Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 153 (2019) 45–54

Procedia Computer Science

www.elsevier.com/locate/procedia

17th Annual Conference on Systems Engineering Research (CSER)

Does every formal peer review really need to take place? An industrial case study

Keith Roseberry[a,*], Mary Ann Sheppard[b], Robert Wallis[c], Ye Yang[d]

[a]Collins Aerospace, Raleigh, NC 27604 USA
*[b]NAVAIR**, Lakehurst, NJ 08733 USA*
[c]Leidos, Gaithersburg, MD 20879 USA
[d]Stevens Institute of Technology, Hoboken, NJ 07030 USA

Abstract

Existing studies on formal peer reviews (FPR) are mostly based on design and code review practices observed from the open source software (OSS) community, and there is a lack of studies and evidence from proprietary, large scale software development in systems engineering settings. This study fills the gap by reporting an industrial case study on FPR practices in a safety-critical embedded system development organization. First, we follow the Goal-Question-Metric (GQM) methodology to establish a set of metrics for understanding FPR practices and the resulting effectiveness. Second, we collect the metric data from 100 programs employing FPR practices on a variety of document types. Third, we analyze the data and derive answers to a set of questions defined in the GQM. The study contributes to more comprehensive understanding of FPR practices and effectiveness evaluation, and the findings reported here may serve as a basis for process improvements initiatives in similar settings.

Keywords: Formal Peer Review, Inspections, Review Effectiveness, Defect, Defect Density, GQM

* Corresponding author. Tel.: +1-252-293-7530;
E-mail address: keith.roseberry@collins.com;
** The views expressed in this paper do not necessarily represent the views of NAVAIR or the United States.

This document does not contain export controlled technical data.

10.1016/j.procs.2019.05.054

1. Introduction

A Formal Peer Review (FPR) can be defined as "an activity in which one or more persons other than the author of a work product examine that product with the intent of finding defects and improvement opportunities" [5]. FPRs are performed primarily to verify and improve the quality of the artifacts produced during the product development life cycle. Consequently, a variety of different FPRs (also known as Inspections), which are typically organized as collaborative meetings among a group of pre-identified peers with the goal of identifying defects in corresponding intermediate and/or end products [6], are conducted at the conclusion of each phase of the life cycle. Besides the main focus of defect discovery, FPRs can also help the reviewers gain understanding about the system and aim to improve the maintainability of code for future changes [16].

The benefits of FPRs are well-accepted; however, the practices by which FPRs are conducted vary dramatically depending upon different development methodologies, domains, cultures and project constraints. Existing literature has raised various concerns related to FPRs such as high cost, ineffectiveness and inefficiencies. For example, in the Requirements phase of the life cycle, difficulties such as the inability to identify a defect in requirements, unrealistic requirements, lack of review team commitment, team conflict, and lack of support tools were cited [13]. Meeting-based FPRs were found to be much costlier but did not find more defects [14], as compared with non-meeting-based FPRs. In some cases, FPRs can be cumbersome and time-consuming thus hindering the successful adoption of this practice [15].

Since existing studies are mostly based on design and code review practices observed from the open source community, there is a lack of studies and evidence on broader types of FPR from proprietary, large scale, safety-critical development settings. Such industrial projects are frequently under conflicting constraints of satisfying strict process and/or domain compliance requirements as well as trying to improve process effectiveness. Continually improving the quality of systems in an efficient manner led us to the evaluation of FPR effectiveness. The genesis of this paper was driven by the literature gap and increasing practical needs for effective means to determine whether employing full, intensive FPRs justifies the cost of performing the reviews based on the number of defects identified during the review. More specifically, this paper proposes a set of metrics for evaluating FPR effectiveness, and demonstrating its application through an industrial case study using FPR data from a safety-critical embedded development organization. While the study analyzes FPR data from software development projects, the FPR methods described and the conclusions reached are generic and apply to non-software disciplines. Additionally, the FPR practices employed by the organization from which the data was gathered are done so at the system level as well as the software and firmware levels.

The main results in this study include: 1) The metrics gathered exhibited little correlation between labor hours and the number of defects found. 2) FPRs were much more effective when performed early in the life cycle. 3) FPRs performed during the Test phase and after became increasingly less effective. 4) Some of the data showed boundaries in measures that can be exploited to maximize the efficiency of defect detection. The study contributes to more comprehensive understanding of FPR practices and effectiveness evaluation, and the findings and models reported here may serve as a basis for process improvements for projects/programs in similar settings.

The rest of the paper is organized as follows: Section 2 presents the related work; Section 3 introduces the proposed GQM model for measuring FPR effectiveness; Section 4 reports the case study analysis and results; Section 5 is the discussion; finally, Section 6 is the conclusion.

2. Related Work

Fagan's seminal work in 1976 defined a formal, systematic design and code inspection process to reduce errors in software development [7]. Formal inspection, or FPR, has been adopted in many process models [6][18] and international/industry standards [12][19] as an important technique for improving both product quality and process productivity.

While effective in identifying defects, a FPR in its traditional form is typically labor intensive and prevents wide adoption and efficiency with respect to drastically different project settings [8]. With the prevalent adoption of open source development and agile methodologies, there is increasing need for light weight, less-rigid FPR practices. Rigby and Bird [9] presented an exploration of 6 aspects of contemporary peer reviews in software projects that span varying domains, organizations, and development processes, and found that while there were some minor divergences in contemporary practice "their similarities outweighed their differences." Though their findings were derived using three projects representative of development at Microsoft, AMD, and two Google-led OSS projects, the reviews are all at code level, and most are OSS projects. This study considers a variety of review types and represents practices from more complex embedded system domains.

In the embedded system domain, a recent survey conducted by Barr Group [10] reported concerning trends around best practices for embedded system development: "Of over 1,700 qualified respondents involved in the design of embedded devices that could kill or injure if the device malfunctioned – so-called safety-critical devices", it is found that 41 percent are not performing comprehensive code reviews. Possible reasons might include the development teams are under tight schedule pressure or budget constraint, however, it also indicates that there is lack of methods and data to support better decision making when considering how much FPR is needed. The lack of using FPRs due to these obstacles is also a key motivator for the study reported here.

3. Model for measuring FPR effectiveness

3.1. Working definitions

For the purpose of this study, we provide two working definitions for defect and formal peer review, respectively.

Defect: An imperfection or deficiency in a work product where that work product does not meet its requirements or specifications and needs to be either repaired or replaced [11]. There are varying severities of defects ranging from Trivial to Critical [2] and Inconsequential to Blocking [11]. In the scope of this paper, we exclude defects classified as Trivial and Inconsequential, and focus on non-trivial defects only, which is believed to be a better representation for true FPR effectiveness relevant to the study context.

Formal Peer Review (FPR): An effective FPR is one that identifies defects that require correction prior to the next phase of the life cycle. Defects that are deferred and low-severity defects (typographic errors, simple mistakes) do not substantially contribute to the effectiveness of a FPR.

3.2. Applying Goal-Question-Metric (GQM) to establish the measurement model

The Goal-Question-Metric (GQM) [1] method was used to evaluate the FPR data to determine whether FPRs are an effective activity in identifying defects. The GQM approach uses an identified project/process goal to drive questions that characterize the achievement of the goal that when answered result in a set of metrics. This section describes the use of the GQM approach to create the metrics needed to answer the question of FPR effectiveness.

The goal of the study of FPR is to evaluate the effectiveness of FPRs at identifying defects. To achieve this goal, the following questions are formulated for exploration in this study:

- Q1: What is the cost of a FPR?
- Q2: What is the effectiveness of a FPR?
- Q3: Is there any relationship between FPR effectiveness and the review type?
- Q4: Is there any relationship between FPR effectiveness and the review size?

The cost of a FPR is characterized by the effort expended to complete the review. This is most obviously stated in terms of labor hours. Review size can be quantified by counting the elements of the review. Effectiveness is influenced by the type of the review, since defects identified in earlier phases are considered more cost-beneficial and therefore the FPR is more effective.

The following metrics were collected:

- M1: Review Type (Requirements Review, Design Review, Code Review, etc.)
- M2: Number of Defects identified during a FPR
- M3: Number of labor hours expended during a FPR
- M4: Size of the FPR.

3.3. M1 factor

The M1 Factor is proposed to model the relative FPR effectiveness based on relative cost for fixing defects discovered across different life cycle phases. It is slightly adapted from the traditional cost association to defect resolution [3]. The basic rationale is that: since the defects found in phases before Code Review are not defects in the delivered end-product, but are defects associated with the artifacts produced within that phase, the cost to resolve these defects is much lower than if the defects were in the actual product.

As summarized in Table 1, the M1 Factor is then adjusted for the other phases of the life cycle by inferring a relative cost of a defect in the data produced in a phase to the cost of correcting the defect in the product if the defect were allowed to propagate into the product. For example, finding a defect in a requirement is highly valuable since if the defect propagated into the code the cost of finding the defect would be much higher. Conversely, finding a defect in the test data is less valuable since the product is already built and the defect in the test data only represents an inability to detect a defect in the product.

Table 1. Review types and factors *(Note: The lowest software factor in the range is used in the analysis).*

Phase	Review type	Cost factor	M1 factor
Requirements	Requirements	1X	10
Design	Design	5X – 7X	2
Build	Code	10X – 26X	1
Test	Test	50X – 177X	0.237
N/A	Document	N/A	0.2
N/A	Configuration Accounting	N/A	0.1

In Dabney et al. [3], the authors conclude that the cost factors to resolve defects in the phases of the life cycle are exponential in nature. In order to derive the corresponding M1 Factor for different review types, we first aligned the Cost Factor with the Coding (Build) phase, where finding a defect in the coding phase equates to finding a defect in the product. Therefore, the Coding (Build) phase cost factor is 1X and the remaining factors adjusted accordingly. In [4] the defect density for testing was measured at 4.5 defects/KLOC, whereas the defect density for construction was 19 defects/KLOC. For this organization, which is similar to the organization from which the FPR data was harvested, the testing defect density indicates that 23.7% of the defects through the coding phase remain in the product during the testing phase and are available to be detected by test activities. Therefore, we set the M1 Factor to 0.237 for testing. We estimated he Configuration Accounting Reviews and Document Reviews factors to be 0.1 and 0.2 because their reviews only cover defects in data associated with the product, not the product itself. We consider requirements, design and code reviews that uncover defects much more effective at ensuring production quality than other reviews.

3.4. FPR effectiveness model

Effectiveness of a FPR can be characterized by the number of identified defects related to the number of hours expended during the FPR, adjusted by the weighting factor determined by the type of the review (M1 Factor). For example, a 10-hour FPR that identifies several defects can be considered ineffective in relation to a 4-hour FPR that identifies a dozen defects. Therefore, the metric to be collected and analyzed is:

$$\text{FPR\_Eff} = \left(\frac{Defects}{Labor\ Hours}\right) \times \text{Review Type Factor} \tag{1}$$

4. Industrial Case Study

4.1. Company Background

The organization from which the review data was harvested develops safety-critical embedded systems in the commercial aviation domain, by following a well-defined life cycle that is compliant to the objectives of ARP4754A, DO-178C, and DO-254. The projects from which the FPR data was gathered comprised Design Assurance Levels A and B of DO-178C, which need to satisfy a higher number of objectives than lower assurance levels. Additionally, the satisfaction of almost half of these objectives must be with independence, for which the review of the data must be performed by someone other than the originator of the data [12].

The company's life cycle process includes a FPR activity at the end of each phase. The FPR is structured to ensure all products are comprehensively reviewed by independent persons and all identified defects are addressed before the product under review is accepted. The review team size is typically 2 to 5 persons with little variation in the membership during a project. The fundamental steps of the FPR process include:

1. Organize the FPR. A review leader and reviewers are established. Goals for the FPR are identified and the product(s) under review are placed in a configuration management library.
2. Review the Product(s). The reviewers inspect the product(s) with the guidance of one or more checklists and identify defects.
3. Meeting (Optional). The reviewers meet and discuss their findings to reach consensus on the outcome of the FPR.
4. Resolve the Defects. The originator(s) of the product(s) make changes to the product(s) to resolve the defects identified by the reviewers. All changes are independently verified.
5. Conclude the FPR. The review leader completes the FPR record and documents the reviewers' conclusion and the total labor hours of all reviewers. If the result is "not accepted", the product(s) are reworked and a subsequent FPR is conducted. If the result is "accepted", work continues with the next phase of the life cycle.

The life cycle process is defined to promote incremental development of small sets of functionalities to enable rapid delivery to internal stakeholders (integrators, testers, etc.) and customer facilities (integration laboratories, system test environments, etc.). While not strictly an Agile-compliant process, the development approach adopts certain agile methods to accommodate prototyping, incorporation of late changing requirements, and adaptation of functionality through experimentation.

4.2. Data collection and cleansing

The following information was gathered:

- Requirements for Requirements Reviews
- Pages for Design, Document, and Configuration Accounting Reviews
- Software Modules for Code Reviews
- Test Cases and Test Procedures for Test Reviews
- Labor hours from the reviews. The labor hours value is a total for all reviewers.
- The number of defects identified during the review, excluding the Trivial or Inconsequential defects.

The full data includes a total of 100 reviews were harvested from the organization, which has had a FPR process in place for over 15 years. The breakdown of the FPRs includes the following 6 datasets categorized by review type.

- 18 Requirements Reviews
- 21 Design Reviews
- 14 Code Reviews
- 16 Test Reviews
- 11 Document Reviews

- 20 Configuration Accounting Reviews

The statistics of the full and reduced data sets are shown in Table 2.

Table 2: Data set statistics

	Full data set statistics					Reduced data set statistics				
Measure	Min	Max	Mean	Median	Mode	Min	Max	Mean	Median	Mode
Defects	0	62	4.92	1	0	1	62	6.7636	3	1
Labor Hours	0.2	108	6.61	3.13	1.5	0.5	108	7.49	4.8	1.5
Review Size	1	824	74.2	18.5	2	1	824	106.036	25	44

Examining the mean and median of the measures shows that the mean is affected by the large variation in the data and is significantly different than the median. However, the data does not follow a normal distribution; therefore, the mean is not applicable for our analysis and does not indicate the central tendency of the data. The median is a much better indicator of central tendency and we see that the median is less than the mean for all measures, indicating that the distribution is skewed to the left (lower values). This observation is reinforced by the mode values being smaller than the median, indicating the most commonly encountered values are to the left of center.

The data set was reduced to eliminate those reviews that resulted in finding no defects. This reduced the data set by 45 reviews. We see some interesting similarities and obvious differences. First, the mean has moved notably to the right for nearly all measures. However, the Review Size measure's mean value has moved the furthest. Likewise, the median has moved to the right. Finally, the mode value for the Review Size measure is the only one that is higher than the median. The distribution is still skewed significantly to the left.

4.3. Analysis and results

This section presents the analysis and answers to the questions defined in the GQM. In the following charts, the numbers along the x-axis identify different review types, for which the mapping is defined as: 1) Code Review; 2) Configuration Accounting Review; 3) Design Review; 4) Document Review; 5) Requirements Review and 6) Test Review.

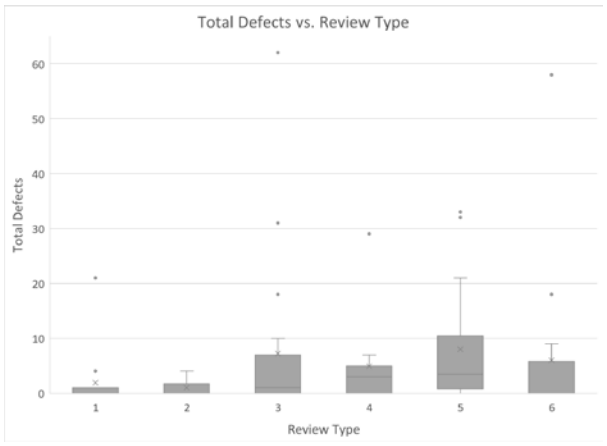

Figure 1: Defects versus review type

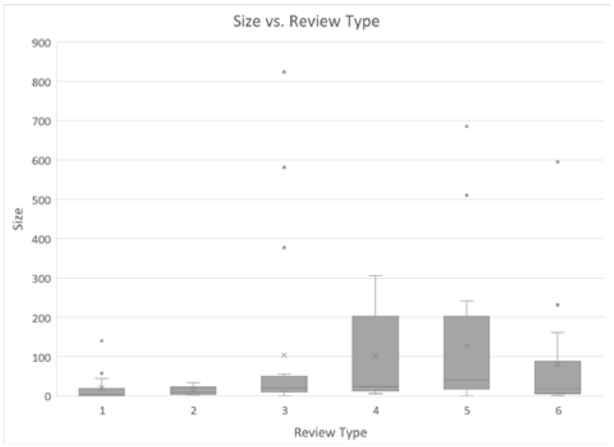

Figure 2: Review size versus type

General results reported in Figure 1 show that document reviews (#4), and requirement reviews (#5) discover more defects than other types of reviews. One reason is that their size is relatively larger. Figure 2 shows that reviews of requirements and documents are the largest reviews that were conducted. Code and configuration accounting reviews were the smallest, on average. While the average design review size was small, there were several design reviews that were very large. Also, a couple of test reviews additionally were very large in relation to their average.

4.3.1. Answers to Q1: cost of FPR

Our observations from the chart in Figure 3 are that code reviews, configuration accounting reviews, and design reviews take the least effort, with relatively low variance; and document reviews and requirement reviews take more effort, with largest variance.

4.3.2. Answers to Q2: effectiveness of FPR

Overall, FPRs are effective in finding defects as indicated by Figure 3 and Figure 4. Requirements and design reviews were the most effective in finding defects in the least amount of time. This tracks with the findings that identifying and eliminating defects is more effective earlier in the life cycle.

4.3.3. Answers to Q3: FPR effectiveness by review type

From the details in Figure 4, we can see that requirements reviews (#5) have the highest effectiveness value of all reviews, followed by design reviews (#3). However, these reviews also had the widest variation, which requires further analysis to determine the source of that variation.

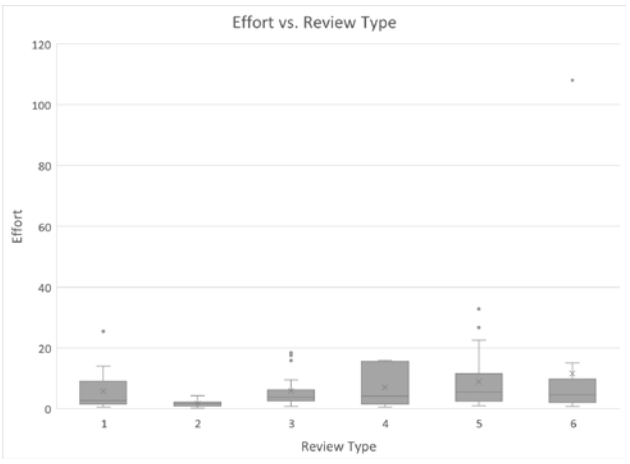

Figure 3: Review effort versus type

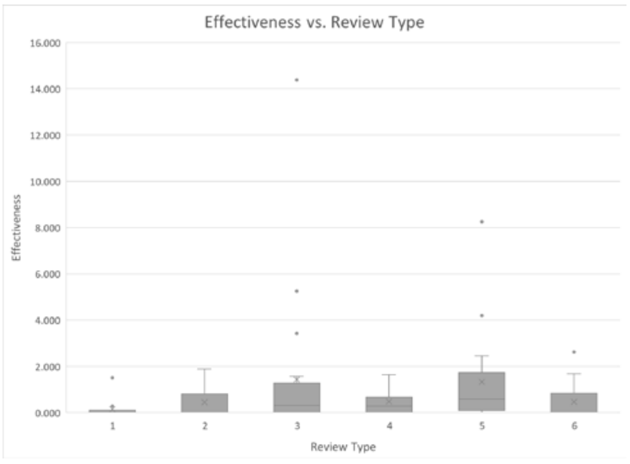

Figure 4: Review effectiveness versus type

5. Evaluation and discussion

5.1. Do higher FPR_Eff scores correlate to fewer failure incidents?

To evaluate the validity of the proposed FPR_Eff metric, we conducted an in-depth investigation as to whether there is correlation between the derived FPR_Eff scores and the number of fielded incidence reports, for which we apply the proposed GQM to a different set of 197 FPRs collected from 8 projects within the same company. The data includes the date, types of the FPRs, size, and the number of defects identified. We aggregated the same types of FPRs in each project, and applied the FPR_Eff metric at the project level. Table 3 summarizes the aggregated FPR_Effs by review type, as indicated by the last column of "SumFPR".

The "Incidents" column corresponds to the number of major incidents due to elapsed defects. As shown in the table, the four projects with high total FPR_Eff scores, i.e. S01588, S01589, S01591, and S01592, correspond to zero incidents. On the other hand, the two projects with the lowest FPR score, i.e. S01593 and S01592, correspond to more incidents, i.e. 4 and 3 respectively, higher than the average of 2 incidents. These results, though preliminary, are very encouraging in demonstrating the usefulness of the FPR score in an industrial project setting, which potentially can be used to facilitate planning decisions on various types of FPRs.

Table 3: FPR effectiveness of larger data set

ID	Requirements Review	Design Review	Code Review	Test Review	Document Review	Configuration Accounting Review	Incidents	SumFPR
S01588	2.79	0.30	0.85	0.05	0.00	0.02	0.00	4.01
S01589	1.23	0.31	0.42	0.05	0.01	0.01	1.00	2.04
S01590	0.62	0.14	0.00	0.07	0.00	0.02	0.00	0.85
S01591	1.89	0.80	0.62	0.01	0.00	0.03	0.00	3.34
S01592	1.85	0.27	0.06	0.04	0.00	0.02	0.00	2.23
S01593	0.00	0.35	0.02	0.00	0.00	0.02	4.00	0.38
S01594	1.43	0.24	0.00	0.05	0.00	0.02	7.00	1.73
S01595	0.00	0.20	0.03	0.02	0.00	0.02	3.00	0.27
Mean	1.23	0.32	0.25	0.04	0.00	0.02	2.00	1.86
Stdev.	0.98	0.20	0.34	0.02	0.00	0.00	2.77	1.35

Nonetheless, there are two projects that demonstrate conflicting results. One is project S01594, which has a FPR_Eff score of 1.73, yet relating to the highest number of 7 incidents. The other is project S01590, with a relative low FPR_Eff score of 0.85, but not associated with any incident. To better understand the underlying reasons, more information will need to be collected and is currently planned as future work.

5.2. Does every formal peer review really need to take place?

This section presents the analyses performed with the defects measure versus the review size (number of requirements) for requirements reviews.

Figure 5 shows the majority of FPRs occurred on a set of less than 50 requirements. This matches the development practices at the organization due to the highly incremental nature of their work. There are then singular FPRs at higher levels of requirements counts.

The plot of the defects found in the requirements FPRs versus the number of requirements in each review along with the logarithmic trend line of the data is shown in Figure 6. The plot shows that the data is clustered in the lower left of the chart because there are more FPRs performed on smaller sets of requirements, which also find lower numbers of defects. However, there are some significant outliers shown, too, which leads to a poor R2 value of the trend line at 0.2549.

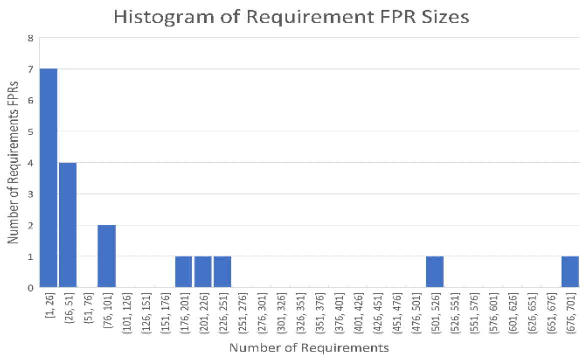

Figure 5: Histogram of requirements review sizes

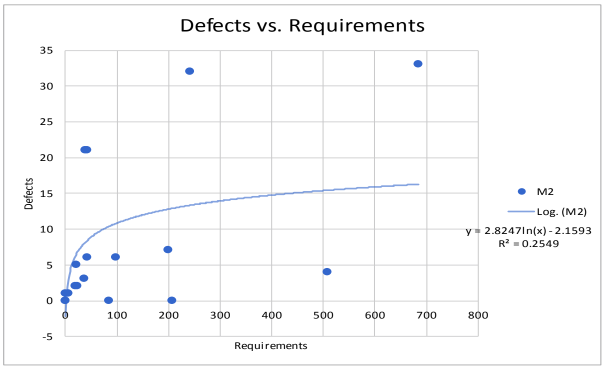

Figure 6: Defects vs. requirements

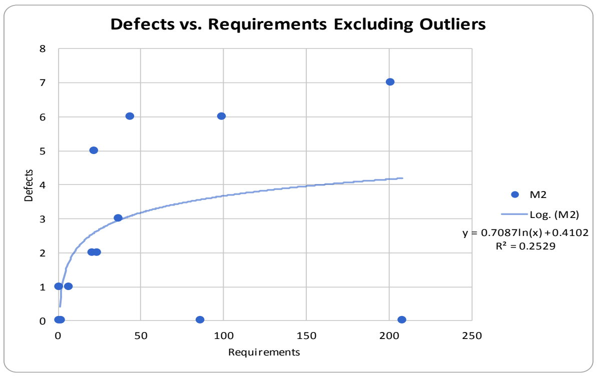

Figure 7: Defects vs. requirements excluding outliers

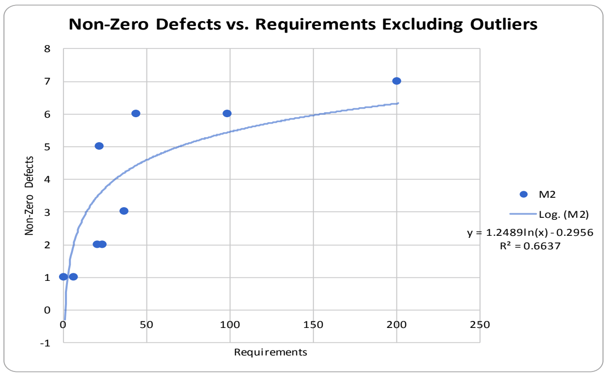

Figure 8: Defects > 0 vs. requirements excluding outliers

The next plot in Figure 7 shows the requirements data excluding the outliers where the number of defects was greater than 20 or the review size was greater than 400. This plot shows most FPRs consisting of less than 50 requirements and the number of defects at 7 or less. However, the R2 of the trend line is slightly less, at 0.2529. This may be due to the reviews along the X axis that had no defects detected. Removing the FPRs that resulted in no defects detected resulted in the plot in Figure 8. This chart shows an improved fit to the logarithmic trend line with a R2 value of 0.6637.

The same data as the previous section was analyzed against a polynomial trend line, which is shown in Figure 9. In this chart we now see a clear determinism from the trend line to the data with a high R2 value. Even though the R2 value is not considered very high (>= 0.80), it is high enough that we can reach some conclusions. First, there is a clear maximum of the trend line at 150 requirements and 7 defects (yellow vertical line). In fact, looking closely at the data points we can conclude that requirements FPRs that consider more than 50 requirements do not identify a significantly higher number of defects (green vertical line). The upper threshold of 50 requirements in a requirements FPR can be used to determine when a FPR needs to be split into multiple sessions of less than 50 requirements each. Second, there is a vague minimum of the data points around the 20 requirements size and 2 defects (red vertical line). The FPRs that were conducted on less than 20 requirements found only 1 defect each, which could be considered non-value added. The lower threshold of 20 requirements can be used to waive the need to perform a FPR, opting instead for other techniques for defect removal (application of standards, informal inspections, etc.).

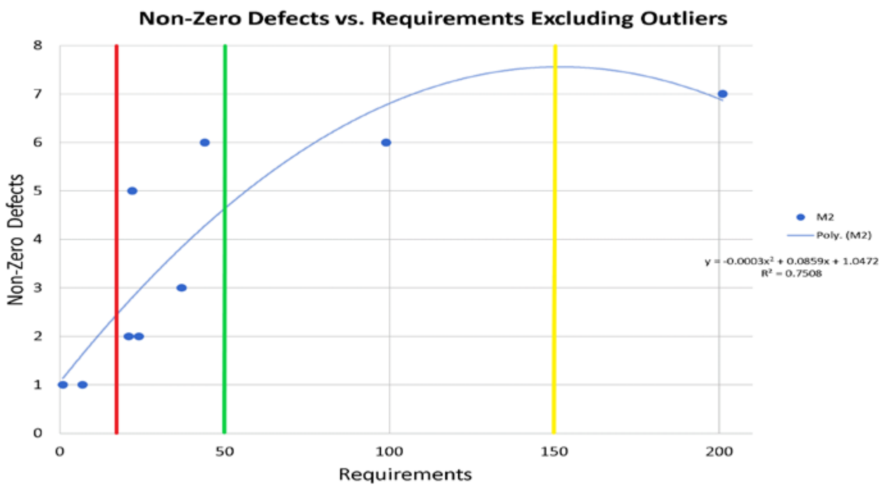

Figure 9: Defects > 0 vs. requirements, excluding outliers with polynomial trend line

Using these lower and upper bounds we find an optimal range of requirements review sizes of between 20 and 50 requirements, which yield valuable defect identification. It is important to note that these conclusions are based on a very small sample size of 9 FPRs. It would be prudent to sample additional FPRs to gain confidence in the thresholds and to refine the conclusions.

6. Conclusions

This paper reported an industrial case study on the effectiveness of formal peer review. Main analysis results on the FPR Effectiveness demonstrate the importance of FPRs earlier in the life cycle. In addition, most of the reviews had an effectiveness of less than 1.0, indicating they were less effective than desired.

During the study, the paper examined the linear, logarithmic and polynomial trend lines for the various types of inspections and found optimal sizes for the FPR types. As an example, we found that 50 is the optimal number of requirements to have in a single requirements FPR. Beyond that number, finding new defects takes more time and is less cost effective. An extension to this paper would expand on the optimal size versus the labor cost to determine the most cost and labor efficient sizes.

The initial set of data is limited to only 100 reviews, 45 of which had no defects. This data was gathered from a single organization's set of programs. A larger research effort should use more data collected from the embedded system development industry and open source community to analyze how effective FPRs in general.

An interesting set of data was discovered when we looked at the number of defects found versus the labor cost for the review. We could optimize review sizes to uncover defects most efficiently for the labor cost.

The effectiveness of FPRs follows the standard life cycle model where they are most effective between the requirements phase and code development. After that, FPRs are less effective in finding defects.

References

[1] Basili, Victor; Gianluigi Caldiera; H. Dieter Rombach (1994) The Goal Question Metric Approach. https://www.cs.umd.edu/~mvz/handouts/gqm.pdf

[2] Cheylene T (2015) How to Classify Bug Severity in Your Bug Reports. https://leantesting.com/how-to-classify-bug-severity/

[3] Dabney, Jim; Dick, Brandon; Haskins, Bill; Lovell, Randy; Moroney, Gregory; Stecklein, Jonette M. (2004-06-19), Error Cost Escalation Through the Project Life Cycle, 14th Annual International Symposium of the International Council on Systems Engineering (INCOSE) Foundation. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100036670.pdf

[4] Hollenbach, Craig, (2003-11-19), Quantitatively Measured Process Improvements at Northrup Grumman IT. http://seir.sei.cmu.edu/cmmiresearch/results/pdfs/2003-CMMI-018.pdf

[5] Weigers, Karl E. (2002), Peer Reviews in Software, A Practical Guide

[6] Humphrey, Watts; Chick, Timothy; Nichols, William; Pomeroy-Huff, Marsha (2010-07), Team Software Process (TSP) Body of Knowledge (BOK), p17. https://resources.sei.cmu.edu/asset_files/TechnicalReport/2010_005_001_15254.pdf

[7] M. Fagan. Design and Code Inspections to Reduce Errors in Program Development. IBM Systems Journal, 15(3):182–211, 1976.

[8] L. G. Votta. Does every inspection need a meeting? SIGSOFT Softw. Eng. Notes, 18(5):107–114, 1993.

[9] P. Rigby, B. Cleary, F. Painchaud, M. A. Storey, and D. German, "Contemporary Peer Review in Action: Lessons from Open Source Development," IEEE Software, vol. 29, no. 6, pp. 56–61, Nov. 2012.

[10] "Safety-critical software development surprisingly short on standards, analysis, and review." [Online]. Available: http://www.embedded-computing.com/embedded-computing-design/safety-critical-software-development-surprisingly-short-on-standards-analysis-and-review. [Accessed: 12-Jul-2018].

[11] IEEE Std 1044-2009, IEEE Standard Classification for Software Anomalies.

[12] RTCA (2011) DO-178C Software Considerations in Airborne Systems and Equipment Certification.

[13] Mokhtar, Siti Osnita; Nordin, Rosmawati; Aziz, Zalilah Abd; Rawi, Rashidah Md (2017) Issues and Challenges of Requirements Review in the Industry. http://www.indjst.org/index.php/indjst/article/view/110613

[14] Johnson, Philip M.; Tjahjono, Danu; (1998) Does Every Inspection Really Need a Meeting? https://link.springer.com/article/10.1023%2FA%3A1009787822215

[15] Sadowski, Caitlin; Soderberg, Emma; Church, Luke; Spiko, Michal; Bacchelli, Alberto (2018) Modern Code Review: A Case Study at Google. http://dl.acm.org/citation.cfm?doid=3183519.3183525

[16] McIntosh Shane; Kamei Yasutaka; Adams, Bram; Hassan Ahmed E. (2015) An Empirical Study of the Impact of Modern Code Review Practices on Software Quality. http://dl.acm.org/citation.cfm?doid=3183519.3183525

[17] Soilingen, Rini va; Berghout, Egon (1999) The Goal/Question/Metric Method: a practical guide for quality improvement of software. https://courses.cs.ut.ee/MTAT.03.243/2015_spring/uploads/Main/GQM_book.pdf

[18] Chrissis, Mary Beth; Konrad, Mike; Shrum, Sandy (2006) CMMI: Guidelines for Process Integration and Product Improvement (2nd Edition).

[19] ISO/IEC 90003:2014, Software Engineering – Guidelines for the application of ISO 9001:2008 to computer software.