Detection of Defect Potentials using Peer Reviews: An Agile Approach

Chandravardhan Singh Raghaw
Dept of Information Technology
Govt. Engg. College Ajmer
Ajmer, India

Prakriti Trivedi
Dept of Computer Science & Information Technology
Govt. Engg. College Ajmer
Ajmer, India

Vipul Sharma
Dept of Computer Science & Information Technology
Govt. Engg. College Ajmer
Ajmer, India

ABSTRACT

Maintain quality is always an important issue for the software end product. This paper includes how to improve quality of product by detecting possible number of defects potentials. The authors are using an approach called Peer Review for detection of defect potential. In this paper peer review is done using an agile approach. A study has been conducted on three final year projects and includes the detection of maximum number of defect potentials using different peer review techniques. Finally, as a result, total numbers of defects with different review techniques are compared with possible defects.

General Terms

Software Quality, Defects Removal.

Keywords

Defect Potential, Peer Review, Agile Approach.

1. INTRODUCTION

The main goal of the software industry is to achieve a quality product. The term quality refers to complex mix of factors that will vary across different applications and the customers who request them [1]. Software Quality are mainly affected by two factors which are broadly classified into following two subgroups: (1) factors that can be directly measured (e.g., defects per function-point) and (2) factors that can be measured only indirectly (e.g., usability or maintainability) [2]. The goal can be achieved only if number of defects can be properly removed and defects can removed if they are detected first.

1.1 Defect Potentials

The term defect potential simply means the approximate numbers of defects that will be found during the development of software applications [3]. Defect potentials should be measured with function points (FP) and not with lines of code (LOC). This is because most of the serious defects are not found in the coding phase, but rather in requirements and design phase. Defect potentials correlate with application size. As application sizes increase, defect potentials also rise. The relationship between defect potential and defect size simply uses the thumb rule: the number of maximum defect potentials is equal to the number of Function Points [4].

The relationship between the number of defect potential and the number of function points is:

$$\boldsymbol{Defect\ Potentials} = \mathbf{1.2} \times \boldsymbol{Function\ Point}$$

1.2 Agile Method

An agile approach follows rapid delivery of high-quality software. Many associations accept Agile Methods (AMs) as a main development methodology that advances the qualitative products in software industry. This process saves majority of time. The main target of the agile method is to minimize risk and defect by developing the software product in small time boxes, known as iterations, which typically last for short duration of time and depends on the project as shown below in Figure 2. Each iteration for software development admits the release of new functionality with each small increment. Agile method directly aims to find something better at the final phase of each iteration.

Agile methods support an iterative approach for software product development [5]. For quick changes in the product agile practices are the most suitable technique. This method focuses of quick product delivery for those customers, in which requirements can be changed or new requirements are faced in later iterations of the software system [6].

1.3 Peer Reviews

The significant step for engineering projects and system during its implementation and design phase is Peer Review. This modern technique shows about the design evaluation concepts, documentation, and various primary concepts to maintain the quality standards of a project as well to reach central objective. Independent groups as well as individual which are not tying up with the original team of design are generally carried peer reviews. Peer review is terminology related to software engineering that stands for a type of review where work product is reviewed so as to evaluate its technological information along with its quality [8]. Its main objective is actually to deliver a new way for detecting and fixing flaws in designed software items to prevent their persistence directly into operational usage of the product [9].

It has been found that peer review is the one of the best efficient way to enhance the quality as well as productivity of various design procedures not just in software engineering but additionally in various engineering disciplines which includes mechanical [13], civil [11][12], electrical [10], and engineering of fire protection [14]. Peer reviews also support the detection of all probable defects as much as possible. Many experts of various software industries have outlined this

peer review technique as one of the best software development practice which was purely based on demonstrated value [15], [16].

Furthermore, peer reviews for a particularly project may possibly uncover almost all probable number of defects that generally found at the beginning period of software development and also it would be effective in terms of monetary value. Peer review is 10 to 100 times less expensive to resolve the makeover of any system carried at stage of system testing.

In this experiment, the author applied peer reviews in an agile way during the development of final year projects of a Rajasthan Technical University. The results from the instructor and the students' in terms of defects with peer reviews are reported in this paper. The author believes that the recommendations derived from the data analysis are of general interest, and it is hoped the findings will encourage more use of peer reviews in other similar fields too; after all, engineering is by nature a peer-review-based discipline.

The rest of this article is structured as follows. The experiment setup details (including the hypothesis, course project description, people involved, and the peer-review process) are explained in Section 2. The outcomes of the experiment are reported and analyzed in Section 3. Conclusions and future work are discussed in Section 4.

2. RESEARCH APPROACH

In this paper author finds the total number of defects using function point method. During the calculation of Function Point the average Weighing factor is used. Different independent testing teams' applied peer reviews on three sample projects and calculated the total number of defects. As a result total numbers of defects are compared with the defects found using different peer review techniques by separate independent teams i.e., an agile approach and finally efficiency for different peer review techniques are analyzed.

2.1 Function Point Calculation

The Table 1 shows that how to calculate function point (FP). This particular table consists of five major information domain characteristics and the corresponding counts are presented inside the appropriate table column [17].

- ***User inputs:*** It shows the total user input which provides specific application-oriented data. Inputs ought to be distinguished from inquiries, which might be counted independently.

- ***User outputs***: It shows the output which provides specific application-oriented information. This output cites to screens, error messages, reports, etc. Specific informative item in the report are certainly not counted individually.

- ***User inquiries***: It's usually an on-line input in which final results are generated with a contiguous software response and it would be in the form of an on-line output and counts each different inquiry made by user.

- ***File Numbers***: It consists of sum of numerous master files (i.e., a group of logical data that would be a part of an individual file or large database) which is to be counted.

- ***External interfaces***: It consists of almost all interfaces that are already counted (e.g., data files on storage media) and used for transmission of information to other systems.

The following relationship is used for computation of function points (FP) [18].

$$\boldsymbol{FP = count\ total \times [0.65 + 0.01 \times \sum(F_i)]}$$

Where count total represents the cumulative sum of all the function point values as well as represents the cumulative sum of all the function point values taken from Table 1.

The Fi (i=1 to 14) are "complexity adjustment values" based on responses to the following questions [19].

1. Does the actual system need recovery and backup?

2. Data communications are required?

3. Distributed processing functions are there?

4. Whether performance is critical?

5. Does the system operate in heavily utilized environment?

6. Is there any system requirement for on-line data entry?

7. Will on-line data entry require input transactions in order to build over multiple operations or screens?

8. Does master files regularly updated on-line?

9. Does inquiries, outputs, files, or inputs complex?

10 Does internal processing is complex?

11. Does the program coding support reusability?

12. Does the design part include installation and conversion?

13 .Does systems support multiple installations for various organizations?

14. What are the procedures that are required for application designing that perform change and ease of work?

2.2 Peer Review Technique

Peer reviews can take many methods. Some of them are inspection, pair-programming and walkthrough. An inspection adopts methodical technique as well as it is one of a in-depth kind of peer reviews. It is a multistage process that involves assignment of specific roles to particular individuals. As compare to informal reviews, inspections are much efficient in finding defects [8]. Pair Programming means working concurrently at same workstation on the similar program and reviewing work constantly. It is not feasible in terms of viewpoint when an individual who is not attached with the code that brings a formal review. Walkthrough is not a formal review. It is not formal as it doesn't follow a pre-defined process as well as it doesn't specify any exit criteria and make no metrics. Inspections and walkthrough differ from each other mainly because the author's role is dominant when compared with other team members.

2.3 Efficiency Calculation

Total efficiency (η) can be derived using below formula:

$$\boldsymbol{\eta = \frac{Defects\ in\ Peer\ Review\ Technique}{Total\ number\ of\ Defects}}$$

The above formula calculates the efficiency of the peer-review techniques.

3. EXPERIMENTAL RESULTS

Using the above formulas the author is trying to calculate the total number of defects using function point method. The

statics for the Project 1, Project 2 and Project 3 are given below in the following Table 2, Table 3 and Table 4.

After detection of maximum probable defects using function point method, the author then finds the maximum possible number of defects using peer reviews technique with the help of different independent testing teams and compares it with defects calculated by function point and calculates the efficiency of different peer review technique. The statics of defects found during peer review technique are mentioned in Table 5 below.

After finding the probable number of defects by different independent testing teams which are involved in peer review. The efficiency is calculated which is observed from the various peer review techniques. The efficiency values are shown in Table 6 below.

The plotted graph for efficiency versus different peer review techniques are shown in Figure 2. The below figure depicts various efficiency level for the peer review techniques. The efficiency calculations are performed on different final year projects.

4. FUTURE WORK AND CONCLUSION

Detection of Defect Potentials using Peer Reviews in Agile Approach has been studied. Results have already been shown in the above paragraph. Further we can extend this work for study of effect of refactoring on post delivery maintenance of software development using agile approach.

ACKNOWLWDGEMENT

The author is grateful to the final year students and also their respected instructor of the project for their enthusiasm and collaboration in this experiment. The article author also appreciates for useful remarks, comments and ideas to anonymous reviewers that make this article much better.

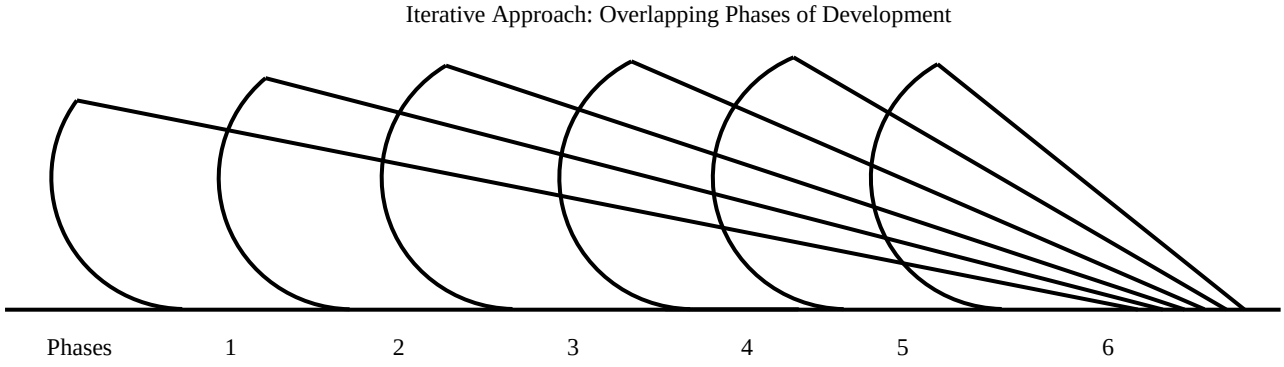

Fig 1: Agile Approach

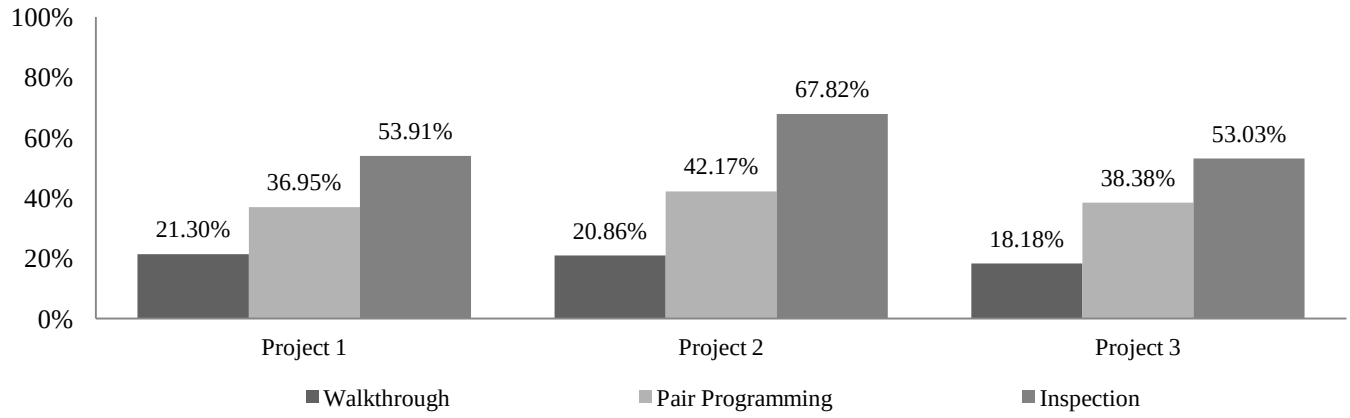

Figure 2: Efficiency for Peer Review Techniques

Table 1: Function Point Calculation

Measurement parameter	Count		Weighting factor			
			Simple	Average	Complex	
User inputs		×	3	4	6 =	
User outputs		×	4	5	7 =	
User inquiries		×	3	4	6 =	
File Numbers		×	7	10	15 =	
External interfaces		×	5	7	10 =	
Count total →						

Table 2: Project 1 Statics

Total Count	=	230
Complexity Adjustment Values(F_i)	=	1.07
Function Point	=	246
Maximum Probable Defects	=	295

Table 3: Project 2 Statics

Total Count	=	291
Complexity Adjustment Values(F_i)	=	1.07
Function Point	=	311
Maximum Probable Defects	=	374

Table 4: Project 3 Statics

Total Count	=	309
Complexity Adjustment Values(F_i)	=	1.07
Function Point	=	311
Maximum Probable Defects	=	374

Table 5: Statics using Peer Review

Total Count	=	309
Complexity Adjustment Values(F_i)	=	1.07
Function Point	=	311
Maximum Probable Defects	=	374

Table 6: Statics for Efficiency

Peer Review Technique	Project 1 (Efficiency)	Project 2 (Efficiency)	Project 3 (Efficiency)
Walkthrough	21.30%	20.86%	18.18%
Pair Programming	36.95%	42.17%	38.38%
Inspection	53.91%	67.82%	53.03%

REFERENCES

[1] Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 5th ed. New York, U.S.A.: McGraw-Hill, 2001, ch. 19, pp. 508.

[2] McCall, J.P. Richards and G. Walters, "Factors in Software Quality," three volumes, NTIS AD-A049-014, 015, 055, November 1977.

[3] Capers Jones, "Measuring Defect Potentials and Defect Removal Efficiency", *The Journal of Defense Software Engineering*, June 2008

[4] David Longstreet, "Fundamentals of Function Point Analysis". Available: http://www.softwaremetrics.com/Articles/defects.htm.

[5] Ms Ramandeep Kaur, Mrs Manmohan Choudhary, Mr Rahul Mehta, "Agile Process: An Enhancement to The Process Of Software Development", *IJCSNS International Journal of Computer Science and Network Security*, vol. 12, No.7, July 2012

[6] Ian Sommerville, *Software Engineering*, 9th ed. Boston, U.S.A.: Pearson, 2011, ch. 3, pp. 59.

[7] Vahid Garousi, "Applying Peer Reviews in Software Engineering Education: An Experiment and Lessons Learned", *IEEE Transactions on Education*, vol. 53, No. 2, May 2010

[8] Karl E. Wiegers, Peer Reviews in Software: A Practical Guide Reading, Addison-Wesley, 2002.

[9] M. B. Chrissis, M. Konrad, and S. Shrum, *CMMI: Guideline for Process Integration and Product Improvement.* Reading, MA: Addison-Wesley, 2003.

[10] "Training course: Electrical engineering for buildings, industry & utilities," Ballengearry Consulting, Last accessed Apr. 2008. Available: http://ballengearry.com.au/ee01.html.

[11] A. E. Gallegos, "Environmental restoration project quality procedure for design review," U.S. Dept. of Energy, Document Catalog #ER2002-0352, 2002.

[12] *Use of Peer Reviews for Projects*, Policy #117, American Society of Civil Engineers, Last accessed Aug. 2008. Available: http://www.asce.org/pressroom/news/policy_details.cfm?hdlid=117.

[13] "Engineering peer reviews (Revision: 1.0)," NASA Goddard Space Flight Center, Reference GPG 8700.6, 2005.

[14] "Guidelines for peer review in the fire protection design process," Society of Fire Protection Engineers, Last accessed Aug. 2008. Available: http://www.sfpe.org/upload/peer_review_guidelines.pdf.

[15] B. Boehm and V. R. Basili, "Software defect reduction top 10 list," *IEEE Comput.*, vol. 34, no. 1, pp. 135–137, Jan. 2001.

[16] R. L. Glass, "Inspections-Some interesting findings," *Commun. ACM*, vol. 42, no. 4, pp. 17–19, 1999.

[17] Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 5th ed. New York, U.S.A.: McGraw-Hill, 2001, ch. 4, pp. 89.

[18] Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 5th ed. New York, U.S.A.: McGraw-Hill, 2001, ch. 4, pp. 90.

[19] Arthur, L.J., Measuring Programmer Productivity and Software Quality, Wiley-Interscience, 1985.