Quality Assessment for Primary Studies on Identifying Refactoring Opportunities in Object-Oriented Code

Jehad Al DALLAL

j.aldallal@ku.edu.kw
(Geliş/Received: 29.10.2015; Kabul/Accepted: 05.04.2016)
DOI: 10.17671/btd.00357

***Abstract*—** Refactoring is a maintenance task that refers to the process of restructuring software source code to enhance its quality without affecting its external behavior. Manually inspecting and analyzing the source code of the system under consideration to identify refactoring opportunities are time consuming and costly process. Researchers typically propose fully or semi-automated techniques to predict or identify the refactoring opportunities. In the present study, we demonstrate an application for a proposed framework that assesses the quality of the studies reported in the published primary studies (PSs) on the refactoring prediction/ identification techniques. The framework is applied on 47 selected PSs to assess the quality of the studies based on their design, conduct, analysis, and conclusion. We used the results to comment on the weakness of the existing PSs and the issues that have to be considered with much attention in the future studies.

***Keywords*—** Refactoring activity, refactoring opportunity, object-oriented code, study quality assessment.

1. INTRODUCTION

Refactoring refers to altering the internal structure of an existing object-oriented software code without changing its external behavior [21] and it aims at improving several quality attributes, of a considered code, such as testability and maintainability [21]. Fowler [21] defined several refactoring scenarios and explained how they can be applied. Researchers have identified four stages during which refactoring process is typically carried out [33, 40, 53]. In the first stage, the refactoring candidates are determined. In the second stage, the advantage and cost of carrying out the refactoring are analyzed. In the third stage, the consequent code modification is performed, and in the last stage, the performed code modification is examined for its preservation for the external software behavior. In this paper, we are interested in assessing the quality of studies that propose or apply techniques related to the first refactoring stage. Assessing the quality of studies related to techniques applied during other refactoring stages can be considered in future researches. This paper considers 47 papers identified by Al Dallal [2] as the PSs that empirically evaluate techniques for identifying refactoring opportunities. An assessment framework proposed by Al Dallal [2] is applied to assess the quality of selected PSs in terms of their design, conduct, analysis, and conclusion. The obtained results are useful for software engineers to select the best available techniques once performing refactoring. In addition, the framework is useful for researchers in the area of software refactoring as it guides them to perform and report studies that are potentially practical and trustable. This paper extends the conference version paper [54] that introduces the considered framework but with less details.

This paper is organized as follows. Section 2 provides an overview of the assessment framework. Section 3 reports and discusses the assessment results. Finally, Section 4 presents our conclusions.

2. ASSESSMENT FRAMEWORK

We followed the guidelines suggested by Kitchenham and Charters [28] to construct the quality checklist given in Table 1. The checklist includes 18 quality assessment questions, considers several quality aspects, including the study design, conduct, analysis, and conclusion, and was applied to assess the corresponding quality of each of the PSs considered. Each question is evaluated as "Yes", "Partially", or "No" with a corresponding score of 1, 0.5, or 0, respectively. For example, regarding QA1, the score was given as "0" if the author did not mention which identification technique was applied in the study, a score of "0.5" was reported if the author indicated which technique was applied but did not describe the technique or described the technique unclearly, and a score of "1" was given if the author clearly described the applied technique. For QA2,

the score was given as "0" if the author did not mention which refactoring activities were applied, a score of "0.5" was reported if the author reported the applied refactoring activities but did not define them, and a score of "1" was given if the author stated and defined the applied refactoring activities. Some of the questions were not applicable to some PSs; these PSs were not evaluated for those questions. For example, some PSs did not use any statistical methods, and therefore, the corresponding questions QA10 and QA11 were not evaluated.

Table 1: PS quality assessment questions

ID	Question
Design	
QA1	Are the applied identification techniques for refactoring opportunities clearly described?
QA2	Are the refactoring activities considered clearly stated and defined?
QA3	Are the aims of the study clearly stated?
QA4	Was the sample size justified?
QA5	Are the evaluation measures fully defined?
Conduct	
QA6	Are the data collection methods adequately described?
Analysis	
QA7	Are the results of applying the identification techniques evaluated?
QA8	Are the data sets adequately described? (size, programming languages, source)
QA9	Are the study participants or observational units adequately described?
QA10	Are the statistical methods described?
QA11	Are the statistical methods justified?
QA12	Is the purpose of the analysis clear?
QA13	Are the scoring systems (performance evaluation) described?
Conclusion	
QA14	Are all study questions answered?
QA15	Are negative findings presented?
QA16	Are the results compared with previous reports?
QA17	Do the results add to the literature?
QA18	Are validity threats discussed?

The research assistant extracted and reported the data corresponding to the study quality assessment. In addition, for each PS, the research assistant highlighted the paper text related to each quality assessment question and provided his assessment score (i.e., 0, 0.5, or 1). The author checked all this work. Disagreements were discussed until a consensus was reached.

To assess a PS, we added the scores for each question and found the percentage over the applicable questions for that PS. For example, the summation of the scores for S15, as shown in Table 3, is 8 and the number of applicable questions is 16, and thus, the overall percentage is 50%.

3. ASSESSMENT RESULTS

We applied the assessment framework given in Table 1 to evaluate the quality of the 47 selected papers given in Table 2. The framework application results are provided in Table 3. The results of the quality assessment study show that the scores of the PSs range widely from 33.3% to 94.4% with an average of 69.2%. More specifically, the scores of conference PSs range between 44.4% and 88.9% with an average of 64.2%, and the scores of journal PSs range from 33.3% to 94.4% with an average of 73.5%, considerably better than the conference PSs. We found that 14 journal PSs (52.4%) scored above 77.8% (higher than the second best conference PS score). These results were expected and confirm the results found in other literature reviews (e.g., [41]) that, with some exceptions, journal papers are typically more complete and of better quality than conference papers for several reasons, including space limitations, number of pages, and the depth required of the reported empirical evaluation. For interested readers, we recommend the six PSs (S8, S10, S37, S6, S1, and S27, in order from the highest) that scored higher than 85%.

Table 2: Mapping between the identifiers and references of the PSs

S1: [1]	S13: [14]	S25: [27]	S37: [43]
S2: [3]	S14: [15]	S26: [29]	S38: [44]
S3: [4]	S15: [16]	S27: [30]	S39: [45]
S4: [5]	S16: [17]	S28: [31]	S40: [46]
S5: [6]	S17: [18]	S29: [32]	S41: [47]
S6: [7]	S18: [19]	S30: [34]	S42: [48]
S7: [8]	S19: [20]	S31: [35]	S43: [49]
S8: [9]	S20: [22]	S32: [36]	S44: [50]
S9: [10]	S21: [23]	S33: [37]	S45: [51]
S10: [11]	S22: [24]	S34: [38]	S46: [52]
S11: [12]	S23: [25]	S35: [39]	S47: [53]
S12: [13]	S24: [26]	S36: [42]	

The results in Table 3 show that all of the PSs either partially or adequately described the techniques applied for identifying refactoring opportunities (QA1), stated the aims

of the study (QA3), and answered all study questions (QA14). Most of the PSs adequately described the data sets (QA8), clearly stated the purpose of the analysis (QA12), and reported analysis results that sufficiently add to the literature (QA17).

Table 3: Assessment Results

ID	PS Quality Assessment Results																		Total
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
S1	1	1	1	1	1	1	1	1	0.5	1	1	1	1	1	0	0	1	1	86.1
S2	1	1	1	0.5	1	1	1	1	0.5	1	1	1	1	1	0	0	1	1	83.3
S3	1	0.5	1	0	0	0	0	1	0	1	0.5	1	0	1		0	0.5	0	44.1
S4	1	0.5	1	0	0	0.5	1	1	0	1	1	0	0	1	0.5	0	0.5	0	50.0
S5	1	0.5	1	0	0	0.5	1	1	0.5	1	0.5	1	0	1		0	0.5	0	55.9
S6	1	1	1	1	1	0.5	1	1	0.5	1	1	1	0	1	1	1	1	1	88.9
S7	1	0.5	1	0.5	1	1	1	0.5	1	1	1	1	0.5	1	1	0	1	1	83.3
S8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	94.4
S9	1	1	1	1	0.5	1	1	0.5	1	1	1	1	0	1	1	0	1	1	83.3
S10	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0.5	1	1	1	91.7
S11	1	0.5	1	0	0	0.5	1	0.5	0.5	1	1	1	0	1	0	1	1	0	61.1
S12	1	1	1	0	0.5	1	1	0.5	1	1	1	0.5	0	0.5	1	0	1	0	66.7
S13	1	1	1	0.5	0.5	1	1	1	0.5	1	1	1	0	1	1	1	1	0	80.6
S14	1	1	1	0	1	0.5	1	1	0.5	1	1	1	0	1	1	1	1	0	77.8
S15	0.5	1	1	0	0	0.5	1	0.5	0.5			1	0	1	0.5	0	0.5	0	50.0
S16	1	1	1	0	0.5	1	1	0.5	0.5	1	0.5	1	0	1	1	0	0.5	0	63.9
S17	1	0.5	1	0.5	0	1	1	0	0.5	0.5	0	1	0	1	0.5	0	1	0	52.8
S18	1	1	1	0.5	0.5	1	1	1	1	1	1	1	0	1	1	0	1	1	83.3
S19	0.5	0.5	1	0	0	1	1	0.5	0.5	0	0	1	0	1	1	0	0	0	44.4
S20	1	1	1	0.5	0	0.5	0	1	0.5	1	0.5	0.5	0	1	0	0	1	0	52.8
S21	1	0.5	0.5	0.5	0	1	0	1	0.5			0	0	0.5	0	0	0.5	0	33.3
S22	1	1	1	1	0	0.5	1	1	0.5			1	0	1	0.5	1	1	0.5	75.0
S23	1	1	1	0	1	1	1	0.5	0.5	1	0.5	0.5	1	0.5	0	1	1	0	69.4
S24	1	1	1	0	1	1	1	1	0.5			1	1	0.5	1	0	0.5	0.5	75.0
S25	1	0.5	1	1	0	1	1	1	0.5			1	0	1	0.5	0	1	0.5	68.8
S26	1	0.5	1	1	0	1	0	1	1			1	0	1	0	0	1	1	65.6
S27	1	0	1	1	1	1	1	1	0.5	1	0.5	1	0.5	1	1	1	1	1	86.1
S28	1		1	1	1	0.5	1	1	0.5	1	1	1	1	1	0	1	1	0	77.8
S29	1	0.5	1	1	0.5	0.5	0.5	1	0			0	0	0.5		0	0.5	0	46.7
S30	1	0.5	1	0.5	0	0.5	1	1	0.5			0.5	0	1	1	0	1	0.5	62.5
S31	0.5	1	1	1	0.5	0.5	1	1	0	0.5	1	1	0	1	0	0	1	0.5	63.9
S32	1	1	1	0	1	1	1	1	0.5	1	1	1	0	1		1	1	0.5	82.4
S33	1	1	1	0	1	1	1	1	0.5	1	1	1	0	1		1	1	0.5	82.4
S34	1	0.5	1	0	0.5	1	1	1	0.5	1	1	0.5	0.5	1		0	1	0.5	70.6
S35	1	1	1	0	0.5	0.5	1	1	0.5	0	0.5	1	0	0.5		0	1	0.5	58.8
S36	1	1	1	0	0	0.5	1	1	0	1	1	0.5	0	0.5	0	0	1	0	52.8
S37	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	88.9
S38	1	0.5	1	0.5	0.5	0.5	1	1	0	1	1	1	0	1	1	0	1	0	66.7
S39	1	1	1	0	0	0	1	1	0	1	1	1	0	1		1	1	0	64.7
S40	1	0.5	1	1	0	0	0	1	1			1	0	1	0	1	1	0.5	62.5
S41	1	0.5	1	0	0	1	0	1	1			0.5	0	1	0	0	0.5	0	46.9
S42	1	1	1	0	0.5	1	1	1	1			1	0	1	1	0	1	0.5	75.0
S43	1	1	1	1	0.5	1	1	1	0.5	1	1	1	0	1	1	0	1	0	77.8
S44	1	1	1	0.5	1	1	1	1	1			1	0	1	1	0	1	1	84.4
S45	1	1	1	0	0.5	1	1	1	1			1	0	1	1	0	1	1	78.1
S46	1	0.5	1	0.5	0.5	1	1	1	0.5	1	0.5	1	0	1	1	0	1	1	75.0
S47	1	0	1	0	0.5	1	1	0.5	1			1	0	1	1	0	0.5	1	65.6

Regarding QA2, it was found that a considerable percentage of the PSs (36.2% + 4.3% = 40.5%) did not provide definitions of the refactoring activities considered, and two PSs (S27 and S47) did not specify the refactoring activities. In PS S27, the authors stated that their technique is applicable to generalization refactoring activities without stating or defining these activities. In addition, the technique proposed in S47 is applicable for Extract Class refactoring, although the authors never used this refactoring activity name. Question QA2 is not applicable to S28 because the proposed technique was claimed to be applicable to all refactoring activities.

For QA4, we considered the total sizes of the data sets used in a PS, where large, medium, and small data sets were given values of 1, 0.5, and 0, respectively. Regarding QA5, it was found that only 31.9% of the PSs fully defined the measures applied to evaluate the refactoring results. The rest of the PSs either did not define the evaluation measures (36.2%) or defined only some of them (31.9%). These PSs either assumed that the readers were familiar with the evaluation measures used or omitted the definitions because of space limitations. Most of the PSs (80.8%) failed to describe the scoring systems for the evaluation measures and thus left the conclusions subjective. We believe that the reason for the lack of scoring system descriptions was that the authors relied on existing evaluation measures, such as *Precision* and *Recall* that do not have common standard score descriptions. For example, a score of 65 does not have a common interpretation (i.e., whether it is good, satisfactory, or poor). Describing the scoring systems (QA13) makes the conclusions objective and facilitates comparing the results with the results obtained in other studies. In addition, defining the evaluation measures is important to eliminate any ambiguity regarding the interpretation of the results.

More than half of the PSs adequately described the data collection methods (QA6), which helps to make the reported studies repeatable. Regarding QA9, most of the PSs (70.2%) failed to provide sufficient information regarding the expertise of the study participants, and the evaluation results greatly depend on such experience. The absence of such information makes the evaluation results questionable. PSs that adequately described the applied tools but reported insufficient information about the study participants were given a value of 0.5 and made up 55.3% of the PSs. Among the PSs that applied statistical methods, a high percentage clearly described the statistical methods and at least partially provided justifications for applying them (related to QA10 and QA11). Regarding QA15, we observed that more than half of the PSs with negative findings represented and interpreted the negative findings. The rest of the PSs either listed the negative findings but failed to interpret them or did not present the negative findings at all. Presenting and interpreting negative findings is important to understand the limitations of the proposed technique and thus help to find ways to improve the technique. The results of QA16 are related to the results reported in Section 4.4.4 and show that most of the PSs lack comparison studies. Finally, most of the PSs did not sufficiently discuss validity threats, which makes the causal inference and generalization of the results questionable. We noted that only 13.6% of the conference studies sufficiently discussed validity threats, whereas 48% of the journal studies did. This difference may be due to the conference papers' space limitations.

4. CONCLUSIONS

The paper presents a framework that can be applied by researchers in the field to assess the quality of the existing papers or new papers to be published. As a case study, the framework is applied on existing PSs in the field of identifying refactoring opportunities. The proposed framework can be also used as a checklist for researchers to ensure that they included the main elements and considered the key factors in their study.

Generally, the quality assessment study shows that most of the PSs scored well for questions QA1, QA3, QA7, QA8, QA10, QA11, QA12, QA14, and QA17. Researchers are advised to focus more on issues regarding questions QA2, QA4, QA5, QA9, QA13, QA16, and QA18, where we found that most of the PSs have weaknesses. Only 57%, 32%, 32%, 30%, 13%, 30%, and 32% of the papers addressed these questions, respectively, well. More specifically, researchers have to ensure that they clearly state and define the considered refactoring activities and applied evaluation measures and scoring system. In addition, they have to explicitly provide descriptions for the applied software tools and details of the study participants. Finally, researchers are advised to compare their results with existing ones, use data with relatively large enough size, and state and discuss the validity threats of their studies.

REFERENCES

[1] Al Dallal J. 2012. Constructing models for predicting extract subclass refactoring opportunities using object-oriented quality metrics, *Journal Information and Software Technology archive*. 54, 10, 1125-1141.

[2] Al Dallal J. 2015. Identifying refactoring opportunities in object-oriented code: a systematic literature review, *Information and Software Technology*. 58, 231–249.

[3] Al Dallal J. and Briand L.C. 2012. A Precise Method-Method Interaction-Based Cohesion Metric for Object-Oriented Classes, *ACM Transactions on Software Engineering and Methodology (TOSEM) TOSEM*, 21, 2, Article No. 8.

[4] Alkhalid A., Alshayeb M., and Mahmoud S. 2010. Software refactoring at the function level using new Adaptive K-Nearest Neighbor algorithm, *Advances in Engineering Software*, 41, 10-11, 1160-1178.

[5] Alkhalid A., Alshayeb M., and Mahmoud S.A. 2011. Software refactoring at the class level using clustering techniques, *Journal of Research and Practice in Information Technology*, 43, 4, 285-306.

[6] Alkhalid A., Alshayeb M., and Mahmoud S.A. 2011. Software refactoring at the package level using clustering techniques, *Software IET*, 5, 3, 276-284.

[7] Bavota G., De Lucia A., Marcus A., and Oliveto R. 2013a, Automating extract class refactoring: an improved method and its evaluation, *Empirical Software Engineering*, 1-48.

[8] Bavota G., De Lucia A,, Marcus A., and Oliveto R. 2013b. Using structural and semantic measures to improve software modularization, *Empirical Software Engineering*, 18, 5, 901-932.

[9] Bavota G., De Lucia A., and Oliveto R. 2011. Identifying Extract Class refactoring opportunities using structural and semantic cohesion measures, *Journal of Systems and Software*, 84, 3, 397-414.

[10] Bavota G., Gethers M., Oliveto R., Poshyvanyk D., and De Lucia A. 2014. Improving software modularization via automated analysis of latent topics and dependencies, *ACM Transactions on Software Engineering and Methodologies*, 23, 1, Article No. 4.

[11] Bavota G., Oliveto R., Gethers M., Poshyvanyk D., and De Lucia A. 2014. Methodbook: Recommending move method refactorings via relational topic models, *IEEE Transactions on Software Engineering*, 40, 7, 671-694.

[12] Bavota G., OlivetoR., De Lucia A., Antoniol G., and Gueheneuc Y. 2010. Playing with refactoring: Identifying extract class opportunities through game theory, In: *IEEE International Conference on Software Maintenance (ICSM)*, 1-5.

[13] Cassell K., Andreae P., and Groves L. 2011. A dual clustering approach to the extract class refactoring, In: *Proceedings on the 23rd International Conference on Software Engineering and Knowledge Engineering (SEKE'11),* Miami, FL, USA.

[14] Czibula I.G. and Czibula G.2008. Hierarchical clustering based automatic refactorings detection, *WSEAS Transactions on Electronics*, 5, 7, 291-302.

[15] Czibula I.G. and Serban G. 2006. Improving systems design using a clustering approach, *IJCSNS International Journal of Computer Science and Network Security*, 6, 12, 40-49.

[16] Du Bois B., Demeyer S., and Verelst J. 2004. Refactoring - improving coupling and cohesion of existing code, In: *Proceedings of the 11th Working Conference on Reverse Engineering*, 144-151.

[17] Fokaefs M., Tsantalis N., Chatzigeorgiou A., and Sander J. 2009. Decomposing object-oriented class modules using an agglomerative clustering technique, *IEEE International Conference on Software Maintenance*, Canada, 93-101.

[18] Fokaefs M., Tsantalis N., and Stroulia E. 2011. A. Chatzigeorgiou, JDeodorant: identification and application of extract class refactorings, In: *Proceedings of the 33rd International Conference on Software Engineering*, 1037-1039.

[19] Fokaefs M., Tsantalis N., Stroulia E., and Chatzigeorgiou A. 2012. Identification and application of Extract Class refactorings in object-oriented systems, *Journal of Systems and Software*, 85, 10, 2241-2260.

[20] Fokaefs M., Tsantalis N., Stroulia E., and Chatzigeorgiou A.2007. JDeodorant: Identification and removal of Feature Envy bad smells, In: *Proceedings of IEEE International Conference on Software Maintenance*, 467-468.

[21] Fowler M. 1999. Refactoring: improving the design of existing code, *Addison-Wesley Longman Publishing Co., Inc.*, Boston, MA.

[22] Higo Y., Kamiya T., Kusumoto S., and Inoue K. 2004. Aries: Refactoring support environment based on code clone analysis, In: *Proceedings of the 8th IASTED International Conference on Software Engineering and Applications*, Article No. 436-084, 222-229.

[23] Higo Y., Kusumoto S., and Inoue K. 2008. A metric-based approach to identifying refactoring opportunities for merging code clones in a Java software system, *Journal of Software Maintenance and Evolution: Research and Practice*, 20, 6, 435-461.

[24] Hotta K., Higo Y., and Kusumoto S. 2012. Identifying, Tailoring, and Suggesting Form Template Method Refactoring Opportunities with Program Dependence Graph, In: *Proceedings of the 16th European Conference on Software Maintenance and Reengineering*, 53-62.

[25] Kanemitsu T., Higo Y., and Kusumoto S. 2011. A visualization method of program dependency graph for identifying extract method opportunity, In: *Proceedings of the 4th Workshop on Refactoring Tools*, 8-14.

[26] Kataoka Y., Notkin D., Ernst M.D., and Griswold W.G. 2001. Automated Support for Program Refactoring using Invariants, In: *Proceedings of the IEEE International Conference on Software Maintenance*, 736.

[27] Kimura S., Higo Y., Igaki H., and Kusumoto S. 2012. Move code refactoring with dynamic analysis, In: *Proceedings of the IEEE International Conference on Software Maintenance (ICSM)*, 575-578.

[28] Kitchenham B. and Charters S. 2007. Guidelines for performing systematic literature reviews in software engineering, *Technical Report EBSE*, Keele University, UK.

[29] Lee S., Bae G., Chae S.H., Bae D., and Kwon Y.R. 2011. Automated scheduling for clone-based refactoring using a competent GA, *Software—Practice & Experience*, 41, 5, 521-550.

[30] Liu H., Niu Z., Ma Z., and Shao W. 2013. Identification of generalization refactoring opportunities, *Automated Software Engineering*, 20, 1, 81-110.

[31] Mahouachi R., Kessentini M., and Ghedira K. 2012. A new design defects classification: marrying detection and correction, In: *Proceedings of the 15th international conference on Fundamental Approaches to Software Engineering*, 455-470.

[32] Melton H. and Tempero E. 2007. The CRSS metric for package design quality, In: *Proceedings of the 30th Australasian conference on Computer science*, 62, 201-210.

[33] Mens T. and Tourwé T. 2004. A survey of software refactoring, *IEEE Transactions on Software Engineering*, 30, 2, 126-139.

[34] Mens T., Tourwé T., and Muñoz F. 2003. Beyond the Refactoring Browser: Advanced Tool Support for Software Refactoring, In: *Proceedings of the 6th International Workshop on Principles of Software Evolution*, 39.

[35] Oliveto R., Gethers M., Bavota G., Poshyvanyk D., and De Lucia A. 2011. Identifying method friendships to remove the feature envy bad smell (NIER track), In: *Proceedings of the 33rd International Conference on Software Engineering*,820-823.

[36] Pan WF., JiangB., and Li B. 2013. Refactoring software packages via community detection in complex software networks, *International Journal of Automation and Computing*, 10, 2, 157-166.

[37] Pan W., Jiang WB., and Xu Y. 2013. Refactoring packages of object-oriented software using genetic algorithm based community detection technique, *International Journal of Computer Applications in Technology*, 48, 3, 185-194.

[38] PanW., Li B., Ma Y., Liu J., and Qin Y. 2009. Class structure refactoring of object-oriented softwares using community detection in

dependency networks, *Frontiers of Computer Science in China*, 3, 3, 396-404.

[39] Pan WF., Wang J. and Wang MC. 2013. Identifying the move method refactoring opportunities based on evolutionary algorithm, *International Journal of Modelling, Identification and Control*, 18, 2, 182-189.

[40] Piveta E. 2009. Improving the search for refactoring opportunities on object-oriented and aspect-oriented software, Ph.D. thesis, *Univeridade Federal Do Rio Grande Do Sul*, Porto Alegre.

[41] RadjenovićD., Heričko M., Torkar R., and Živkovič A. 2013. Software fault prediction metrics: a systematic literature review, *Information and Software Technology*, 55, 8, 1397-1418.

[42] Rao AA. And Reddy Kn. 2011. Identifying clusters of concepts in a low cohesive class for extract class refactoring using metrics supplemented agglomerative clustering technique, *International Journal of Computer Science Issues*, 8,5 5-2, 185-194.

[43] Sales V., Terra R., Miranda L.F., and Valente M.T. 2013. Recommending move method refactorings using dependency sets, *IEEE 20th Working Conference on Reverse Engineering (WCRE),* 232-241.

[44] Seng O., Stammel J., and Burkhart D. 2006. Search-based determination of refactorings for improving the class structure of object-oriented systems, In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 1909-1916.

[45] Serban G. and Czibula I.G. 2007. Restructuring software systems using clustering, *22nd international symposium on Computer and information sciences*, 1-6.

[46] Tairas R. and Gray J. 2012. Increasing clone maintenance support by unifying clone detection and refactoring activities, *Information and Software Technology*, 54, 12, 1297-1307.

[47] Tourwé T. and Mens T. 2003. Identifying refactoring opportunities using logic meta programming, In: *Proceedings of the 7th European Conference on Software Maintenance and Reengineering*, 91-100.

[48] Tsantalis N. and Chatzigeorgiou A. 2009. Identification of Extract Method Refactoring Opportunities, In: *Proceedings of the 13th European Conference on Software Maintenance and Reengineering*, 119-128.

[49] Tsantalis N. and Chatzigeorgiou A. 2009. Identification of Move Method Refactoring Opportunities, *IEEE Transactions on Software Engineering*, 35, 3, 347-367.

[50] Tsantalis N. and Chatzigeorgiou A. 2010. Identification of refactoring opportunities introducing polymorphism, *Journal of Systems and Software*, 83 ,3, 391-404.

[51] Tsantalis N. and Chatzigeorgiou A. 2011. Identification of extract method refactoring opportunities for the decomposition of methods, *Journal of Systems and Software*, 84, 10, 1757-1782.

[52] Yang L., Liu H., and NiuZ. 2009. Identifying Fragments to be Extracted from Long Methods, In: *Proceedings of the 16th Asia-Pacific Software Engineering Conference*, 43-49.

[53] Zhao L. and Hayes J. 2006. Predicting classes in need of refactoring: an application of static metrics, In: *Proceedings of the 2nd International PROMISE Workshop*, Philadelphia, Pennsylvania USA.

[54] J. Al Dallal, Evaluating quality of primary studies on determining object-oriented code refactoring candidates, International Conference on Engineering & MIS 2015, Istanbul , Turkey, 2015.