 3 Issue 1, January 2021***
(https://fcihib.journals.ekb.eg)

# A Survey of Refactoring Impact on Code Quality

Amany M. Draz
Software Engineering department,
Faculty of Computers and Information,
Suez University, Suez, Egypt
amany.mohamed@suezuni.edu.eg

Marwa S. Farhan
Information system department,
Faculty of Computers and Artificial Intelligence,
Helwan University, Cairo, Egypt
marwa.salah@fci.helwan.edu.eg

Mai M. Eldefrawi
Information system department,
Faculty of Computers and Artificial Intelligence,
Helwan University, Cairo Egypt
mai.eldefrawi@fci.helwan.edu.eg

***Abstract*— Code refactoring is an important procedure for improving the code quality; it is used to improve the internal structure without changing the external behavior of software. It is achieved through various techniques e.g., extract methods, move methods, rename methods, and classes in code. The main goal is how code refactoring can be used to improve the quality. Moreover, this study investigates the impact of code refactoring on software internal and external quality attributes. As the previous studies focus on the selected groups of either internal or external quality attributes, the elected refactoring techniques can be used to measure the impact on code quality enhancement. According to the previous studies illustrated in this survey, we demonstrate both internal and external quality attributes that have positive or negative influences on the improvement of code quality. This paper shows the internal quality attributes, which have a positive improvement after refactoring operations such as complexity, inheritance, coupling, and cohesion except size. In addition, the external quality attributes having improvement on code quality are maintainability, reusability, understandability, and efficiency, except for performance has a negative effect.**

***Index Terms*—code quality, refactoring, internal quality attributes, external quality attributes, software metrics.**

## I. INTRODUCTION

Complexity of software systems has quickly increased leading software companies to anticipate constant change. Due to stakeholder's sustained requirements, companies will be under continual pressure [1]. Software systems expose to not only high levels of complexity but also different weaknesses in code quality and sometimes failure. Developers spend more than 60% of their time understanding the code before maintenance, which means a huge cost [2]. The quality of code plays an important role in software development. Provided the quality of a code reaches the mark, it will be available, readable, testable and maintainable; any high-quality code can be re-developed and re-used. In case its quality has very low standards, the code will never last and will use more time and energy [3].

Code Quality is a measurement in which there are some principles and standards that together assure it and has good qualities e.g., clarity, maintainability, documentation, refactoring, well-tested, efficiency, usability, and performance [4].

Refactoring is one of the important measurements that improve code quality; it is used to modify the internal structure of software component in such a way that is more understandable and easily modifiable without making any changes in the external behavior of software [5]. Developers improve the structure of the code by applying some upgrades, such as inserting code lines, which indicate adding the latest functionality and reduce software complexity by repairing these errors [5]. The basic objective of software refactoring is the safe transformation to enhance quality. The advantage of code refactoring is used to improve code quality by directly measuring software metrics and indirectly the quality attributes of software and enhancing its performance [6].

Fowler [7] identifies refactoring techniques used to refine the code. For the implementation of appropriate quality attributes and metrics, different refactoring techniques are created to keep the quality of code high such as move method, inline class method, rename method, duplicate observed data method, extract subclass and extract interface method, etc.

Software quality is considered the basic standard by which software includes a required set of properties used to improve code quality and it is described as software characteristics by which quality is represented and evaluated. It is divided into two sets: internal and external software quality attributes.

Internal software quality attributes are defined as principles of code quality evaluation according to the principles of software design and clean code such as cohesion, coupling, complexity, size, and inheritance [8]. Measuring the internal quality attributes employs software quality metrics that are used for estimating the value of the software quality attributes such as weighted methods per class (WMC), the coupling between object classes (CBO) and, lines of code (LOC) which are defined as follows:

1) Response for Class (RFC): this is the number of methods in an entity that can be executed when a message is received by an object in this class.
2) Weighted Methods per Class (WMC): it represents the number of methodological complexities.
3) Lack of cohesion between methods (LCOM): the number of methods in a class, which does not have at least one class.

4) Line of Codes (LOC): the number of lines in the code, except for abandoned lines and comments, is counted.
5) Coupling between object classes (CBO): this indicates the number of classes attached to a given class. Via field access, arguments, method calls, return type, and exceptions.
6) Cyclomatic Complexity (CC): it indicates the complexity of a program. The number of linearly independent paths measures it. Also, it is called McCabe.
7) Depth of inheritance tree (DIT): the length of the hierarchy tree from the class to the parent class is calculated.
8) Number of Children (NOC): this calculates the number of the class's immediate descendants.

These software quality attributes are represented in the category with its software metrics to be defined as shown in Table I.

Table I
Internal Quality Attributes and Its Software Metrics [9]

| Quality Attribute | Software Metrics |
|---|---|
| Cohesion | LCOM |
| Coupling | CBO<br>RFC |
| Complexity | CC<br>WMC<br>RFC<br>LCOM<br>Max Nest |
| Inheritance | DIT<br>NOC |
| Polymorphism | WMC<br>RFC |
| Encapsulation | WMC<br>LCOM |
| Abstraction | WMC<br>LCOM |
| Design Size | LOC<br>CLOC<br>STMTC<br>NIM |

External software quality attributes are used to evaluate software systems such as refactoring, maintainability, readability, performance, security, reliability, and testability. Fig 1 demonstrates the relation between software internal and external quality attributes according to ISO / IEC-25010 standard [10].

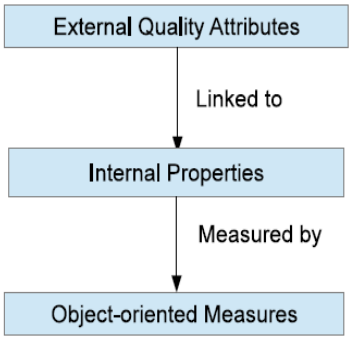


Fig.1. The relation between software Internal and External Attributes [10]

The goal of this survey is to focus on how refactoring enhances code quality and investigates its impact on both internal and external software quality attributes.

The rest of this paper is organized as follows: section 2 introduces related work, section 3 describes the research methodology, and section 4 explains the conclusion and future work.

## II. Related Work

This section provides a detailed discussion on studies related to this survey research. It covers how code refactoring can be used to improve the quality of code. Also, it investigates the impact of software refactoring on both internal and external quality aspects.

Kaur et al. [6] analyze the software project for refactoring and maintaining code collection releases for the project. The ref-Finder is used to extract the refactoring code as well as using it as a method not only to compare the effects of previous releases but also to evaluate the present releases. The code maintainability index is used to measure the code metrics for each update. Adding some new functionality and using better-refactoring approaches measures code metrics and compares the latest release output to the current release results.

Kannangara et al. [11] investigate the effect of code refactoring on software quality with five metrics by using ten techniques. The result showed that refactoring has a positive impact on software maintainability.

In another research, Kannangara et al. [12] aimed at proving the request through which refactoring enhanced the software quality. The objective is achieved by using both an experimental study and refactoring techniques chosen for analysis. The effect of each refactoring is developed based on external estimates, and they are: analyzability, the behavior of time, and the deployment of the resource. After the results of the experiment were analyzed," Polymorphism with Replace Conditional "was arranged between the top 10 refactoring of techniques tested as it showed that the percentage of enhancement in the quality of the code is high. Whereas "Introduce Null Object" was classified as the worst on the quality of the code, the results of the test hypotheses indicated that the analyzability of the refactored code is less than the non-refactored code for all the refactoring techniques tested, except for "Replace conditional polymorphism."

Demeyer [13] develops a comparative study for checking the influence of refactoring operation on software quality attributes such as performance. It was found that the performance of the software is improved after refactoring the application.

Leitch et al. [14] check the effect of the refactoring techniques such as extract and move methods through using the dependency graph on maintainability attributes by two applications. They found that refactoring of software can improve software quality by decreasing design size, increasing

the number of operations, minimizing dependencies of date, and reducing software testability.

Bios et al. [15] establish a framework for analyzing the influence of three software refactoring techniques on five internal quality features of software such as cohesion, coupling, complexity, inheritance, and size. The results showed that refactoring has positive and negative effects on the selected software measures.

Bios et al. [16] evaluate a set of guidelines for improving cohesion and coupling attributes. They used guidelines on a software project and found that the refactoring influence is ranged from negative to positive.

Moser et al. [17] provide internal software quality attributes such as lines of code, CK, and MCC to verify the effect of refactoring on software attributes reuse by a business application, and they found that refactoring can improve software reuse in the hard-to-reuse class.

Wilking et al. [18] propose an empirical study to verify the effect of refactoring on external quality attributes e.g., maintainability and software modification. They investigated the effect of software maintainability by including flaws in the code. They also verified the modification of software by adding new features and measuring line of code (LOC) metrics, as well as calculating the time taken to develop these features. The results showed that the effect of remodeling on maintainability and modification is unclear.

Spinellis et al. [19] developed four open-source software projects to check the impact of the change happening after refactoring the source code of these four open-source software. They found that code refactoring had no measurable effect on the software quality of the system.

Alshayeb [20] investigates the impact of refactoring on software quality attributes using quantitative analysis. The quality attributes focus on adaptability, maintainability, reusability, understandability, and testability. He applied software refactoring on three open-source software (e.g., terpPaint, UML, and Rabtpad). After these refactoring operations, the result indicated that there was no necessity to increase the quality of software.

Hegedus et al. [21] investigate the impact of software refactoring operations on the traits of testing, false vulnerability, and maintainability. The result shows that refactoring has undesirable impacts, which can affect the quality of software and reduce it.

Bavota et al. [22] employ RefFinder2 and that tool is used not only to discover the refactoring but also to record the history of the development for three open-source projects. In particular, they check the relations between refactoring and software quality. The results of the study show that 42% of the refactoring operation was affected by the smell of code; only in 7% of the items, refactoring was able to remove unpleasant code smell.

Cedrim et al. [23] propose a study that contained 25 projects for checking the enhancement of the quality. They analyze the relation between code refactoring and code smell for identifying the refactoring processes based on the addition or removal of code poor structures. The results show that only 2.24% of the reconstructions were deleted from software smell and 2.66% were new scents.

Chavez et al. [24] investigate the impact of refactoring on five software internal quality features such as cohesion, coupling, complexity, inheritance, and size using 25 measures of quality. The study shows that the processes related to root canal refactoring improve or at least do not exacerbate internal quality features. Also, 55% of operations related to dental floss refactoring are used; these processes improved the features, while the quality decreased by only 10%.

Kádár et al. [25] propose an examination of code refactoring to produce the important open dataset for software code metrics and to use refactoring across multiple versions of the seven systems. Exploring the quality trait of duplicate classes of source code and the efficiency of updating source code metrics uses rebuilding techniques. Additionally, they assessed the relationship between maintenance and rebuilding methods, and they also looked at how the refactoring effect was achieved for source code metrics. They propose a dataset that includes data refactoring and over 50 software code metrics for 37 versions for 7open-source software in the class and action level.

Shahjahan et al. [26] propose a study to improve the properties of the code by using graph theory techniques. Code refactoring is a method to improve the quality without causing any changes in its internal basic and external behavior. Hypothesis techniques are used to relate the results that occurred. In this study, response time is also enhanced. The ability to analyze, to change, the behavior of time, and the use of resources constitute the four main quality characteristics used to enhance code quality.

Vasileva et al. [27] show an effective combination of the quality of the code calculation in the software development process. Inadequacy concepts are important pre-requisites for code quality in addition to the selection of appropriate code analysis tool. This study shows that the implementation of measurement and didactic procedures in several iteration cycles can ensure the long-term integration of quality aspects. Simple refactoring techniques are used such as renames; they have been successfully used by all teams. Their investigation has shown that the time limit for working with tough refactoring techniques is very complex for inexperienced developers. They concluded that a good internal quality of the code can be achieved without a high level of effort or achievement. The quality of the code can be achieved at the start of the project as early as in the design phase in case the objective is set correctly. Therefore, they focus on the beginning phase of their future research; also, they planned to include calculations of the quality aspects of the model and successful didactic methods to improve modeling results.

Fontana et al. [28] check the effect of refactoring of clones on the internal quality attributes such as complexity, coupling, and cohesion. They used three tools of clone detection on two open-source programs, and they are: Ant and Ghantt Project. The tool used for refactoring purposes is Intellij IDEA. They

found that there was an enhancement in cohesion attribute and a decrease in coupling, complexity, and code lines after refactoring.

Shatnawi et al. [29] apply an empirical study to measure the effect of refactoring on software quality by using many techniques on two open-source software. The result of these most refactoring techniques has a positive effect on software quality however some refactoring techniques do not have any noticeable impact on its external quality.

Stroulia et al. [30] check the effect of internal software quality attributes, such as coupling and size on the quality after making refactoring of software application. The result of the attributes of coupling and size was decreased after software refactoring.

Moser et al. [31] present a technique to investigate the impact of refactoring on software reusability. Their analysis shows that code refactoring had a positive effect on the reusability of software.

Bavota et al. [32] check an empirical study on coupling attributes. The study was conducted on three Java open-source projects. The results showed that coupling measure allows it to be the best one for the model of developers about the other measures of coupling; the relations between classes are encapsulated in the source code and cannot be easy to derive it except by the only method calls.

Chapparo et al. [33] investigate the effect of software refactoring techniques on software quality of source code via Refactoring Impact Prediction (RIPE) by applying 12 refactoring operations and 11 software metrics. Also, RIPE is used to measure the effect on 8,103 metric values, for 504 refactoring operations from 15 open-source systems. The correct ratio of the measures is 38%, and the median of the measures are approximately 5% with an average 31%.

Szoke et al. [34] use the 198 commits of refactoring of five to check their impact on the software quality. The developers identified the causes which made code was changed were fixing coding bugs, enhancing the software metrics. The results of this study show that a single refactoring of operation had a negative impact on the software quality, however it makes refactoring in blocks such as fixing code bugs or enhancing the software quality metrics, which improve its quality.

AL Dallal et al. [35] propose an empirical study analysis of several, previous studies that summarize the effect of code refactoring on software quality. Their results show that the use of various refactoring techniques yielded opposite results on the quality of refactored components.

Al Dallal [36] also proposed a framework to assess the effect of various primary studies carried out on the assessment of the impact of code refactoring on software quality applications.

Jonsson in his work [37] applied on the software applications, which were simultaneously being modified and refactored. Its findings showed that refactoring had a positive impact on measures of software quality application.

Ouni et al. [38] presented a multi-criterion refactoring method where refactoring was done simultaneously in an automated manner using various techniques. Their findings showed that refactoring had a positive effect on software cyclomatic complexity.

Pantiuchina et al. [39] applied an empirical study to check The relation among seven software code metrics and the enhancement of the quality. The result of the study showed that quality metrics sometimes did not affect the quality enhancement made by developers. They found that rarely the changes of refactoring affected software quality.

In another research, Ouni et al. [40] develop an exploratory study to measure the position between quality enhancement and software metrics via 8 internal quality attributes and 27 software metrics. The results indicate that there are variety of measures of enhancement and degradation of software quality. Many of the metrics used to improve the quality attributes (e.g., cohesion, coupling, and complexity) are assigned as key software quality features capturing the developer's intentions to improve the recorded quality in the confirmation letters, but they are not assigned for some quality features.

Fernandes et al. [9] use 23 software systems with more than 29 thousand refactoring activities, and almost 50% of them were refactoring in o order to understand the effect of refactoring and re-refactoring on any level. They assess five factors that are cohesion, complexity, coupling, inheritance, and size and combine explanatory analysis to statistical studies. This analysis shows that 90% of refactoring and 100% of re-refactoring are applied to code features with at least one important attribute e.g., high elevation, which applied to code features.

Kiyak [41] presents the use of data mining with refactoring although automated refactoring does not provide the desired performance; manual refactoring is time-consuming by unsupervised learning algorithms that reduce the number of refactoring options. Refactoring had a beneficial influence on the quality of the software. It enhances code readability and code maintainability of the source code; it reduces the complexity of the software system.

Kaur et al. [8] check the effect of clones refactoring on software quality by using four various open-source software through the experimental study. The result shows that the complexity attribute is decreased after the refactoring operation. Also, the software functionality and reusability are increased, while other software attributes such as understandability, effectiveness, flexibility, and extendibility are reduced. They conclude that refactoring techniques had a positive or negative effect on software quality attributes.

Alawairdhi [7] investigates the effect of code refactoring on both internal and external quality attributes such as maintainability, performance, efficiency, and lines of code. After refactoring, six internal and four external quality measures were quantitatively evaluated for each component. The findings show that refactoring had a substantial effect on

the internal quality aspects of the application. However, the effect of the refactoring code on external quality was limited.

Table II shows the summary of previous related work about software internal and external quality attributes that are used to measure the effect of software refactoring on code quality.

Table II:
A summary of the effect of refactoring on internal software quality attributes and external software quality attributes

| Study | Year | Software Metric | Internal Quality Attributes | External Quality Attributes |
|---|---|---|---|---|
| Hegedus et al. [21 ] | 2010 | CK | Coupling, Complexity, Size | Maintainability, Testability, Error Proneness, Changeability Stability, Analyzability |
| Shatnawi et al. [29] | 2011 | CK, QMOOD | Inheritance, Cohesion, Coupling, Polymorphism, Size, Encapsulation, Composition, Abstraction, Messaging | Reusability, Flexibility, Extendibility, Effectiveness |
| Bavota et al.[32 ] | 2013 | ICP, IC-CD, CCBC | Coupling | - |
| Szoke et al.[34 ] | 2014 | CC ,NOA ,NII ,NAni LOC, NUMPAR, NMni, NA | Size, Complexity | - |
| Bavota et al. [22] | 2015 | CK, LOC, NOA, NOO C3, CCBC | Inheritance, Cohesion, Coupling Size, Complexity | - |
| Cedrim et al.[23] | 2016 | LOC, CBO, NOM, CC FANOUT, FANIN | Cohesion, Coupling, Complexity | - |
| Chavez et al.[24] | 2017 | CBO ,WMC , DIT ,NOC, TCC, FANIN, FANOUT, CINT, CDISP, CC, NPATH, Max Nest, IFANIN , CLOC, STMTC, CDL, NIV, NIM, NOPA | Inheritance, Cohesion, Coupling, Size, Complexity | - |
| Pantiuchina et al. [39] | 2018 | LCOM, CBO, WMC, RFC B&W, LOC, LCOM2, LCOM3, WOC | Cohesion, Coupling, Complexity | Readability |
| Ouni et al. [37] | 2019 | LCOM, CBO, RFC, CC, WMC, RFC, Max Nest, CLOC, CDL, NIV, NIM, STMTC, FANIN | Cohesion, Coupling, Complexity, Inheritance, Encapsulation, Abstraction, Polymorphism, Design, Size. | - |
| Fernandes et al.[9] | 2020 | CBO, LOC, LCOM, DIT, TCC | Cohesion, Coupling, Complexity, Size, Inheritance | - |

Table III:
Comparison between Related Work on Code Quality

| Reference | Approach | Result |
|---|---|---|
| Kannangara et al. [11] | Evaluating ten refactoring techniques to understand the effect on software code for quality measures(maintainability) | Code refactoring had a visibly positive effect on maintainability. |
| Kiyak [41] | Using data mining with refactoring. Although automated refactoring did not provide the required performance, manual refactoring was time-consuming by unsupervised learning algorithms. | Refactoring has had a positive effect on the quality of software applications. It enhanced code readability and code maintainability of the source code and reduced the complexity of the software system. |
| Stroulia et al. [30] | Investigating the influence of internal quality aspects such as size and coupling after making the refactoring of the software application. | size and coupling metrics decreased after refactoring |
| Moser et al. [31] | Evaluating a technique to determine the effect of refactoring on the software reusability. | Code refactoring has had a positive effect on software reusability. |
| Alawairdhi [7] | Investigating the effect of code refactoring on internal and external software quality aspects such as maintainability, performance, lines of code. | Code refactoring had a significant effect on the internal quality aspects of the application such as complexity. |

As shown in Table III, we summarize all of the related work in a nutshell; we come up with the following drawbacks that include:

1) There is still no definitive conclusion of whether the refactoring of code has a positive effect on the quality of software as well as determining its degree.
2) Most of researches have concentrated on either internal or external quality attributes. No one has extensively covered both attributes of quality and impact investigation of refactoring on every one of the quality attributes.

3) There are not any systematic researches showing experimental analysis for both external and internal quality attributes and the possible relationships between them.
4) Most of researches focus on using the analysis quantitative to measure the impact of software refactoring on both internal and external quality attribute.

## III. RESEARCH METHODOLOGY

The main goal of this study is to measure the impact of refactoring on code quality using various internal and external software quality attributes to enhance its quality. To achieve the above goal, the study is conducted using two attributes: internal and external attributes. Most of the previous studies focus on either internal or external attributes. Thus, this study mainly focuses on measuring both scales separately to measure the effect of the refactoring on the quality of the code.

Fowler in his study [7] classifies the 72 types of refactoring techniques. In our study we identify the 10 techniques of refactoring that can be used, and they are as follows:

1) Introduce Local Extension
2) Duplicate Observed Data
3) Replace Type Code with Subclasses
4) Replace Type Code with State/Strategy
5) Replace Conditional with Polymorphism
6) Introduce Null Object
7) Extract Subclass
8) Extract Interface
9) Form Template Method
10) Push Down Method

To apply 10 refactoring techniques to the source code project, we identify the application that is implemented. The projects are legacy systems.

The selected internal and external quality attributes are:

Internal quality attributes used to enhance the effect of code refactoring on software quality. The attributes chosen from ISO Quality organization used in our study are: Complexity, Inheritance, Coupling, Cohesion, and size. To measure the effect of refactoring on external quality attributes, the five quality attributes used in this study are maintainability, reusability, understandability, efficiency, and performance.

The software metrics are used to measure the effect of refactoring on internal software quality in our study are Cyclomatic Complexity, Depth of Inheritance, Class Coupling, Lack of Cohesion of Methods, and Line of Code.

The following steps show the process of refactoring that is applied to measure the effect of software refactoring on both internal and external quality attributes for enhancing the code quality:

1) Identifying the code that should be refactored.
2) Determining the refactoring techniques that be applied.
3) Undertaking that the applied refactoring techniques maintain the behavior of software after refactoring.
4) Applying the selected refactoring techniques.
5) Evaluating the impact of refactoring techniques on both internal and external software quality attributes.
6) Keeping the identification between the software that was refactored and other software configurations.

Fig 2 illustrates the flow chart of refactoring steps applied to measure the code quality.

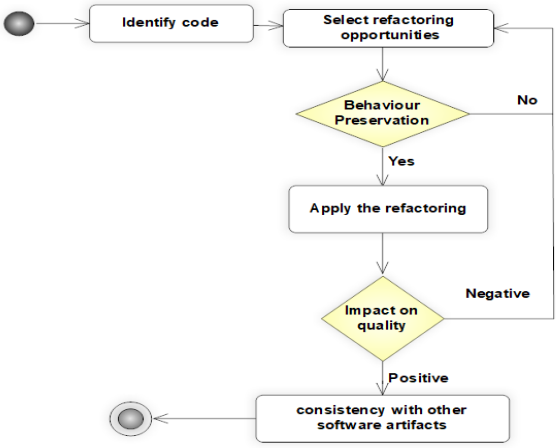


Fig. 2. Flow chart of refactoring steps on code quality

The results of the study show that there are positive improvements in code quality of the internal quality attributes after refactoring operations as the effect of refactoring is positive on the selected quality attributes except size. Also, the external quality attributes show that the improvement is in the maintainability, reusability, understandability, and efficiency, except performance has a negative effect.

## IV. CONCLUSION

This paper provides an extensive summary of research in the area of code quality. We came to the fact that code quality is essential for good software development; according to the survey, we focus on code refactoring that can be used to enhance the code quality. Also, it investigates the effect of refactoring on both internal and external software quality attributes. In this survey, we show the internal and external quality attributes that have whether a positive or negative effect on improvement of the code quality. Internal quality attributes having an improvement on the quality of code after refactoring operations are complexity, inheritance, coupling, cohesion except size. Additionally, the external quality attributes that have improvement on the code quality are maintainability, reusability, understandability, and efficiency, except for performance has a negative effect. After going through the current studies, we came across two limitations: one of them is that most researches rely on quantitative analysis rather than experimental analysis with code refactoring techniques. The second limitation is that researchers focus on either internal or external quality attributes. Moreover, there is no systematic research that shows the experimental analysis for both external and internal quality attributes and the possible relationships between them. In future work, we intend to increase the scope of work by focusing on more internal and external software quality measures from the ISO quality organization.

## REFERENCES

[1] S. Ling, and A. Finkelstein," Anticipating Change in Requirements Engineering "In; /: *Relating Software Requirements and Architectures*, Springer, Berlin, Heidelberg, 2011, DOI: https://doi.org/10.1007/978-3642-21001-3_3.

[2] J. Börstler, "I know it when I see it–Perceptions of Code Quality," in Proc of ACM Conference on Innovation and Technology in Computer Science Education, Bologna, Italy, 2017, DOI: http://dx.doi.org/10.1145/3059009.3081328.

[3] N. Rufus," Importance of Code Quality," 2019. [Online]. Available: https://www.infognana.com/importance-of-code-quality/

[4] M. Török, "Comprehensive Guide to Code Quality Best Practices and Tools,"2017[Online].Available:https://codingsans.com/blog/code-quality.

[5] A. Bibiano, E. Fernandes, D. Oliveira, and A. Garcia, "A Quantitative Study on Characteristics and Effect of Batch Refactoring on Code Smells," *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Porto de Galinhas, Recife, Brazil, 2019, pp. 1-11, DOI: 10.1109/ESEM.2019.8870183.

[6] G. Kaur, and B. Singh, "Improving the quality of software by refactoring," *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, 2017, pp. 185-191, DOI: 10.1109/ICCONS.2017.8250707.

[7] M. Alawairdhi, "Code Refactoring and its Impact on Internal and External Software Quality: An Experimental Study," IJCSNS International Journal of Computer Science and Network Security, VOL.19, No.6, June 2019.

[8] P. Kaur, and P. Mittal," Impact of Clones Refactoring on External Quality Attributes of Open Source Software," International Journal of Advanced Research in Computer Science, Vol. 8, No.5, June .2017.

[9] E. Fernandes, A. Chavez, A. Garcia, I. Ferreira, D. Cedrim, L. Sousa, and W. Oizumi, "Refactoring Effect on Internal Quality Attributes: What Haven't They Told You Yet?, "Information and Software Technology, 2020, DOI: https://doi.org/10.1016/j.infsof.2020.106347.

[10] R. Jabangwe, and J. Börstler, Smite, "Empirical evidence on the link between object-oriented measures and external quality attributes: a systematic literature review". Empirical Software Engineering 20, 640–693, 2015. https://doi.org/10.1007/s10664-013-9291-7

[11] S. Kannangara, and W. Wijayanayake, "Measuring the Impact of Refactoring on Code Quality Improvement Using Internal Measures," In Proc. of the International Conference on Business & Information, Sri Lanka, December 2013.

[12] S. Kannangara, and W. Wijayanayake," Impact of Refactoring on External Code Quality Improvement: An Empirical Evaluation", In Proc. of International Conference on Advances in ICT for Emerging Regions, Sri Lanka, 2013.

[13] S. Demeyer," Maintainability versus performance: What's the Effect of introducing polymorphism," 2003.

[14] R. Leitch and E. Stroulia," Assessing the maintainability benefits of design restructuring using dependency analysis," In Proceedings.5th International Workshop on Enterprise Networking and Computing in Healthcare Industry (IEEE Cat. No. 03EX717), 2003.

[15] B. Bois, and T. Mens," Describing the impact of refactoring on internal program quality," In International Workshop on Evolution of Large-scale Industrial Software Applications, 2003.

[16] B. Bois, S. Demeyer, and J. Verelst, "Refactoring-improving coupling and cohesion of existing code, "In 11th working conference on reverse engineering, IEEE, 2004.

[17] R. Moser, A. Sillitti, P. Abrahamsson, and G. Succi," Does refactoring improve reusability?" In International Conference on Software Reuse, Springer, 2006.

[18] D. Wilking, U. Kahn, and S. Kowalewski," An empirical evaluation of refactoring," e-Informatics, 1(1):27–42, 2007.

[19] K. Stroggylos, and D. Spinellis, "Refactoring – does it improve software quality?" In Proceedings of 5th International Workshop on Software Quality (WoSQ'07: ICSE Workshops), 10–16, 2007.

[20] M. Alshayeb," Empirical investigation of refactoring effect on software quality," Information and software technology, 51(9):1319–1326, 2009.

[21] G. Hegedus, G. Hrabovszki, and I. Siket, "Effect of object-oriented refactoring on testability, error proneness, and other maintainability attributes," In Proceedings of the 1st Workshop on Testing Object-Oriented Systems, ACM, 2010.

[22] G. Bavota, A. Lucia, M. Penta, R. Oliveto, and F. Palomba, "An experimental investigation on the innate relationship between quality and refactoring, "Journal of Systems and Software, 107:1–14, 2015.

[23] D. Cedrim, L. Sousa, A. Garcia, and R. Gheyi, "Does refactoring improve software structural quality? A longitudinal study of 25 projects," In Proceedings of the 30th Brazilian Symposium on Software Engineering, ACM, 2016.

[24] A. Chávez, I. Ferreira, E. Fernandes, D. Cedrim, and A. Garcia., "How does refactoring affect internal quality attributes? A multi-project study, "In Proceedings of the 31st Brazilian Symposium on Software Engineering, 2017.

[25] I. Kádár, and P. Hegedus, "A Code Refactoring Dataset and Its Assessment Regarding Software Maintainability," *IEEE 23rd international conference on software Analysis*, 2016.

[26] A. Shahjahan, "Impact of Refactoring on Code Quality by using Graph Theory: An Empirical Evaluation," pp. 595–600, 2015.

[27] A. Vasileva, and D. Schmedding, "How to Improve Code Quality by Measurement and Refactoring," 2016.

[28] F. Fontana, M. Zanoni, A. Ranchetti, and D. Ranchetti,"Software Clone Detection and Refactoring," ISRN Software Engineering, 2013.

[29] R. Shatnawi, and W. Li, "An Empirical Assessment of Refactoring Impact on Software Quality Using a Hierarchical Quality Model," International Journal of Software Engineering and Its Applications, 5(4):127–149, 2011.

[30] E. Stroulia, and R. Kapoor," Metrics of refactoring-based development: An experience report," In OOIS 2001, Springer, 2001.

[31] R. Moser, P. Abrahamsson, W. Pedryc, A. Sillitti, and G.Succi," A case study on the impact of refactoring on quality and productivity in an agile team", In Proceeding of the Central and East-European Conference on Software Engineering Techniques, Poznan, Poland, 2007.

[32] G. Bavota, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, "An empirical study on the developers' perception of software coupling," 2013 35th International Conference on Software Engineering (ICSE),SanFrancisco,CA,2013,pp.692701,DOI:10.1109/ICSE.2013.6606615..

[33] O. Chaparro, G. Bavota, and A. Marcus, "On the impact of refactoring operations on code quality metrics," Software Maintenance and Evolution (ICSME), IEEE International Conference on IEEE, 2014.

[34] G. Szóke, G. Antal, C. Nagy, R. Ferenc, and T. Gyimóthy," Bulk fixing coding issues and its effects on software quality: Is it worth refactoring? ,"In IEEE 14th International Working Conference on Source Code Analysis and Manipulation, IEEE, 2014.

[35] J. Al Dallal, and A. Abdin, "Empirical Evaluation of the Impact of Object-Oriented Code Refactoring on Quality Attributes: A Systematic Literature Review," IEEE Transactions on Software Engineering, 2017.

[36] J. Al Dallal, "Evaluating quality of primary studies on determining object-oriented code refactoring candidates," Proceedings of the International Conference on Engineering & MIS ACM, 2015.

[37] A. Jonsson,"The Impact of Refactoring Legacy Systems on Code Quality Metrics," 2017.

[38] A. Ouni, M. Kessentini, and H. Sahraoui,"Multi-criteria code refactoring using search-based software engineering: An industrial case study," ACM Transactions on Software Engineering and Methodology, 2016.

[39] J. Pantiuchina, M. Lanza, and G. Bavota, "Improving Code: The (Mis) Perception of Quality Metrics, "*IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Madrid, 2018, pp. 80-91, DOI: 10.1109/ICSME.2018.00017.

[40] A. Ouni, E. Alomar, M. Mkaouer, and M. Kessentini, "On the Impact of Refactoring on the Relationship between Quality Attributes and Design Metrics," *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Recife, Brazil, 2019, pp. 1-11, DOI: 10.1109/ESEM.2019.8870177.

[41] E. Kiyak," Data Mining and Machine Learning for Software Engineering", Data Mining - Methods, Applications, and Systems, March 5th, 2020, DOI: 10.5772/*intechopen.*91448, [Online]. Available: https://www.intechopen.com/online-first/data-mining-and-machine-learning-for-software-engineering.