I.J. Information Engineering and Electronic Business, 2016, 5, 44-51
Published Online September 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijieeb.2016.05.06

Proposal to Decrease Code Defects to Improve Software Quality

Ohood A. Aljohani, Rizwan J. Qureshi
Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
Email: ohoodcs@gmail.com, rmuhammd@kau.edu.sa

Abstract—Software quality is an important topic of software development and it is always challenging to deliver high-quality software. The major challenges, to complete the software, are time and cost without losing the software quality. Software quality has a significant impact on software performance. The acceptability, success, and failure of software are depending on its level of quality and number of defects. Software defects are one of the fundamental factors that can determine the time of software delivery. In addition**,** defects or errors need to be eliminated before software delivery. Software companies spend a lot to reduce code defects. The aim is to detect defects early with cheaper methods. This paper proposes a code quality scanner to decrease the code defects. The proposed solution is a combination of code scanner and code review. Moreover, the paper presents results using quantitative analysis to show the effectiveness of the proposed solution. The results are found encouraging.

Index Terms—Quality, Defects, Code Quality Scanner, Code Review.

I. Introduction

Developing high-quality software is a very challenging process. The challenges are, finishing the software with low cost and under certain time without losing software quality. Software success, acceptance, and failure are depending on its level of quality [1].Software quality is a very important topic for research within software engineering field. Recently, software plays a critical role in government, business, and institution also many other domains. Therefore, improving software productivity and quality are mandatory in software engineering. Software quality has a great impact on software performance. The quality for users is linked with performance and response time while quality for programmer could be error free and perfect program code [1]. Several stakeholders use software for various purposes while they assure different quality features depending on their customization. In contrast, companies emphasize product quality that meets customer satisfaction, improve business growth, and increase company profit [2]. Therefore, quality of the software is not only a question of how good is the software in terms of technical aspect, but also how much it meets the customers' requirements [3]. However during the development of any software, there are high probabilities to have errors in the various phases of software development lifecycle (SDLC) [2]. Defects or errors removal worthiness is a direct indicator of the ability of software development process in eliminating defects before software delivery [4]. Static analysis techniques deal with the representation of source for software, where the aim is to detect defects early.

Code review operation, when a programmer shows the code to one or more programmers, is one of the most common static analysis techniques. Code review is often a cost effective defect finding technique because it detects bugs in the early phase, also, it is a less expensive tool. Code reviews are consuming time, because of that the researchers investigated the factors influencing and benefits of code reviews [18].

The paper is organized as follows. Section 2 summarizes the related work. Section 3 defines the research problem. The proposed solution is presented in section 4. Section 5 validates the proposed solution, and section 6 concludes the paper.

II. Related Work

Software quality has been a substantial topic from the starting of software development [1]. It is discussed to explain that quality is a critical factor for any successful software. Nowadays, we are using a lot of software for almost every task in our daily life. Therefore, we need high-quality software that will satisfy the users' expectations. Lee [1] proposed a framework to solve customer value evaluation problem. The framework has a combination of software quality criteria. The framework is validated on only one software project it needs to be tested on different software to generalize the results. Lee et al. [2] explain a solution to validate and verify the software quality during software development process. The researchers use formal approaches to characterize the essential aspects of software. They discussed several metrics for each type of quality metrics which are in-process quality, product quality, maintenance quality, customer satisfaction quality, and product quality. Their discussed solution need to be extended on more than one software project to generalize the results.

The low-quality software is a problem that could cause software failure and waste both time and cost [3]. Mireles et al. [3] conduct a systematic mapping study for software

quality process. It is found that the main quality attributes are usability, security, and reliability. Mireles et al. [3] do not present the results regarding the software product quality.

Software quality is influenced not only by the users but also the software enterprises [4]. Measuring the quality is the critical point to develop high-quality software. Sun [4] explained how and when to use quality control and measurement, also the relation among each other. Sun [4] did not provide a specific measurement to show an example of how to measure quality.

Software quality and risk are two concepts appear to be coiled with each other [5]. The quality is important to reduce the risk. There are negative effects of people quality on software project risk probability as discussed by Sarigiannidis et al. [5]. The researchers did not test the validity of their results to have approved results.

To deliver high-quality software, software testing is mandatory [6]. However, software system testing is heavily obliged by time and budget. Lin et al. [6] combined two software specifications and testing methods, which are Markov chain statistical testing and sequence software specification to get a feasible and economical way to obtain high-quality software. The researchers method have been tested on a chosen application that is BlackBoard Quiz Editor.

Software quality attributes are estimated by using a suitable metrics [7]. However, choosing the right metrics is not always clear. Software Metric Fluctuation proposed by Arvanitou et al. [7] would quantify the degree of the metrics. Arvanitou et al. [7] classify the metrics as either sensitive or stable. Arvanitou et al. [7] did not cover a variety of metrics to examine the difference.

Recently, the maintenance cost has been raised, researches on software quality are becoming more substantial because having high-quality software led to maintainable software [8]. Tekin et al. [8] proposed visualization software quality tool based on object oriented and they call it E-Quality, which extracts metrics of the software quality automatically, relations of classes from the source code then displays them on a graph environment. The environment will efficiently refactoring and simplifies any complexity in a software system. E-Quality tool is applied only to java code.

Software defects are important factors that can determine the time of software delivery [9].Focusing on the number of defects and type of defects are a serious aspect that affects the software quality. The majority of software quality estimation approaches use software defects as an attributes to evaluate software quality. Using a clustered approach as proposed by Dhiman et al. [9] to classify software defects, the approach analyzes defects and its integration regarding software quality. The result of approaches is not compared with available approaches [9]. Singh et al. [10] discussed data mining methodologies that have been used to construct defect prediction model to use it for quality insurance. Singh et al. [10] did not discuss particular data mining methodology to understand the results. Code review is a cost effective defect finding technique because it detects bugs in an early phase, also it is a less expensive tool. Bacchelli et al. [11] discussed tool-based code review and the benefits of code reviews which are transferred team knowledge, improve team awareness, motivate interaction between team members, and facilitate finding alternative solutions to problems. Bacchelli et al. [11] made an interview and survey among diverse managers at Microsoft. It is found that code changes are the secret of code reviews while developers employ many tools to meet their needs. Kononenko et al. [12] investigated code review for a large open source project Mozilla. Kononenko et al. [12] discussed the relation between code reviewers' and set of factors that may affect the quality of review inspection. Kononenko et al. [12] applied SZZ algorithm to find bug changes then linked it with code review results from the tracking system. The paper found that 54% of code changes lead to bugs. Their paper presented that reviewers' experience, and a number of reviewers involved, are important factors for the quality of code review process. The code review is significantly affecting software quality; an empirical presentation done by McIntosh et al [13].The paper contribution confirms that poor code review had a negative effect on software quality. Beller et al. [14] found that the changes types because of modern code review for open source software are identical to those in the industry also to academic systems, featuring identical percentage of maintainability regarding functional issues. Code review evaluation study is done by Bosu et al. [15] using ReviewBoard in open source software showed that most of the review requests got a feedback within a day.

Improving the software quality by mapping the relational objects of software as explained by Muthukumar et al. [16], where the data of defects also number defects will be gathered and categorized. Muthukumar et al. [16] focused only on defects clusters.

The software integration is done during the software development lifecycle [17]. It has been confirmed that software integration and integration testing could have more than 40% of the project cost [17]. Therefore, it is critical to be done efficiently in order to manage the risks. A framework was presented by Hamdan et al. [17] which distinguishes quality features of the development process while applying the practices of continuous integration in the software projects development. The main focus was on continuous integration practices of agile software development [17].

A majority of the researchers deal with software quality models [18]. Each model must be analyzed before starting point. Vara et al. [18] discussed their experience in dealing with software model quality providing pieces of advice and their learned lessons. The experience of Vara et al. [18] is based on requirements model and conceptual model specification, where there are more models to test.

Project Management Methodologies could be used to achieve quality software [19]. PRINCE methodology has been discussed by Xu et al. [19] to test if PRINCE is sufficient to be used for quality. Xu et al. [19] found that

PRINCE cannot solve all software quality problems. The the paper did not test PRINCE in all software problems to generalize the results.

There are many metrics and measurements to examine software quality [20]. The quality factors of ISO/IEC 9126 model are discussed [20]. Challa et al. [20] used fuzzy approach to estimate the quality.

Table 1. Limitations of the papers comprising the literature review.

Title	Limitation
Software Quality Factors and Software Quality Metrics to Enhance Software Quality Assurance [1].	The paper test the solution on a single project, it is difficult to generalize the results.
Software Measurement and Software Metrics in Software Quality [2].	The paper validates the solution for only one software project; it is difficult to generalize the results.
Approaches to promoting product quality within software process improvement initiatives: A mapping study [3].	The paper does not provide results regarding software product quality that are addressed by software process improvement.
Knowledge for Software Quality Control and Measurement [4].	The paper does not provide a specific measurement to show an example of how to measure quality.
Quality vs risk: An investigation of their relationship in software development projects [5].	The paper does not test the validity of the results to have approved results.
Quality Assurance through Rigorous Software Specification and Testing: A Case Study [6].	The paper test only one chosen application (BlackBoard Quiz Editor), it is difficult to generalize the results.
Software metrics fluctuation: a property for assisting the metric selection process [7].	The paper does not cover different metrics to examine the difference.
E-Quality: A Graph Based Object Oriented Software Quality Visualization Tool [8].	The proposed tool extracts metrics and class relation only from java programming language.
A Clustered Approach to Analyze the Software Quality using Software Defects [9].	The paper does not provide the result of other used approaches to compare with.
Assuring Software Quality using Data Mining Methodology: A Literature Study [10].	A particular data mining methodology is not discussed to understand the results.
Enhancing Software Quality through Systematic Object Mapping [16].	The paper focuses only the defects clusters.
A quality framework for software continuous integration [17].	The paper focus only on the continuous integration of agile software development.
Dealing with Software Model Quality in Practice [18].	The paper experience is based on two models: requirements model and conceptual model specification there are more models to test.
Project Management Methodologies: Are They Sufficient To Develop Quality software [19].	The paper did not test PRINCE in all software problems to generalize the results.
Quantification of Software Quality Parameters using Fuzzy Multi Criteria Approach [20].	The paper did not consider more factors to quantify software quality to check each factor.
Expectations, Outcomes, and Challenges of Modern Code Review [11].	The paper generates the results using CodeFlow tool, result generalization is difficult.
Investigating Code Review Quality: Do People and Participation Matter? [12].	The paper did only a quantitative investigation of what factors may influence code review quality, difficult to generalize the results.
The Impact of Code Review Coverage and Code Review Participation on Software Quality [13].	The case study did on particular projects, result generalization is not sufficient.
Modern Code Reviews in Open-Source Projects: Which Problems Do They Fix? [14]	The study did on two OSS projects, generalizes the result is not sufficient.
Peer Code Review in Open Source Communities Using ReviewBoard [15].	The paper used code review for comments without the ability to change.

III. Problem Statement

The software companies want to develop software to be delivered with minimum defects. The quality of the code has a direct impact on the number of defects [18]. Code review tools have been proposed in order to reduce code defects. Bosu et al. [20] used a code review tool. However, simple changes in the code can be addressed by the reviewers. The objective of this step is to save the time of the developers. This paper attempts to examine a solution to reduce defects number to improve the quality of code by employing code quality scanner and code review that provides the changeability for reviewers.

IV. The Proposed Solution

The bugs that found in early program development are cheaper to fix. The software is written by human beings. Therefore, software often has some mistakes. In order to address this problem, a literature survey was performed to

extract the requirements from the analysis of the concepts: "software quality", "software code quality", and "software code review". The existing software code review offers great benefits, but it has not reached its full potential. The proposed solution is to use code quality scanner that has predefined rules. The developer will upload his/her code to the proposed scanner tool then the code will be checked and scanned against the predefined rules. Moreover, each organization has its own code quality rules. Therefore, programmer or manager has the ability to add more rules that are specific to the organization needs. As scanning finishes, an alert will be notified to the programmers if any section of the code has defects or mismatch the rules. The programmer needs to check his code and edit it to match organization rules. Then, a programmer could start the code review session. The code review provides the structure and the mechanisms for addressing code review tools problem. The programmers in organization upload the finished part of his code, each code section will be classified by privileges, and then other company programmer reviews it as soon as their workflow permits.

A. Start Code Quality Scanner

The programmer starts to post the finished code to be scanned in code quality scanner. The code will be scanned and checked against predefined rules.

B. Apply Predefined Rules

The programmer code will be scanned and checked against predefined rules. New rules could be added by the administrator of an organization.

C. Edit Scanned code

The programmer checks the code scanner mismatch then he changed the code to match rules.

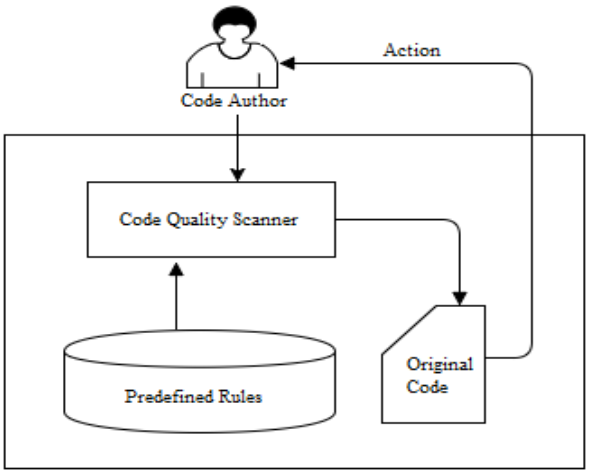

Fig.1. The Proposed Code Quality Scanner

D. Open a New Code Review Session

After code scanner finishes the programmer starts to post the finished code to be reviewed in code review tool. Reviewers will be notified to start the review session.

E. Define Code Privilege

The programmer gives his/her code specific privilege which is public or protected.

- Public code grants the ability to code reviewers to change in the code after the code author submits the code for the review session. Each reviewer can review the code and he can make changes directly or just add their comments.
- Protected code where reviewers can review the author code without ability to did any change in code, they just add their comments, after that code author will be notified about comments, code author have the decision either to follows their comments and updates the code or rejects change, because some change could enhance performance but violate software system in term of security.
- The code author accepts the reviewers' comments about changes and he made suggested changes in the code. The code review process is repeated again and again until it is approved by the reviewers.
- The changed code should be highlighted with a color that represents to the reviewer who has changed it.

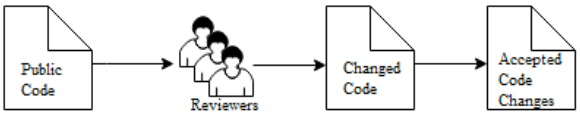

Fig.2. Public Code Review

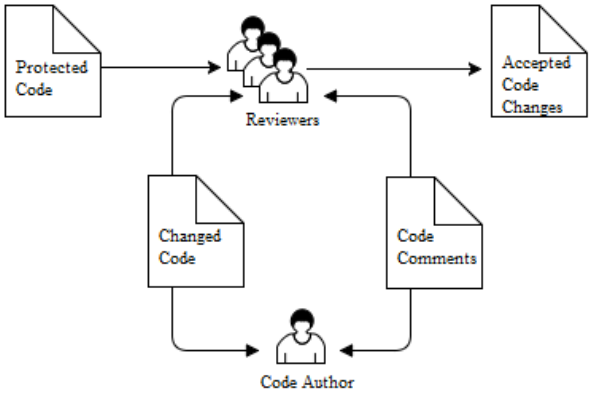

Fig.3. Protected Code Review

F. Reviewers Prioritization

To change the code, reviewers need to follow some rules where the priority is diverse among reviewers depending on their level of experience and history of their previous review. For example, code review tool gives reviewer one point for each acceptance code changes he/she made.

G. Code Review Database

After code review session termination all changes done will be saved in the database for further modification and

to save code in case of any failure. Code saving is done for code author account.

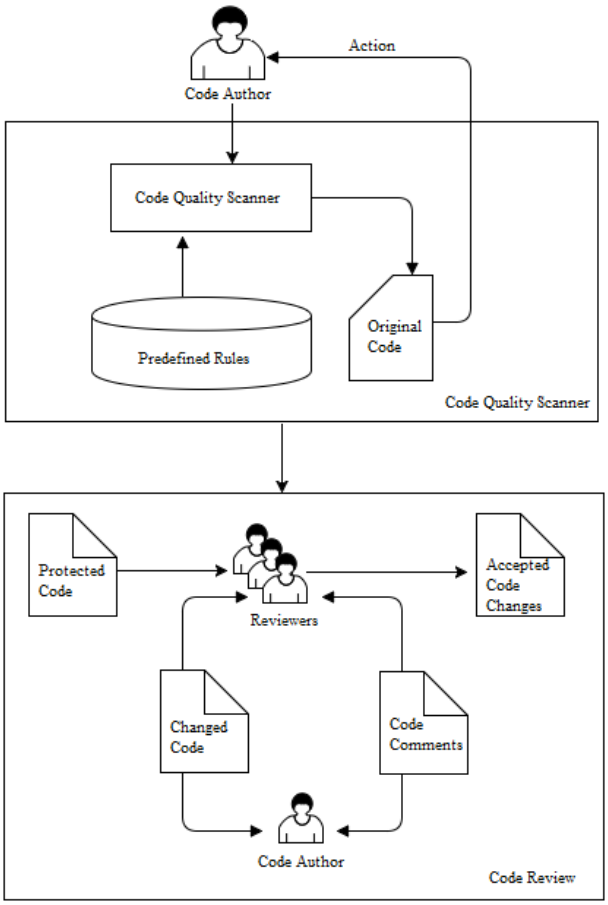

Fig.4. Integration of Code Scanner and Code review.

V. VALIDATION

In order to validate the proposed model, a closed-ended survey was distributed among IT students and staff containing 21 questions that covered many of the goals of the proposed solution. Five goals were used for data collection.

Goal 1 Suitability of the proposed solution for decreasing the number of code defects.
Goal 2 The flexibility to add new rules in code quality scanner.
Goal 3 The frequency of using code review in companies.
Goal 4 The effect of the used solution on cost saving.
Goal 5 The effect of saving programmer time after reviewers change feature.

The questions were answered using a 5-point Likert scale. The respondents were 63 IT staff and students. Google tools and Microsoft Excel were used for statistical analysis. The results are concluded mainly through frequency tables and bar charts.

A. Cumulative Statistical Analysis of Goal 1

The results are shown in Table 2.

Table 2. Cumulative Statistical Analysis of goal 1.

Q. No	Very Low	Low	Nominal	High	Very High
1	1	3	12	26	21
2	1	2	10	27	23
3	1	1	11	25	25
4	0	6	10	19	28
5	0	0	12	24	27
Total	3	12	55	121	124
Avg. %	1%	4%	17%	38%	39%

It is shown in Table 2 that 39% of the respondents believed the suitability of Goal 1, and 38% of the respondents are in favor of high. 17% of the professional remain neutral. 4% of the software engineers are supporting low and 1% of the participants are agreed to very low. Fig. 5 presents the aggregate analyses of goal 1.

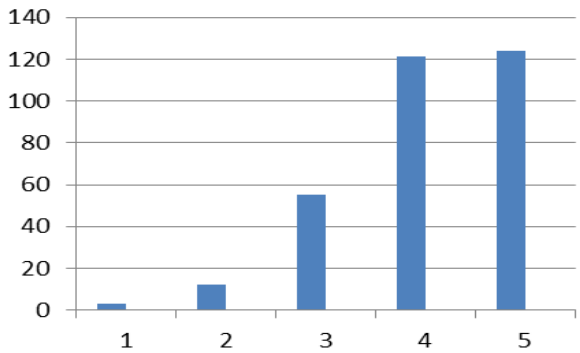

Fig.5. Aggregate analyses of goal 1.

B. Cumulative Statistical Analysis of Goal 2

Cumulative static analysis of goal 2 is shown in Table 3.

Table 3. Cumulative Statistical Analysis of goal 2.

Q. No	Very Low	Low	Nominal	High	Very High
1	0	4	7	25	27
2	2	4	16	21	20
3	1	5	16	25	16
4	2	3	8	23	27
Total	5	16	47	94	90
Avg. %	2%	6%	19%	37%	36%

Table 3 shows that 37% of the participants are agreed on this goal and 36% of the professionals are strongly agreed. Moreover, 6% of the respondents have disagreed and 2% strongly disagree while 19% of the professionals remain neutral. Fig. 6 displays the results of Table 3.

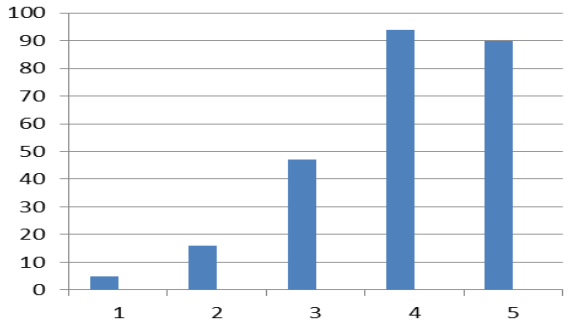

Fig.6. Aggregate analyses of goal 2.

C. Cumulative Statistical Analysis of Goal 3

Cumulative analysis of goal 3 is shown in Table 4.

Table 4. Cumulative Statistical Analysis of goal 3.

Q. No	Very Low	Low	Nominal	High	Very High
1	4	5	14	22	18
2	0	2	7	33	21
3	2	1	9	19	32
4	0	4	11	26	22
Total	6	12	41	100	93
Avg. %	2%	5%	16%	40%	37%

Table 4 shows that 40% of the participants are agreed on this goal and 37% the professionals are strongly agreed. Moreover, 5% of the participants disagree and 2% of the software engineers strongly disagree while 16% of the participants remain neutral. Fig. 7 presents the results of Table 4 graphically.

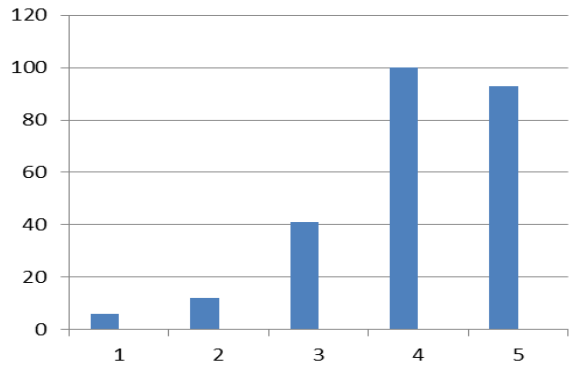

Fig.7. Aggregate analyses of goal 3.

Table 5. Cumulative Statistical Analysis of goal 4.

Q. No	Very Low	Low	Nominal	High	Very High
1	1	1	10	25	26
2	0	1	12	21	29
3	0	1	15	23	24
4	3	2	11	18	29
Total	4	5	48	87	108
Avg. %	2%	2%	19%	35%	43%

D. Cumulative Statistical Analysis of Goal 4

The cumulative analysis of goal 4 is shown in Table 5.

Table 5 shows that 35% of the participants are agreed to goal 4 and 43% of the participants are strongly agreed. 2% of the participants have disagreed and 2% of the professionals strongly disagree. 19% of the participants remain neutral. Fig. 8 shows this graphically as follows.

Figure 8 presents the aggregate analyses of goal 4.

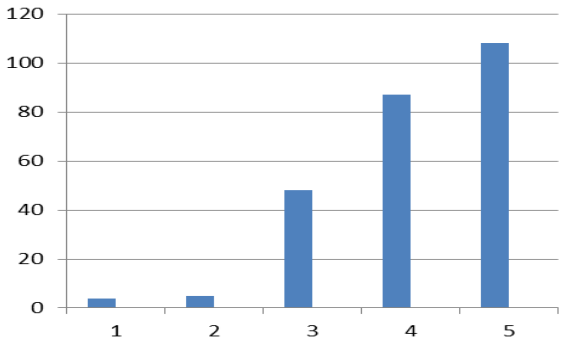

Fig.8. Aggregate analyses of goal 4.

E. Cumulative Statistical Analysis of Goal 5

The results of goal 5 are shown in Table 6.

Table 6. Cumulative Statistical Analysis of goal 5.

Q. No	Very Low	Low	Nominal	High	Very High
1	2	3	10	31	17
2	3	5	15	21	19
3	1	3	8	25	26
4	4	4	9	27	19
Total	10	15	42	104	81
Avg. %	4%	6%	17%	41%	32%

Table 6 shows that 41% of the participants are agreed to goal 4 and 32% of the participants are strongly agreed. 6% of the participants disagree and 4% of the professionals strongly disagree. 17% of the participants remain neutral. Fig. 9 shows this graphically as follows.

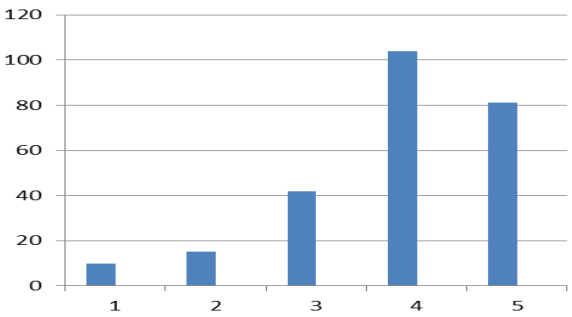

Fig.9. Aggregate analyses of goal 5.

F. The Final Cumulative Evaluation of all goals

Table 7 shows that only 2% of the software engineers report the very low effect of the proposed solution. 5% of

the professionals respond the low effect. 18% of the respondents report the nominal effect of the proposed solution. 38% of the professionals are highly favoring the proposed solution. Among the software professionals, 37% of the participants are very highly favoring the proposed solution. As such 75% support is available. Fig. 10 displays the results of Table 6.

Table 7. Cumulative Statistical Analysis of 5 goals.

All Goals	Very Low	Low	Neutral	High	Very High
Goal 1	1%	4%	17%	38%	39%
Goal 2	2%	6%	19%	37%	36%
Goal 3	2%	5%	16%	40%	37%
Goal 4	2%	2%	19%	35%	43%
Goal 5	4%	6%	17%	41%	32%
Total	11%	23%	88%	191%	187%
Avg. %	2%	5%	18%	38%	37%

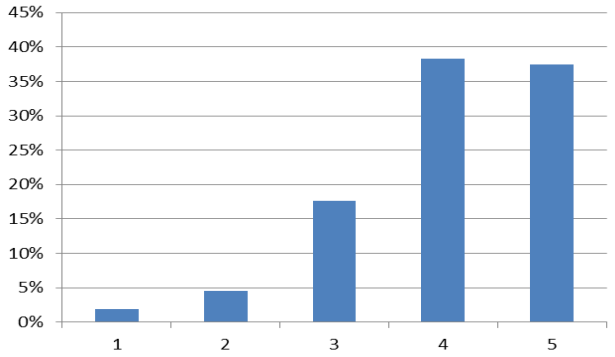

Fig.10. Aggregate analyses of all goals.

VI. Conclusion

The software industry faces significant challenges to achieving high quality. Even though, the software companies invest huge resources to reduce code defects. There are several automated tools available to reduce the testing efforts and improve the quality of software. However, it is important to review the code before performing automated testing to save time and cost. This paper presents the impact of code quality scanner with code review to reduce the number of defects. Moreover, the validation results show that 75% of respondents support the proposed code quality scanner with code review to improve and minimize the problems of code defects. It is anticipated that the proposed tool will help the software companies to identify the defects early before any further consequences.

Acknowledgment

I am thankful to Allah for allowing me to complete this paper. I am also grateful to my parents for supporting me. Special regards and thanks to my supervisor, Dr. R. J. Qureshi. He helped me to learn how to complete this paper.

References

[1] M. C. Lee, "Software Quality Factors and Software Quality Metrics to Enhance Software Quality Assurance," *British Journal of Applied Science & Technology*, 2014.

[2] M. C. Lee, and T. Chang, "Software Measurement and Software Metrics in Softwre Quality," *International Journal of Software Engineering and Its Applications*, vol. 7, July, 2013.

[3] A. Mireles, Á. M. Moraga, F. Garc á, and M. Piattini,"Approaches to promote product quality within software process improvemen tinitiatives: Amapping study," *The Journal of Systems and Software*, vol. 103, pp. 166, May 2015.

[4] Sun, "Knowledge for Software Quality Control and Measurement," *2011 International Conference on Business Computing and Global Informatization*, Shanghai, pp. 468-470, 2011.

[5] L. Sarigiannidis, and P. D. Chatzoglou, "Quality vs risk: An investigation of their relationship," *International Journal of Project Management*, vol. 1073, p. 32, 2014.

[6] L. Lina, J. Hea, Y. Zhanga, and F. Songb, "Quality Assurance through Rigorous Software Specification and," *International Conference on Soft Computing and Software Engineering*, vol. 1877, 2015.

[7] E. M. Arvanitou, A. Ampatzoglou, and P. Avgerioua, "Software metrics fluctuation: a property for assisting the metric selection process," *Information and Software Technology* , vol. 72, pp. 110-124, April 2016.

[8] U. Erdemir, U. Tekin, and F. Buzluca, "E-Quality: A Graph Based Object Oriented Software Quality Visualization Tool", *Visualizing Software for Understanding and Analysis (VISSOFT), 2011 6th IEEE International Workshop on*, Williamsburg, pp. 1-8, 2011.

[9] P. Dhiman, M., and R. Chawla, "A Clustered Approach to Analyze the Software Quality," *2012 Second International Conference on Advanced Computing & Communication Technologies*, pp. 36-40, 2012.

[10] Singh, and R. Singh, "Assuring Software Quality using Data Mining Methodology: A Literature Study," *Information Systems and Computer Networks (ISCON), 2013 International Conference on*, Mathura, pp. 108-113, 2013.

[11] Bacchelli, and C. Bird, "Expectations, Outcomes, and Challenges Of Modern Code Review," *International Conference on Software Engineering*, pp. 712-721, 2013.

[12] Kononenko, O. Baysal, L. Guerrouj, Y. Caoy, and M. W. Godfrey, "Investigating Code Review Quality: Do People and Participation Matter?," *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, Bremen, pp. 111-120, 2015.

[13] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, "The Impact of Code Review Coverage and Code Review Participation on Software Quality," ACM, pp. 192-201, 2014.

[14] M. Beller, A. Bacchelli, A. Zaidman, and E. Juergens, "Modern Code Reviews in Open-Source Projects: Which Problems Do They Fix?," Proceedings of the 11th Working Conference on Mining Software Repositories, *pp.* 202-211,2014.

[15] Bosu, and J. C. Carver, "Peer Code Review For Open Source Comunities Using ReviewBoard," Proceedings of the ACM 4th Annual Workshop on Evaluation and Usability of Programming Languages and Tools, pp. 17-24, 2012.

[16] R. Muthukumal, and D. Damodaran, "Enhancing Software Quality through Systematic Object Mapping," *IEEE*, 2015.

[17] S. Hamdan, and S. Alramouni, "A quality framework for software continuous integration," *International Conference on Applied Human Factors and Ergonomics*, vol. 3, pp. 2019-2025, 2015.
[18] Vara, and H. Espinoza, "Dealing with Software Model Quality in Practice," *International Conference on Quality Software*, pp. 396-405, 2013.
[19] S. Xu, and D. Xu, "Project Management Methodologies:Are They Sufficient To Develop Quality Software," pp. 175-178, *IEEE*, 2011.
[20] S. Challa, A. Paul, Y. Dada, V. Nerella, and P.R. Srivastava, "Quantification of Software Quality Parameters using Fuzzy Multi Criteria Approach," *IEEE*, pp. 1-6 , 2011.

Authors' Profiles

Ohood A. Aljohani is a post-graduate student of Information Technology in King Abdul-Aziz University. She received her Bachelor in Information Technology form King Abdulaziz University, Jeddah, Saudi Arabia.

Dr. Rizwan J. Qureshi received his Ph.D. degree from National College of Business Administration & Economics, Pakistan 2009. He is currently working as an Associate Professor in the Department of IT, King Abdulaziz University, Jeddah, Saudi Arabia. This author is the best researcher awardees from the Department of Information Technology, King Abdulaziz University in 2013 and 2016. He is also honoured as the best researcher from the Department of Computer Science, COMSATS Institute of Information Technology, Pakistan in 2008.

How to cite this paper: Ohood A. Aljohani, Rizwan J. Qureshi,"Proposal to Decrease Code Defects to Improve Software Quality", International Journal of Information Engineering and Electronic Business(IJIEEB), Vol.8, No.5, pp.44-51, 2016. DOI: 10.5815/ijieeb.2016.05.06

